Holonic-based Control System for Automated Material Handling Systems

Radu F. Babiceanu

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Industrial and Systems Engineering

Dr. F. Frank Chen, Chair
Dr. Robert H. Sturges, Jr.
Dr. Subhash C. Sarin
Dr. C. Patrick Koelling
Dr. Philip Y. Huang

July 12, 2005
Blacksburg, Virginia

Keywords: Holonic manufacturing systems, Material handling systems, Real-time scheduling

Holonic-based Control System for Automated Material Handling Systems

Radu F. Babiceanu

ABSTRACT

In real-word manufacturing environments, finding the right job sequences and their associated schedules when resource, precedence, and timing constraints are imposed is a difficult task. For most practical problems classical scheduling easily leads to an exponential growth in the number of possible schedules. Moreover, a decision time period of hours or even minutes is too long. Good solutions are often needed in real-time. The problem becomes even more complicated if changes, such as new orders or resource breakdowns, occur within the manufacturing system. One approach to overcome the challenges of solving classical scheduling problems is the use of distributed schemes such as agent or holonic-based control architectures.

This dissertation presents an innovative control architecture that uses the holonic concept, capable of delivering good solutions when applied in dynamic environments. The general holonic control framework presented in this research has specific characteristics not found in others reported so far. Using a modular approach it takes into account all the categories of hardware and software resources of a manufacturing system. Due to its modularity, the holonic control framework can be used for assigning and scheduling different task types, separately or simultaneously. Thus, it can be used not only for assigning and scheduling transport tasks, but also for finding feasible solutions to the job assignment and scheduling of processing tasks, or to better utilize the auxiliary equipment and devices in a manufacturing system.

In the holonic system, under real-time constraints, a feasible schedule for the material handling resources emerges from the combination of individual holons' schedules. Internal evaluation algorithms and coordination mechanisms between the entities in the architecture form the basis for the resultant schedules. The experimental results obtained show a percentage difference between the makespan values obtained using the holonic scheduling approach and the optimal values of under seven percent. Since current control systems in use in industry lack the ability to adapt to dynamic manufacturing environments, the holonic architecture designed and the tests performed in this research could be a part in the effort to build the foundations for the control systems of the next generation manufacturing systems.

Table of Contents

Nomenclature

ACL: Agent Communication Language

AG: Applicability Groups

AGV: Automated Guided Vehicles

AI: Artificial Intelligence

AMHS: Automated Material Handling Systems

ASRS: Automated Storage and Retrieval Systems

B&B: Branch and Bound

CBD: Component-Based Development

CI: Confidence Interval

CIm: Communication Interface sub-module

CIM: Computer Integrated Manufacturing

CMm: Cooperation Mechanisms sub-module

CM: Coordination Module

CNP: Contract Net Protocol

CPU: Central Processing Unit

DAI: Distributed Artificial Intelligence

EBFS: Enhanced Best-First Search

EH: Equipment Holon

EM: Evaluation Module

EtoPlan: Engineering-to-order Planning

FIFO: First-In-First-Out

FIPA: Foundation for Intelligent Physical Agents

FMC: Flexible Manufacturing Cell

FMS: Flexible Manufacturing Systems

FrMS: Fractal Manufacturing Systems

GS: Global Scheduler

HLC: High-Level Control

HMS: Holonic Manufacturing Systems

HMSC: Holonic Manufacturing Systems Consortium

ICS: Intelligent Control System

ID: Identifier

IDM: Internal Database Module

IEC: International Electrotechnical Commission

IMS: Intelligent Manufacturing Systems

IS: Individual Schedule

ISO: International Standards Organization

IT: Information Technology

JC: Job Characteristics

JCTH: Job Completion Time Heuristic

LB: Lower Bound

LLC: Low-Level Control

LM: Learning Module

LSH: Local Search Heuristic

L/U: Loading/Unloading

MAL: Multi-Agent Learning

MAS: Multi-Agent Systems

MASCADA: Manufacturing Control System Capable of Managing Production Change and Disturbances

McH: Machine Holon

MCTH: Machine Completion Time Heuristic

ME: Member Enterprises

MH: Material Handling

MHH: Material Handling Holon

MHH-ECT: Material Handling Holon - Expected Completion Time

MHH-HOP: Material Handling Holon - Help Offer Price

MHH-OP: Material Handling Holon - Offered Price

MH-RCVH: Material Handling - Resource Constraint Violation Heuristic

ML: Machine Learning

MTBF: Mean Time Between Failures

MTTR: Mean Time To Repair

NGM: Next Generation Manufacturing

NIST: National Institute of Standards and Technology

OEH: Open-Ended Hierarchy

OH: Order Holon

OH-ECT: Order Holon - Expected Completion Time

OH-OP: Order Holon - Offered Price

OOP: Object-Oriented Programming

PC: Personal Computer

PROSA: Product Resource Order Staff Architecture

PS: Problem Solver

RE: Recipient

RH: Resource Holon

RL: Reinforcement Learning

RMS: Reconfigurable Manufacturing Systems

SBH: Shifting Bottleneck Heuristic

SAL: Single-Agent Learning

SC: Scheduling Process Characteristics

SE: Sender

SFC: Shop Floor Control

SFCS: Shop Floor Control System

SGM: Schedule Generation Mechanisms sub-module

SLS: System Level Schedule

SMD: System Monitoring and Database

SME: Small to Medium Enterprises

TS: Temporary Storage

UF: Utility Function

UML: Unified Modeling Language

VCM2020: Visionary Manufacturing Challenges for 2020

VE: Virtual Enterprises

List of Tables

List of Figures

Acknowledgements

I would like to express my appreciation to my graduate advisor and committee chair, Professor F. Frank Chen, for his excellent guidance and support throughout all the steps of my Ph.D. program and for his valuable help and expert advice in carrying out this dissertation. I would like to thank Professor Robert H. Sturges for the insightful suggestions and comments he gave me every time we discussed my research progress. Very special thanks go to Professor Subhash C. Sarin who is an outstanding professor. The knowledge I gained by attending his courses made this dissertation possible. A very warm thank you to Professor C. Patrick Koelling and Professor Philip Y. Huang for their assistance and guidance, and especially, for their suggestions and generous help in the last stages of my dissertation. I thank all the other professors in the Industrial and Systems Engineering Department whose courses I took during my Ph.D. program here at Virginia Tech. There is a contribution coming from each of them in this dissertation.

I would like also to thank Taylor & Francis and Elsevier publishing groups for their kind policy of allowing authors to include previous copyrighted materials in their thesis or dissertations provided that they are not to be published commercially, for which the above mentioned publishing groups own the copyright. I made use of this policy and include in this dissertation parts of the articles published during my Ph.D. studies in *International Journal of Production Research* (a Taylor & Francis Group Ltd. journal) and *Robotics and Computer-Integrated Manufacturing* (an Elsevier B.V. journal).

PART I

GENERAL INFORMATION

1. Introduction

Over the last two decades, a vast amount of research has been carried out, in both academia and industry, to improve the performance of manufacturing systems and their responsiveness to changing customer requirements. In addition to improve manufacturing systems productivity and their capabilities to deliver quality and low priced products, in recent years, new requirements have been imposed on the operation of manufacturing systems. Examples of such new requirements include capability to respond in real-time to any disturbance in the system, true fault-tolerance, and hardware/software reconfigurability in reaction to changing environment. Along with other new concepts, the holonic manufacturing systems (HMS) concept is considered a promising solution for next generation manufacturing systems.

1.1. Characteristics of the Global Manufacturing Environment

Global economy, a reality that cannot be ignored anymore by any profit-oriented organization, is forcing the manufacturing sector to continuously adapt to the changing conditions existing in the market. This dynamic manufacturing environment is characterized by a greater variety of products, large fluctuations in demand, shorter life-cycles of the products, and increased customer expectations in terms of quality and delivery time. In the same time, the rapidly changing environment protection regulations and the newly developed processes and technologies are also imposing new requirements on the performance of existing manufacturing systems. Moreover, the classical problems associated with the manufacturing environment such as frequently changing of production orders and delivery dates, unavailable raw materials, or quality problems that stop or delay processing in downstream departments are still creating disturbances within manufacturing systems.

1.2. Automated Material Handling Systems

In manufacturing environments, material handling (MH) is one of the shop floor processes that need to be performed in order to move materials and products from one station to

another until reaching the delivery department. All modern automated manufacturing systems need a MH system capable of moving the materials efficiently throughout the entire production process. Although MH is typically considered a process that is not adding value to the finished product, it can constitute a substantial portion of the whole production time, use significant factory space and thereby add considerably to the finished product cost. In modern manufacturing systems, MH involves not only moving and storing raw materials and finished products, it also involves protecting and controlling materials during processing and transportation, and aids in the integration of the machines and other pieces of equipment to obtain a fluent flow of parts throughout the entire manufacturing system. Therefore, an efficient MH system is essential for every manufacturing system.

Automated material handling systems (AMHS) include robotics systems, automated guided vehicle (AGV) systems, automated storage and retrieval systems (ASRS), automated transporting systems such as conveyors, gantry cranes, etc. The subject of this research is the MH system that moves materials among the processing machines within a manufacturing system, such as robotics systems or AGV systems. Traditionally, MH systems are controlled through a central computer and the individual MH resources schedules are pre-established before the actual operation of the system. Many MH real-world cases involve multiple transporters, machines, and batches of parts to be routed between these machines. This kind of combinatorial problems belong to the scheduling/dispatching operations research problems. Most of them are so complicated, due to the immensely large number of possible combinations, that even the most sophisticated existing computers cannot solve them practically.

2. Research Motivations, Objectives and Contributions

The business conditions imposed by the global manufacturing environment require manufacturing companies to react to any changes that might occur in a rapid manner and with low costs, and to have a manufacturing system capable of increasing/decreasing both the product variety and production quantity without causing major disorders in the production process. A manufacturing system capable of performing these changes efficiently needs a control system flexible enough to run it without significant delays in operation, regardless the changing manufacturing environment.

2.1. Previous Approaches to Manufacturing Systems Research and Implementation

To deal with the increasing challenges faced by manufacturing sector appeared after the mass production concept reached its peak and started to decline in the 70's, manufacturing research looked for solutions in the newly emerging information technology (IT) area. Using IT tools, the computer integrated manufacturing (CIM) concept attempted to integrate all activities within a manufacturing facility, and was the main concept that was believed, in the 80's and early 90's, to be capable of delivering the best solution for all manufacturing problems. However, in most CIM implementation projects, it was "wrongly assumed that the organizational structure and the procedures of a company can automatically be improved by the implementation of computer aided information systems" (Warnecke, 1993). So, contrary to what was expected, CIM implementations resulted in rigid centralized systems, incapable of delivering the expected flexibility in response to changes of any nature.

Moreover, CIM implementations aiming to totally integrate and optimize all activities within a plant usually requested very large investments, making implementation prohibitive for many companies. Integration of a large number of different hardware and software platforms existing within a manufacturing plant was another problem lacking viable solution within CIM projects. As a consequence, many CIM implementation projects failed to deliver the expected improvements, so in the 90's CIM ceased to be viewed as the answer to all operational

improvements needed in manufacturing environments. Researchers started to look to other approaches to improve manufacturing operations. More than ever, due to the clear shift towards global operations in the manufacturing area, new solutions were needed to cope with the unprecedented number of changes that took place in manufacturing environments and the increased customer expectations.

2.2. Requirements for Next Generation Manufacturing Systems

The increasing impact of global economy is changing the traditional way a manufacturing company used to be managed. Real-time response to any changes in shop floor operations, agility in satisfying any customer requests, and reconfigurability in both, software and hardware equipment, are likely to become essential characteristics for next generation manufacturing systems.

Large scale projects carried out during the 90's, such as Next Generation Manufacturing (NGM) and Visionary Manufacturing Challenges for 2020 (VMC2020), aimed at giving a framework to be followed by manufacturers in the United States to become successful in the global economy of the 21st century. A NGM company should "employ an ever-growing knowledge base of manufacturing science to implement reconfigurable, scalable, cost-effective manufacturing processes, equipment, and plants that adapt rapidly to specific production needs" (Agility Forum, 1997). One of the grand challenges identified in the VMC2020 report is to "reconfigure manufacturing enterprises rapidly in response to changing needs and opportunities" (National Research Council, 1998). To obtain such reconfiguration, the report mentions the need for development of self-organizing manufacturing systems based on autonomous modules, biotechnology, and holonic concept.

According to Shen *et al.* (2001), the next generation manufacturing systems will be more strongly time-oriented while still focusing on cost and quality. Requirements for realizing the time-oriented manufacturing systems include enterprise integration and cooperation, agility in responding to customer requests, scalability and fault-tolerance capabilities. While the first two requirements are driven by the global competition environment, the other two are characteristics needed by next generation manufacturing systems for coping with unexpected events.

Scalability allows manufacturing systems to incorporate additional resources, either

hardware or software, when needed and without disrupting the previously established architectures. This is not possible in the existing centralized or hierarchical systems, but becomes feasible when talking about the holonic or agent-based systems. The other requirement, fault-tolerance is the capability of a system to recover from failures within acceptable time limits and minimize the impact on the working environment. Once again, the existing centralized and hierarchical manufacturing control systems do not have this capability, the lack of fault-tolerance being one of their major drawbacks. On the other hand, the holonic systems are designed for fault-tolerance by incorporating specific information exchange and recovery protocols within their control software.

2.3. Research Objectives

This dissertation research tries to integrate the holonic manufacturing control concept in the operations procedures of an AMHS for the purpose of obtaining a better system performance. Because of their rigid structure, the existing MH systems are difficult to adapt to the requirements set on future manufacturing. A MH system equipped with a holonic control system may handle changes efficiently and be able to react in acceptable time to the unpredicted disturbances inherent during the operation of a manufacturing system.

The MH holonic control system will use the general holonic control framework designed for the entire manufacturing system. All the physical resources and the control units existing in a manufacturing system are included in the general holonic control framework. However, because the subject of this research is to control the devices and equipment that move materials within a manufacturing system, the scheduling algorithms and the decision-making process are developed only for controlling the MH system. As a consequence, experimental tests are also carried out only for controlling the MH system.

2.4. Contributions to the Manufacturing Systems Field

This dissertation work proposes an innovative solution for the real-time control of manufacturing resources working in dynamic environments. It includes the necessary background information and a comprehensive state-of-the-art literature review of the holonic-

based manufacturing control research in an attempt to bring together the key issues in the theory and development of holonic control systems. The literature review includes the motivations for searching other manufacturing control solutions, the origins of the holonic concept, its first applications in manufacturing, all the important considerations in designing holonic systems, and an up-to-date review of the applications and implementations of the HMS concept. At the point when this research was started, there was no such comprehensive research review to present all aspects of the theory, development and implementation of the holonic concept in manufacturing.

Even there are other holonic control approaches presented in the literature, the general holonic control framework presented in this dissertation has specific characteristics not found in others reported so far. At the general level, using a modular approach, it takes into account all the categories of hardware and software resources of a manufacturing system. Due to its modularity, the holonic control framework can be used for assigning and scheduling different task types, separately or simultaneously. Thus, it can be used not only for assigning and scheduling transport tasks, but also for finding better solutions to the job assignment and scheduling of processing tasks, to better utilize the auxiliary equipment and devices in a manufacturing system, or a combination of these. Moreover, no research was identified in which the proposed system can provide both the results obtained using the decentralized approach and reliable means to evaluate them.

The holonic architecture designed in this research addresses specific requirements for next generation manufacturing systems, including reliability, real-time scheduling, fault-tolerance, hardware reconfigurability and learning capabilities, as presented below.

- *Reliability:* Due to their reduced computational complexity, all the internal holon evaluation and allocation algorithms and cooperation capabilities lead to simple decision-making processes. Regardless of the dimension of the problem the holonic system is capable of delivering feasible solutions.
- *Real-time scheduling:* By concurrently executing interrelated real-time processes based on the evaluation, allocation, and execution algorithms developed, the MH resource assignment process is performed in real-time in the proposed holonic system.

- *Fault-tolerance:* In real-world manufacturing environments hardware resources may fail during operation. Using message exchange and real-time re-scheduling for the job affected by the breakdown, the holonic system is able to accommodate it to another MH resource.
- *Hardware reconfigurability:* Defined as the capability of the system to easily add or remove resources during operation, MH hardware reconfigurability is supported in the proposed holonic system through specific message communication.
- *Learning:* The entities in the architecture are capable of learning emergent relationships from the past and from the future during the operation of the system. The learning module embedded in the internal structure of the entities may provide faster responses during MH task assignment operation.

Internal evaluation and allocation algorithms and specific cooperation mechanisms between holons in the architecture are the basis for the resultant emergent schedules. These evaluation algorithms and allocation are developed using several scenarios that take into account uncertainties that typically exist in dynamic manufacturing environments, such as new orders entering the system. The resulting schedule serves as a reactive schedule and is updated in real-time when conditions in the system change. Due to the fact that in the large number of existing manufacturing scheduling problems, only a few can be solved optimally using deterministic approaches, the lack of known optimality of the resultant emergent schedule obtained using the decentralized approach is not an important issue overall. Even when an optimal solution can be derived using a centralized system, the resultant schedule may need to be revised or even discarded, due to some changes in the system that occurred after it was developed and thus, make the initial schedule invalid.

Because of their reduced computational complexity, the algorithms developed for task assignment can be applied for large, industry scale problems. Except for particular problems, where optimal algorithms are developed, the proposed holonic system is likely to provide better real-time dispatching of MH transporters for a large percent of the industry problems. Moreover, the MH holonic control architecture could be used for scheduling transport or moving tasks by any organization that performs such activities. Warehousing, docks loading and unloading,

supply chain activities are just a few of the possible applications. Since there are no reported results of improvements made by learning mechanisms associated with MH holonic control systems, the addition of a learning module investigated in this research may be an important starting point for the application of other learning mechanisms, found in the machine learning literature, in already developed holonic structures.

PART II

BACKGROUND INFORMATION AND LITERATURE REVIEW

3. Manufacturing Control Architectures

The architecture of a system is defined by Bauer *et al.* (1994), as "a framework or a set of rules and guidelines for managing the development and operation of complex systems." This definition applies very well to manufacturing control architectures where the objective is to develop a framework and associated set of rules, usually in the form of software programs, to operate a manufacturing system.

The terms manufacturing control and shop floor control (SFC) are used interchangeably in the literature and they do have the same objectives, the only difference comes from the area of application. While SFC is strictly referring to the activities on a single shop floor, manufacturing control could cover a larger area such as two or more departments (shop floors) or even all the production departments in a manufacturing facility. The main functions of SFC are, to ensure that all the production orders received from the production planning department are assigned to the existing resources and executed in time, and to monitor the status of all the resources on the shop floor. Since controlling MH equipment is an integral part of the control of the entire manufacturing system, existing control architectures are used to not only control processing machines but also to control the equipment that moves materials between processing machines.

The published literature in the manufacturing control area (Dilts *et al.,* 1991, Overmars and Toncich, 1996) considers four basic types, namely centralized, hierarchical, modified hierarchical, and heterarchical control architectures, as presented in Figure 3.1. The arrows in this figure indicate the direction in which the control commands flow through the system. A discussion of the four basic types of manufacturing control architectures is presented below, including examples of the most used control architectures.

3.1. Centralized Control Architectures

In the centralized system presented in Figure 3.1a, there is a single control unit which is making all the decisions regarding the flow of materials in the system. The centralized system presents the advantage of having a simple architecture and the possibility of global optimization. However, this architecture presents some big drawbacks such as slow speed of response when the system has a large number of resources, difficulty in making any changes to the control

software, and no fault-tolerance (i.e., when the central computer goes down, the entire system goes down, Overmars and Toncich, 1996).

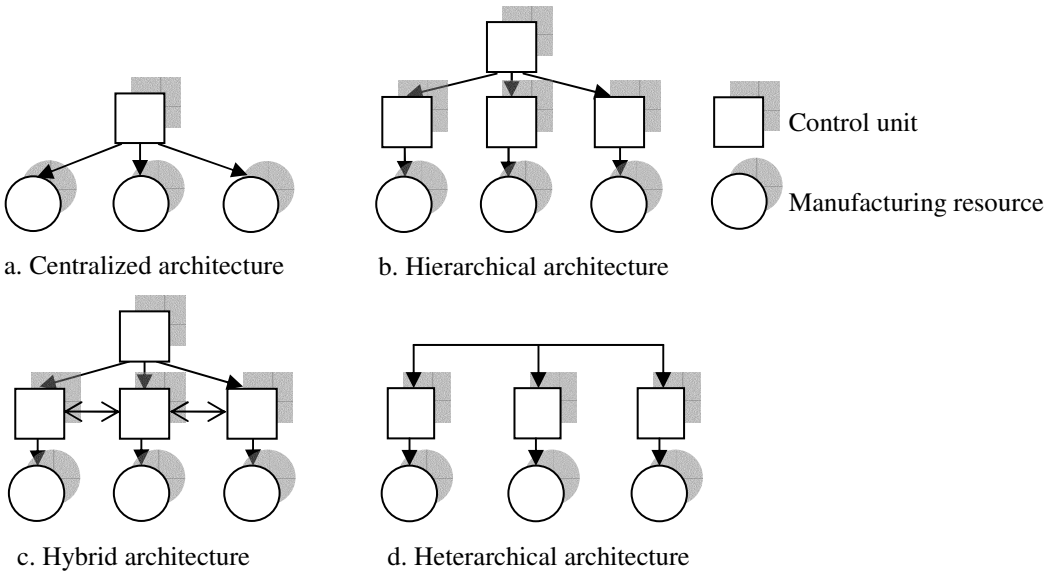

Figure 3.1. Manufacturing control architectures. Adapted from Overmars and Toncich (1996) with the kind permission of Springer Science and Business Media[1].

3.1.1. Example of Centralized Control Architectures

The centralized architecture has been used extensively for manufacturing control in flexible manufacturing cells or systems, and other manufacturing applications. A good example of a manufacturing system run by a centralized control architecture is the flexible manufacturing cell (FMC) installed in Virginia Tech's Flexible Manufacturing Systems (FMS) Lab. A single computer is controlling all the equipment and devices of the cell formed by four processing machines, two MH robots and other devices, which include conveyors, measuring devices, powered rotational buffers, and a quality checking system, as presented in Figure 3.2.

---

[1]Originally published by Kluwer Academic Publishers in *The International Journal of Flexible Manufacturing Systems* 8 (3), figure 1 at page 264. The reference for the original material is as follows: Overmars, A. H. and Toncich, D. J., Hybrid FMS control architectures based on holonic principles. *The International Journal of Flexible Manufacturing Systems,* 8 (3), 263-278, 1996.

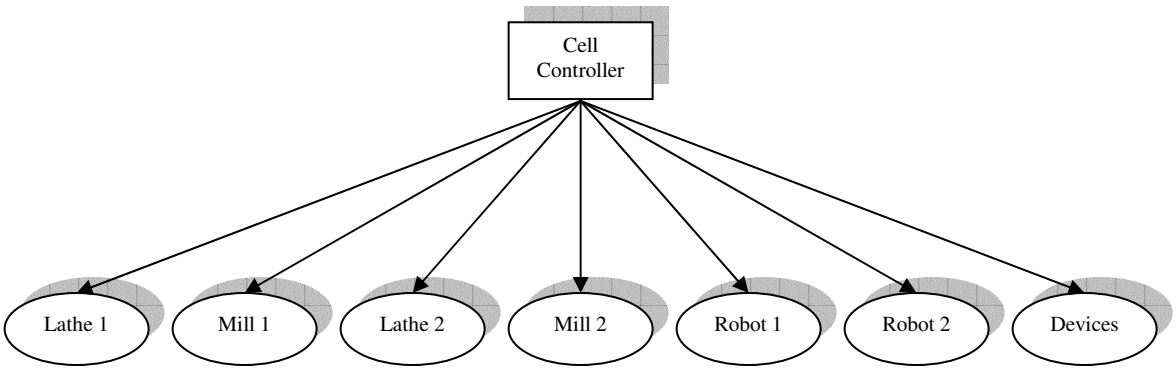

Figure 3.2. Centralized control architecture of the FMC installed in Virginia Tech's FMS Lab.

3.2. Hierarchical Control Architectures

The hierarchical architecture, presented schematic in Figure 3.1b, is based on a top-down approach, where the flow of commands is coming from the higher-level controllers to the lower-level controllers, assuming a deterministic behavior of the system. The centralized architecture is actually a particular case of the hierarchical architecture having only one level of control. Because of the master-slave type relationship between control units at adjacent hierarchy levels, the architecture is easy to understand, and the response time is shorter than in the case of centralized architecture.

Even though the hierarchical system has the possibility to obtain global optimization it also has major disadvantages. The inherent uncertainty of the real-world systems is not taken in consideration. When the initial conditions for which the system was built, as in the case of disturbances such as machine breakdowns are no longer valid, the performance of the system deteriorates drastically. Making any changes in the hardware/software of the system is a difficult process and it can be done only by shutting down the system and updating all the levels of the hierarchy with the new information. The lack of fault-tolerance makes the hierarchical architecture difficult to operate in the case of any machine or equipment breakdown.

Duffie (1990) observed that "the complexity of CIM systems with hierarchical architectures grows rapidly with size, resulting in accompanying high costs of development, implementation, operation, maintenance, and modification." Furthermore, considering the structure of the systems designed based on hierarchical architectures, Duffie noticed that it

"becomes fixed early in their development, making subsequent unforeseen modifications difficult for systems beyond a certain complexity."

Hatvany (1985) stated that "highly centralized and hierarchical ordered systems tend to be rigid, constrained by their very formalism to follow predetermined courses of action." While the idea to integrate all activities seemed interesting, the concept is highly hierarchical and the implementations resulted were very rigid in respect to any change that might occur. The majority of the CIM control architectures were hierarchical, and their implementations were lacking reactivity to disturbances.

3.2.1. Examples of Hierarchical Control Architectures

Hierarchical architectures are the most common control architectures in use in industry. The National Institute of Standards and Technology (NIST) of the United States, and the International Standards Organization (ISO) have developed general hierarchical control models to give the industry tools to develop more detailed control architectures based on the hierarchical control approach. NIST developed the general hierarchical model of a manufacturing facility presented in Figure 3.3 (Bauer *et al.,* 1994) comprising five levels: facility, shop, cell, workstation, and equipment. Considering the schematic representation of the hierarchical architecture from Figure 3.1b and the NIST hierarchy, an illustration of how the control tasks are passed down within a manufacturing facility from the highest level to the bottom is presented schematic in Figure 3.4.

The ISO developed a standard for a hierarchical architecture, called Factory Automation Model, formed by six level of hierarchy, starting from the enterprise level, going through the facility/plant, section/area, cell, and station levels, and then, finally to the equipment level (Bauer *et al.,* 1994). Each of these levels is responsible for specific tasks and operates at decreasing planning time-horizons from months at the enterprise level to real-time at the equipment level.

Based on NIST general hierarchical framework, several control architectures were developed. Two of the most used are the Production Activity Control, located at the cell level, and Factory Coordination, located at the factory level (Bauer *et al.,* 1994). Other hierarchical control architectures presented in the literature are Manufacturing Systems Integration, Real-Time Control System, and Automated Research Facility (Joshi and Smith, 1994).

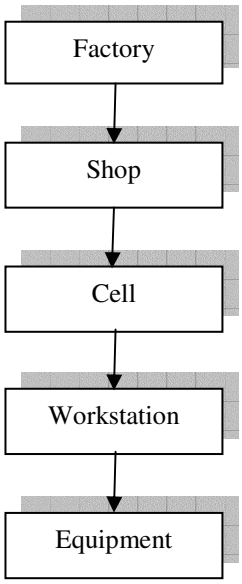

Figure 3.3. The NIST manufacturing control hierarchy. Adapted from Bauer *et al.* (1994) with the kind permission of Springer Science and Business Media[2].

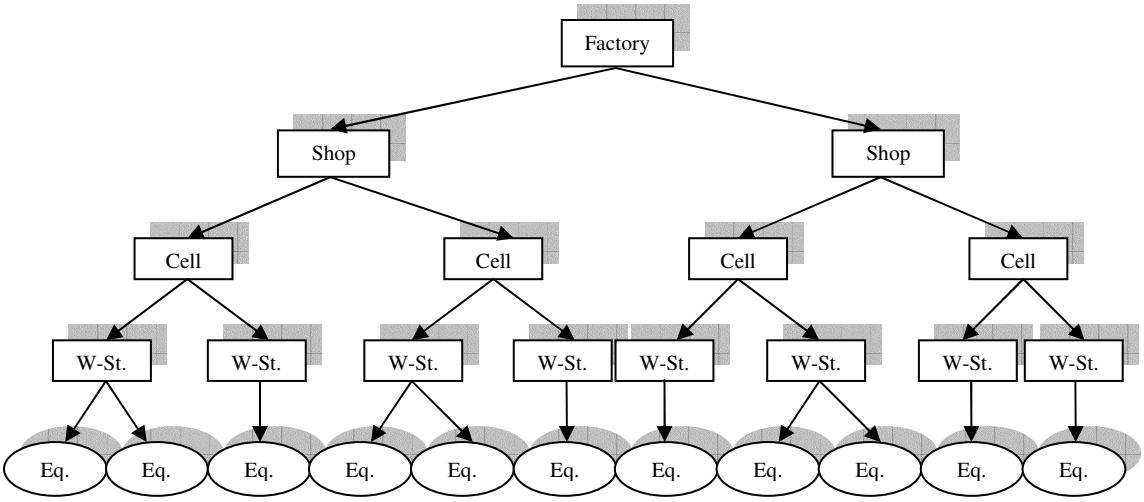

Figure 3.4. A hierarchical architecture.

---

[2]Originally published by Chapman & Hall in *Shop floor control systems - from design to implementation,* chapter 6, figure 1.5 at page 28. The reference for the original material is as follows: Bauer, A., Bowden, R., Browne, J., Duggan, J. and Lyons, G., *Shop floor control systems - from design to implementation,* Chapman & Hall, London, 1994.

3.3. Modified Hierarchical Control Architectures

A further step in decentralizing manufacturing control architectures was the introduction of the modified hierarchical (hybrid) control architectures, which allows horizontal flow of information among the lower level controllers. This architecture, presented in Figure 3.1c, is mainly derived from the hierarchical architecture, yet it allows cooperation and sharing of information among the controllers on the same level of hierarchy, contrary to the proper hierarchical form where the information is flowing only vertically. The supervisor initiates all the activities, and then the subordinates cooperate to perform them. If there are changes made to the initial conditions, the supervisor takes the control, so that the lower level controller decision is limited only to normal operation conditions.

Since the involvement of the lower level controllers in decision-making is limited, the hybrid architecture presents both the advantages and disadvantages of the hierarchical architecture. However, some differentiation exists since the modified hierarchical architecture allows a relatively local autonomy at the lower levels of the hierarchy and decreases the computational load on the master controllers, but in the same time, it is more complex to design and implement than the hierarchical architecture.

3.4. Heterarchical Control Architectures

Traditional centralized and hierarchical control architectures presented above cannot deliver a rapid response to either external stimuli such as changes in production orders or internal stimuli such as breakdowns within the manufacturing system. Their control software is developed for dealing with pre-established situations, and any change in manufacturing conditions could result in large delays and even closing down the whole system. An important step in developing more flexible control systems in regard to changing conditions was the introduction of the heterarchical control architectures, which allows only horizontal flow of information.

The heterarchical architecture, presented in Figure 3.1d, is a totally decentralized architecture, formed by a group of independent entities called agents that have their own internal algorithms embedded in a control unit. The allocation of tasks in the heterarchical architecture is

performed by exchanging information among the agents in the architecture and using a specified mechanism which can be direct bidding, blackboard-based, market-based or some other allocation mechanism. There is no master-slave relationship like in the architectures presented above. All the agents, including the manager of a particular order, can be involved in contracting a specific order. Once the winning agent finishes the task, it automatically becomes the new manger for the incoming task. Due to the decentralized structure the agents have full local autonomy and the system can react promptly to any change made in the system.

Other advantages include reduced software complexity, reconfigurability and scalability, as agents can be added or removed from the architecture in a simple way. However, because the behavior of an order is dependent on the number and characteristics of other orders in the system, it is impossible to seek global control optimization and the performance of the system is thereby unpredictable. When the number of entities in the architecture is large, problems related to the network capacity of carrying a large number of messages, possibly at the same time, are likely to appear. Thus, there is a need to standardize the tools used in the information exchange across the decentralized architecture.

3.4.1. Examples of Heterarchical Control Architectures

As Veermani *et al.* (1993) stated, a heterarchical control system "presumes a high level of intelligence among system entities to permit autonomous decision making." They also noticed that the heterarchical approach is better in accommodating system reconfigurations, and because of the independence of system entities, is more robust to failures in the system than a hierarchical architecture. Comparing the characteristics of hierarchical and heterarchical architectures for the same shop floor arrangement, Veermani *et al.* (1993) argued that with the increase in size and complexity of manufacturing systems, hierarchical control structures become prohibitively complex and unreliable. Their proposed decentralized architecture for a shop floor with six machines is presented in Figure 3.5.

Duffie (1990) noticed that "heterarchical control architectures offer prospects of reduced complexity by localizing information and control, reduced software development costs by eliminating supervisory levels, higher maintainability and modifiability due to improved modularity and self-configurability, and improved reliability by taking a fault-tolerant approach

rather than a fault-free approach." He also developed an experimental architecture for an automated mold production system and concluded that appropriate heuristics and optimization algorithms need to be developed for scheduling heterarchical systems which do not rely on entities having a global view of the entire system.

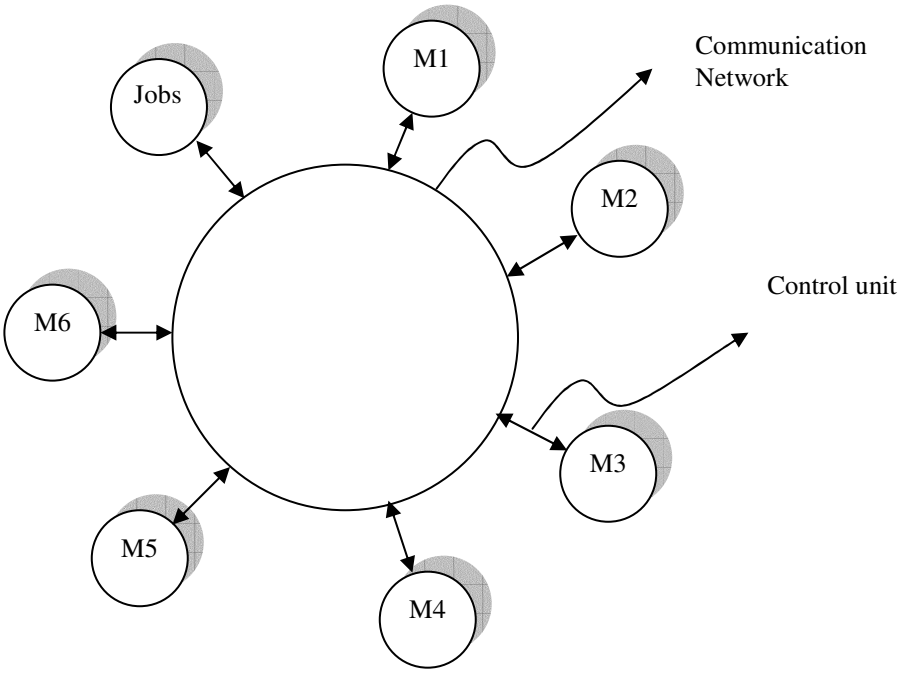

Figure 3.5. Heterarchical control architecture for a shop with six machines. Adapted from Veeramani *et al.* (1993) with the kind permission of Elsevier[3].

As a conclusion, using a completely distributed architecture, heterarchical systems have the advantage of reduced complexity, high flexibility, and robustness against disturbances. However, without any global view entity, these systems are likely to exhibit unpredicted behavior, and worse can lead to chaos.

---

[3]Reprinted from *Computer Integrated Manufacturing Systems* 6 (2). The reference for the original material is as follows: Veeramani, D., Bhargava, B. and Barash, M. M., Information system architecture for heterarchical control of large FMSs. *Computer Integrated Manufacturing Systems,* 6 (2), 76-92, Copyright 1993, with permission from Elsevier.

3.5. New Manufacturing Concepts and Architectures

Considering the deficiencies of the control architectures presented above and the ineffective implementations of CIM, in the last decade, the research community introduced several new concepts for the design of manufacturing systems, such as bionic manufacturing or biological manufacturing systems (BMS), fractal factory or more recently fractal manufacturing systems (FrMS), and holonic manufacturing systems. Each of these concepts attempts to model a manufacturing system based on some analogies with existing natural, theoretical or social organization systems (Tharumarajah *et al.,* 1996).

Multi-agent systems (MAS) have been regularly used in a large number of different applications such as distributed computing, Internet, and telecommunications. More recently, MAS theory is increasingly employed in manufacturing applications. Manufacturing architectures developed based on MAS are now challenging the position held by the hierarchical architectures as the main control structure used in industry. Another approach to the design of manufacturing systems is the introduction of reconfigurable manufacturing systems (RMS) concept. By using modular hardware and software, RMS aim at providing exactly the capacity and functionality needed in response to any changing condition within the system and in the environment (Mehrabi *et al.,* 2000). Each of these new concepts is briefly described below.

3.5.1. Bionic Manufacturing Systems

BMS is a concept inspired by the similarities between biological organisms and manufacturing structures. As Tharumarajah *et al.* (1996) stated "natural life exhibit autonomous and spontaneous behavior, and social harmony within hierarchical ordered relationships." As seen in one of the previous sections, autonomy is a desired characteristic for entities in a manufacturing system, and improves its response to internal or external stimuli. If the spontaneous behavior of the biological structures can be captured and modeled in manufacturing systems, an even better response to unexpected events will result. Moreover, social harmony of biological structures if translated to manufacturing entities will lead to the necessary coordination and cooperation mechanisms between entities in a manufacturing structure. The unit of organization in BMS is an autonomous entity called modelon and includes other sub-

modelons to form a hierarchical structure within the autonomous entity, operators responsible with coordination among the sub-modelons within the autonomous entity, and a common environment used to exchange information. Important to notice is the presence of the autonomous and cooperative entities within a distributed architecture in the BMS concept.

Mill and Sherlock (2000) studied the biological analogies in manufacturing and argued that the main reason for pursuing research in BMS is to translate to manufacturing systems, two of the important characteristics of the biological systems, the self-organization and the adaptive characteristics, the second one leading to evolutionary optimization characteristics. Just like biological systems, the manufacturing system is considered as an organism that should respond to external stimuli and create products. Most of the research in BMS as reported in the literature (Tharumarajah *et al.,* 1996, Leitao and Restivo, 2000) is the result of the work of the Japanese Professor Norio Okino of Kyoto University.

3.5.2. Fractal Factory and Fractal Manufacturing Systems

This concept is the result of the research work of the German professor Warnecke who applied the characteristics of fractal geometry to model the behavior of constituent entities of a manufacturing system. The two important characteristics of fractals, self-similarity and self-organization are used to model the base unit of organization in the fractal factory (Tharumarajah *et al.,* 1996). The fractal architecture is composed of self-similar fractal objects which can be divided in other fractal objects having the same organizational structure and objectives with the parent fractal (Leitao and Restivo, 2000). Self-similarity helps fractals to pursue common goals, while self-organization lets the fractal objects to arrange their internal structure according to their individual goals. Autonomy, expressed by self-organization in this case, and cooperation, expressed by the self-similarity characteristic are once again key attributes for this approach, as well.

More recently, Ryu *et al.* (2001), and Ryu *et al.* (2003) used Warnecke's fractal factory concept to develop a new modeling framework for manufacturing systems in which entities in the architecture are referred as fractals and modeled using MAS technology. Fractals are arranged in hierarchy in a similar way as the structure of holonic systems (described in detail in one of next chapters), and are composed of five different types of functional modules,

specifically: observer, analyzer, solver, organizer, and reporter, which are modeled using MAS. Using predefined rules and an internal knowledge database, the fractals, directly connected to resources on the shop floor, are performing all the tasks coming to the shop floor, without human intervention. Because of the autonomy of its constituents, and the fractal characteristics of self-organization, the FrMS concept is best suited for dynamic and decentralized manufacturing environments.

3.5.3. Holonic Manufacturing Systems

The holonic concept originated from the work of Hungarian author and philosopher Arthur Koestler who tried to capture the behavior of complex systems by considering its constituent entities as being both wholes and parts at the same time (Koestler, 1968). By the time Koestler published his work, artificial intelligence (AI) was still a beginning science and the decentralized systems based on agents were not yet developed. However, the idea on which holons are developed and interact with each other is based on local autonomy and cooperation within the environment, basic characteristics for the MAS described in the distributed artificial intelligence (DAI) field.

By incorporating hierarchy in a distributed architecture, as Bongaerts *et al.* (2000) suggested, it can be combined the advantages of both hierarchical and heterarchical systems, namely "robustness against disturbances and unforeseen changes with performance optimization and predictability." To obtain such a structure, they suggested using a heterarchical structure extended with central agents to coordinate the behavior of local agents. This is, in fact, the manufacturing application of the holonic concept of Koestler, which will be discussed in detail in the next chapters. In a manufacturing environment such a system will work as a hierarchy when the initial conditions for which the control programs were developed stand and no disturbances appear, and as a heterarchical structure in the presence of any unexpected events that move the system from its initial conditions.

3.5.4. Agent-Based Manufacturing Systems

Intelligent software agents, first introduced in DAI field, were developed "due to the

difficulties that have arisen when attempting to solve problems without regard to a real external environment or to the entities involved in that problem-solving process" (d'Inverno and Luck, 2001). There is an obvious similarity for the need to introduce agents in DAI systems and the need that led to the use of intelligent agents in manufacturing area. In both cases the systems developed before did not consider the impact of the environment changes to the overall system behavior, and therefore the results obtained were unrealistic. The distributed architecture of MAS and the agents' characteristics of autonomy and cooperation make MAS a suitable tool for the design, development and implementation of the bionic, fractal and holonic manufacturing concepts.

Agents hold the most important place in DAI field. In fact, all the theory of DAI is constructed around agents. Russell and Norvig (2003) define an agent as "something that perceives and acts in an environment," characterized by a performance measure which "evaluates the behavior of the agent in the environment." An intelligent agent "acts so as to maximize the expected value of the performance measure," so, given the information received from the environment the intelligent agent selects "an action that is expected to maximize its performance measure, given the evidence provided by the percept sequence and whatever built-in knowledge the agent has."

A MAS is a distributed system composed of two or more agents, each having their independent problem-solving mechanism, without a global system objective, but having individual goals and capabilities. To accomplish their goals, agents need to have a common means, usually in the form of a language in which they can communicate. These languages, called agent communication languages (ACL) are also referred in the literature as communication protocols.

For the MAS to achieve an overall system objective, there is a need for coordinating actions among the agents comprising the MAS such that the duplication of effort, and unwittingly hindering of other agents in achieving their goals is avoided (d'Inverno and Luck, 2001). Coordination includes communication to establish a common platform to exchange messages, cooperation, when two or more agents are working on to perform the same assignment, and negotiation, when the objectives of two or more agents are in contradiction and a negotiating platform should be constructed to avoid deadlocks and finish the required assignments.

3.5.5. Reconfigurable Manufacturing Systems

More recently in the second half of the 90's, another concept, reconfigurable manufacturing systems, was introduced as potential solution for the challenges coming with the rapidly changing manufacturing environment. The VMC2020 final report (National Research Council, 1998) identified reconfigurable enterprises as one of the six grand challenges for manufacturing in the next years, and RMS as a priority technology and research opportunity to meet this grand challenge. A view of the RMS capabilities is presentd in Figure 3.6.

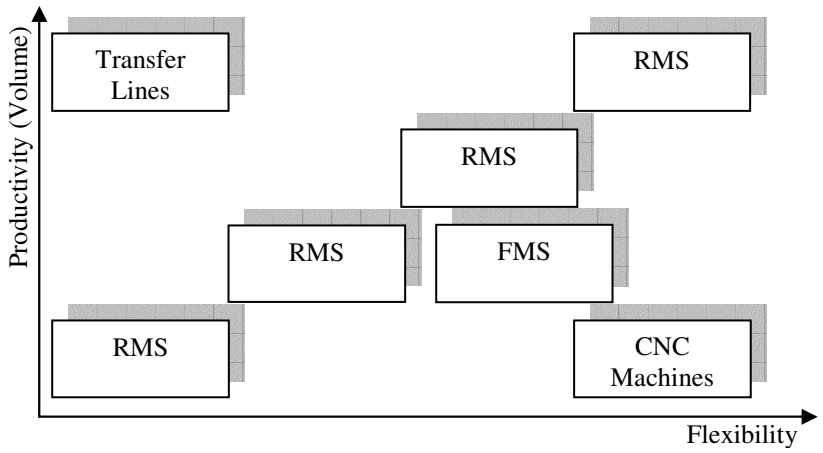

Figure 3.6. A view of the RMS capabilities. Adapted from Mehrabi *et al.* (2000) with the kind permission of Springer Science and Business Media[4].

RMS concept aims at allowing flexibility not only in producing a variety of parts, but also in changing the system itself (Mehrabi *et al.,* 2000). A RMS is viewed as a modular system in regard to both hardware and software, which can be rearranged quickly to adapt to each new manufacturing situation. Mehrabi *et al.* (1998) gave the following definition for RMS: "A RMS is designed for rapid adjustment of production capacity and functionality, in response to new circumstances, by rearrangement or change of its components."

---

[4]Originally published by Kluwer Academic Publishers in *Journal of Intelligent Manufacturing* 11 (4), figure 2 at page 406. The reference for the original material is as follows: Mehrabi, M. G., Ulsoy, A. G. and Koren, Y., Reconfigurable manufacturing systems: key to future manufacturing. *Journal of Intelligent Manufacturing,* 11 (4), 403-419, 2000.

These manufacturing systems are to be installed with the "exact production capacity and functionality needed, and may be upgraded (in terms of both capacity and functionality) in the future, when needed" (Mehrabi *et al.,* 2000). RMS concept aims at allowing flexibility not only in producing a variety of parts, but also in changing the system itself (Mehrabi *et al.,* 2000). A RMS is viewed as a modular system in regard to both hardware and software, which can be rearranged quickly to adapt to each new manufacturing situation. Mehrabi *et al.* (1998) gave the following definition for RMS: "A RMS is designed for rapid adjustment of production capacity and functionality, in response to new circumstances, by rearrangement or change of its components." These manufacturing systems are to be installed with the "exact production capacity and functionality needed, and may be upgraded (in terms of both capacity and functionality) in the future, when needed" (Mehrabi *et al.,* 2000).

By having the possibility to add and remove capacity and functionality, the RMS concept aims at covering the entire space between transfer lines and FMS in the productivity-flexibility diagram, by combining the advantage of high production volumes of transfer lines with the high flexibility of FMS, as presented in Figure 3.6. RMS is an intensive research topic at the National Science Foundation-Engineering Research Center (NSF-ERC) at the University of Michigan in Ann Arbor and under the High Productive And Reconfigurable Manufacturing Systems (HIPARMS) program of the Intelligent Manufacturing System organization. The RMS approach to the design of manufacturing systems is trying to develop new systems having the possibility to change their structure and operation for rapid and economical adjustment to any new production requirements.

4. Scheduling of Manufacturing and Material Handling Tasks

Literature to date in the scheduling arena gives many different definitions on scheduling, but they all have in common the process of allocation of a limited number of resources to a usually greater number of tasks. Lee *et al.* (1997) considered the scheduling process as the allocation of "limited resources to tasks to optimize certain objective functions." Pinedo (2002) stated that "scheduling deals with the allocation of scarce resources to tasks over time," and can be considered a "decision-making process with the goal of optimizing one or more objectives." The complexity associated with scheduling problems has been intensively studied in the last several decades, so now it is widely accepted that most practical real-world scheduling problems cannot be solved optimally in reasonable time due to the combinatorial explosiveness associated with most of them.

4.1. Scheduling Algorithms

Algorithms are the basis of computer science, and can be defined as a step-by-step problem-solving method suitable for implementation as a computer program (Sedgewick, 1998). Every computer program that gives an answer to a problem by calculating the same expression through iteration contains an algorithm. From a logician's point of view, (Berlinski, 2000) an algorithm is "a finite procedure […] governed by precise instructions, moving in discrete steps […], whose execution requires no insight, cleverness, intuition, intelligence, or perspicuity, and that sooner or later comes to an end."

Selection of algorithms for manufacturing and MH scheduling needs a special attention because of the combinatorial explosiveness associated with them. In manufacturing environments, combinatorial explosiveness appears due to an "exponential growth in the number of possible schedules along each dimension such as operations, machines, tools, personnel, or orders" (Bauer *et al.*, 1994). The time to solve optimal algorithms for parallel machines, job-shop and open-shop scheduling problems increases exponentially with the increase in the number of jobs and machines on the shop floor. Moreover, because running time is the most important measure of efficiency for an algorithm, researchers and practitioners are considering using approaches that do not produce optimum solutions, but can be solved in acceptable time, such as

approximation, local improvement or heuristic algorithms, and dispatching rules.

4.2. Manufacturing Scheduling

Scheduling is one of the processes that need to be performed in manufacturing systems, and in its general case includes both scheduling processing operations and MH resources. Using the general definitions of scheduling, manufacturing scheduling can be described as the optimization process that allocates limited manufacturing resources over time among parallel and sequential manufacturing activities (Shen, 2002). In real-world manufacturing environments, finding the right sequences and the associated schedules when resource, precedence, and timing constraints are imposed is a difficult task. For most practical manufacturing problems, the classical scheduling approach leads to an exponential growth in the number of possible schedules. Moreover, a decision time period of hours or even minutes is too long. Good solutions are often needed in real-time. The problem becomes even more complicated if changes, such as new orders or resource breakdowns, occur within the manufacturing system. Therefore, attempting to find optimal solutions using the classical scheduling theory is in many cases unrealistic. One approach to overcome the limitations of classical scheduling is the use of distributed schemes such as agent or holonic-based control architectures.

4.2.1. Predictive Scheduling

There are two known approaches to manufacturing scheduling (Sun and Xue, 2001). The first one, called predictive scheduling, constructs the system schedule based on given requirements and constraints, prior to the production process. This type of scheduling is also known in the literature as classical scheduling. Centralized and hierarchical systems use this approach for assigning jobs to the processing machines in a manufacturing system. If the algorithms are designed to search for the optimal solution and the problem is not combinatorial complex, the resulting schedules are optimal. Otherwise, a close-to-optimal schedule can be built using one of the methods presented above. However, real-world manufacturing systems are not deterministic, so the predefined schedules can rarely be used without modifications. They usually need to be updated or even ignored when the impact of the changes occurred is significant. As

Pendharkar (1999) stated, “scheduling prior to production is often a waste of time as dynamics of production can quickly render a schedule invalid.” Therefore, attempting to find optimal solutions using classical scheduling is in many cases unrealistic.

4.2.2. Reactive Scheduling

The second approach, called reactive scheduling is based on modifying the schedule during the operations to adapt it to changes in the manufacturing environment. Reactive scheduling is used in decentralized systems, such as holonic or agent-based systems to obtain real-time feasible solutions for assigning operations to processing machines and scheduling the MH resources. Pinedo (2002) stated that “distributed reactive scheduling applied to a completely deterministic problem, cannot perform as well as an optimization technique designed to search for a global optimum. However, distributed scheduling may play an important role in practice in more complicated machine environments with a long horizon that are subject to various forms of randomness.” Cardon *et al.* (2000) stated that, to find feasible solutions for real-time scheduling problems a good approach is the use of distributes systems, such as agent-based systems.

Using the distributed scheduling approach, a decentralized schedule emerges during the actual manufacturing operations from the interaction of the entities forming the architecture. The resulting schedule works as a reactive schedule and changes after any unexpected event occurrence, so it is more adapted to the characteristics of dynamic manufacturing environments. Contrary to classical scheduling systems where schedules are computed based on the information available at the start of the operations, in distributed systems schedules emerge during operations as the new orders entering the system are accommodated considering the availability and status of resources in the system. These characteristics lead to a significant reduction, for the decentralized scheduling systems, in the computations and therefore the time needed to obtain a feasible solution. However, as Karageorgos *et al.* (2003) stated, this improvement comes at the expense of not guaranteeing a globally optimum solution.

4.3. Influence of Material Handling in Manufacturing Scheduling

In real-world manufacturing systems, after having developed the schedule of processing

operations, to correctly find the total processing time of the orders released to the shop floor, it is necessary to insert the MH operations between processing operations and recalculate the schedule makespan based on this new information. In MH operations, scheduling gives the sequence in which jobs need to be moved and provides time-routing and dispatching of transporters for job pick-up and delivery (Lee *et al.,* 1997). But, in most cases, the classical manufacturing scheduling theory disregards MH operations by assuming that they are negligible (Smith *et al.*, 1999). This could be a good assumption in the case when the processing time of every operation is much longer than any MH operation, and the MH resources are always available when needed.

However, in real-world manufacturing there are many situations in which the MH resources are not always available when needed because they can be executing other tasks or in the process of traveling among workstations. This results in delaying the loading/unloading (L/U) operations of the machines and increasing individual jobs completion times and schedule makespan. Moreover, as Smith *et al.* (1999) proved for a simple two-machine job-shop (J2 | | C_{max}) problem, the makespan depends also on the order in which the machines are being served as presented in Figure 4.1.

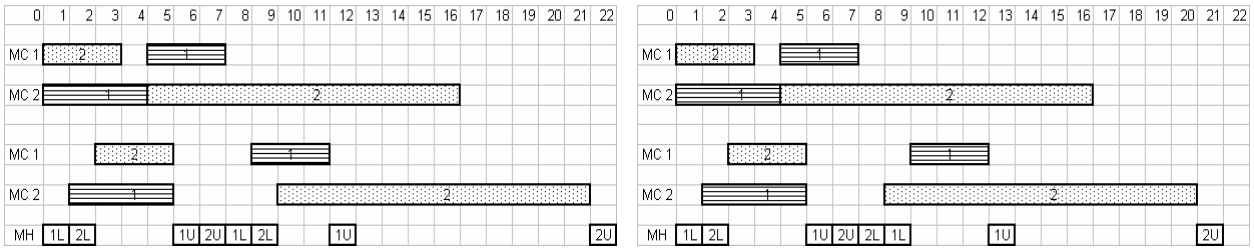

Figure 4.1. Influence of MH on schedule makespan. Adapted from Smith *et al.* (1999) with the kind permission of Taylor & Francis[5].

Depending on the processing times of the operations, changing the order of loading the

[5]Reprinted from *International Journal of Production Research* 37 (2) with permission from Taylor & Francis (http://www.tandf.co.uk). The reference for the original material is as follows: Smith, J. S., Peters, B. A. and Sirinvasan A., Job shop scheduling considering material handling. *International Journal of Production Research,* Vol. 37, No. 2, 1541-1560, 1999.

two machines is giving different results for the schedule makespan. So, even for a deterministic case that does not consider possible changes in the system, the pre-determined schedule becomes invalid as soon as one of the above conditions occurs. Two solutions to account for MH operations are to either solve the problem by disregarding MH operations and then add the MH operations times to the final schedule, or add from the start the L/U times to the processing times of each operation, and then solve the scheduling problem. The results given by these solutions are not always correct since both approaches do not consider the case in which a machine must wait to be served because there is no MH resource available, or the order of machines to be served is not leading to the minimum makespan.

As a result, delays in processing jobs will occur even the predefined schedule had all the jobs supposedly processed by their due dates. Another solution is to consider MH operations as separate operations and MH equipment as resources. By using distributed scheduling algorithms for the MH operations, this approach attempts to find the best order for serving the machines, while considering the waiting times described above, and, as it will be presented in the next chapters, gives the best image of the impact of the MH operations in manufacturing cell environments.

5. Holonic Manufacturing Systems Concept

This chapter presents the origins of the holonic concept, its first applications in manufacturing, and gives the definition of the HMS and its constituent entities as developed by the Holonic Manufacturing Systems Consortium (HMSC) within the Intelligent Manufacturing Systems (IMS) Program.

5.1. Origins of the Concept

In his study (Koestler, 1968) about the different types of organizational structures, the Hungarian author and philosopher, Arthur Koestler observed that although most of the real-world systems are built on a hierarchical structure, with some entities being part of other larger entities, there is not a clear separation between parts and wholes when looking at biological or social systems. Thus, in his vision, the well established definition of a part as "something incomplete which by itself would have no legitimate existence," and of a whole, as "something complete in itself which needs no further explanation," cannot capture the characteristics of the real-world organizational structures, and in fact, as Koestler notes, "wholes and parts, in this absolute sense just do not exist."

To describe a basic unit of organization in biological and social systems, Koestler proposed/invented the word "holon," which comes from the combination of the Greek word for whole, "holos," and the suffix "on" meaning a part or a particle. As, within a social organization, holons behave "partly as wholes and wholly as parts" according to the way one looks at them, Koestler proposed also the concept of open-ended hierarchy (OEH) as the architecture formed out of holons, called holarchy, which is not bounded in both downward and upward directions. An OEH is built on a structure similar to the functional structure of a military organization based on platoons, companies, battalions, divisions, and so on. However, in contrast to the military structure an OEH is open-ended in both downward and upward directions as stated above.

Besides social organizations, Koestler viewed also biological systems as appropriate to be described as holonic structures. He noted that a biological organism "constitutes an […] integrated hierarchy of molecules, cells, organs and organ systems." Viewed from outside a biological organism is "something complete and unique, a whole," but when one looks from

opposite direction it discovers that "it is a part, an elementary unit in one or several social hierarchies." This relation between living organisms and the social organizations in which they live suggests that bionic and holonic manufacturing concepts have much in common. And, in fact, this is the case, both these concepts being developed using DAI tools. Two important properties of holons, make holonic control concept a suitable modeling tool for use in the design of both manufacturing and MH systems. First, autonomy allows holons the right to make decisions without consulting any supervising entity, and secondly, cooperation permits holons to communicate with other peer holons to develop mutually acceptable plans and execute them.

5.2. Early Applications of the Holonic Concept in Manufacturing

Japanese researchers were the first to apply Koestler's holonic concept to manufacturing area during the 1980's, in the design and implementation of a so-called holonic manipulator controller. Hirose *et al.* (1986) presented the design of the holonic manipulator controller, and then in another paper (Hirose *et al.,* 1988) the software environment of the holonic manipulator is described. The prototype implementation for the manipulator was presented in 1990 (Hirose *et al.,* 1990). Reported advantages of applying the holonic concept in designing the manipulator were a more robust design due to a decrease in the wiring complexity and an increase in reliability of the manipulator. The control software for the manipulator requires coordination between the built-in controllers by using dedicated task managers and message exchanging, typical for holonic control software as presented in the following chapters.

5.3. Holonic Manufacturing Systems Concept Defined

The idea of using the holonic concept in the design of manufacturing systems emerged in the early 1990's in the IMS program as a solution to cope with the increased rate of changes that was affecting the entire business world including the manufacturing sector. A consortium with researchers from Australia, Canada, Europe, Japan and the U.S. was established within the IMS program with the goal of developing the tools and implementing the holonic concept in real-world manufacturing industry, and thus obtain the potential benefits offered by holonic organizations such as "stability in the face of disturbances, adaptability in the face of change and

efficient use of available resources" (Van Brussel *et al.,* 1998). To guide the research in the area, the HMS consortium participants established a series of working definitions for the constituent entities of the holonic systems (Christensen, 1994):

- Holon: an autonomous and cooperative building block of a manufacturing system for transforming, transporting, storing and/or validating information and physical objects. The holon consists of an information part and often a physical processing part. A holon can be part of another holon.
- Autonomy: the capability of an entity to create and control the execution of its plans and/or strategies.
- Cooperation: a process whereby a set of entities develops mutually acceptable plans and executes these plans.
- Holarchy: a system of holons which can cooperate to achieve a goal or objective.
- Holonic manufacturing system: a holarchy which integrates the entire range of manufacturing activities from order booking through design, production, and marketing to realize the agile manufacturing enterprise.

Koestler's concepts of holonic systems and OEH formed by autonomous holons presented in the first section of this chapter are adapted in these definitions to manufacturing environment. The holarchy defined by the HMSC is built on the OEH concept and is adapted to include cooperative characteristics specific to decentralized manufacturing environments. This relationship between OEH and the holarchy defined by HMSC is captured by Tharumarajah *et al.* (1996) and depicted in Figure 5.1.

Contrary to the traditional hierarchical systems, a holonic system or holarchy is a decentralized system in which tasks are allocated and executed after exchanging information and coordination among entities in the architecture modeled as autonomous and cooperative holons. The arrows in Figure 5.1 present a series of possible relationships between holons in the holarchy established when performing incoming tasks. However, when looking at the necessary relationships among holons to execute a task, this image, though correct, is not enough to describe the complex relationships in a holonic system. There could be also direct relationships between holons on different levels of the OEH which this figure cannot capture.

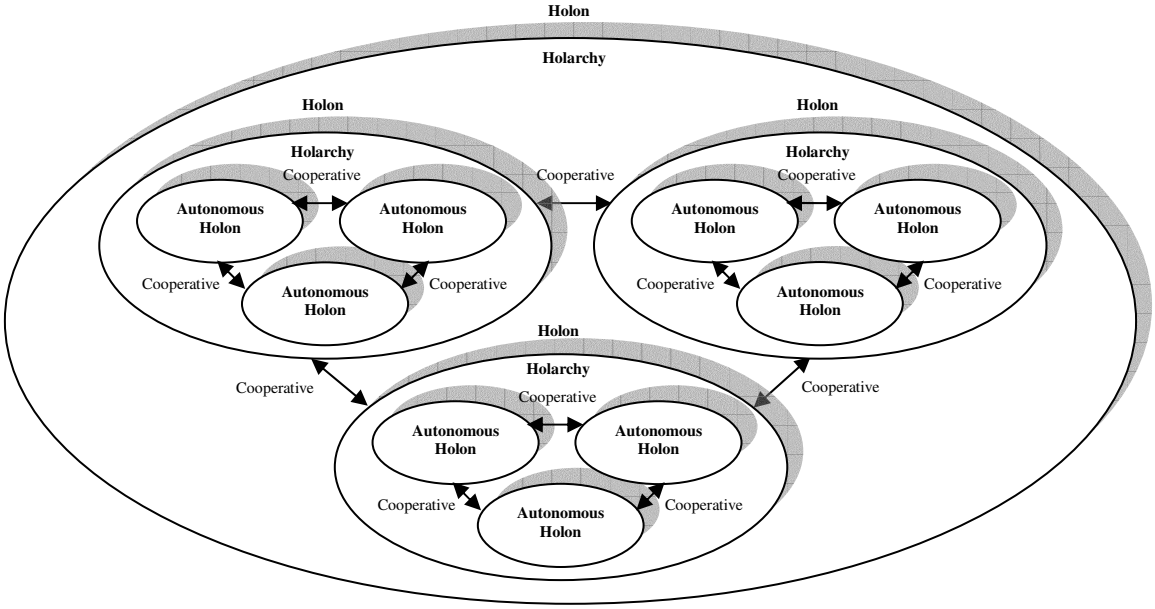

Figure 5.1. A holarchy formed of autonomous and cooperative entities. Adapted from Tharumarajah *et al.* (1996) with the kind permission of Taylor & Francis[6].

[6]Reprinted from *International Journal of Computer Integrated Manufacturing* 9 (3) with permission from Taylor & Francis (http://www.tandf.co.uk). The reference for the original material is as follows: Tharumarajah, A., Wells, A. J. and Nemes, L., Comparison of the bionic, fractal and holonic manufacturing system concepts. *International Journal of Computer Integrated Manufacturing,* Vol. 9, No. 3, 217-226, 1999.

6. Considerations on the Development of Holonic Systems

Several considerations for the development of holonic manufacturing systems and specific holonic system requirements are discussed in this section. The characteristics of individual agents and their relationships within distributed agent architectures make MAS theory appropriate for the implementation of HMS. Thus, HMS research is strongly related to the MAS research within the DAI community. As Ulieru *et al.* (2001) stated "multi-agent systems paradigm seems to be well suited to implementing a holonic abstraction of a problem which is fundamentally distributed in nature." However, there are two differences between the constituent entities in MAS and holonic systems. The first one is related to the aggregation of holons in the holonic architecture. Paolucci and Sacile (2005) observed that "holons in a holarchy are quite similar to agents in MAS, if one disregards the fact that a holon can contain other holons." The other difference comes from the modeling abilities of the constituent entities of the two types of systems. While agents are pure software entities, holons can include both hardware and software parts of the modeled system. Still, MAS are the only platform identified as the modeling tool for developing holonic systems. Table 6.1 gives a classification of the papers reviewed in the holonic systems area based on the main developmental aspect considered by their authors.

6.1. Holonic Systems Design

A holon is an autonomous entity considered a whole that can include sub-holons having inherited characteristics, and in the same time it could be part of a broader holon to whom it can pass on some of its characteristics. Based on Koestler's holonic and OEH considerations, the aggregation of holons and their relationships within a holonic structure can be modeled as in Figure 6.1 where the basic entity is referred as reference holon. This structure permits a clear distinction between all entities included in the architecture and allows the possibility to investigate a particular part of it without considering the entire structure.

Typically, a holon is formed by a physical processing unit and a software control unit, though, for holons not associated with manufacturing resources the holon comprises only the software control unit. To design the control architecture of holonic systems, two types of standards are proposed in the literature (Christensen, 2003). For the low-level control (LLC)

architecture which refers to automation functions, it is proposed the International Electrotechnical Commission (IEC) 61499 series of standards for the use of function blocks in distributed industrial automation and control systems. Under the IMS project a significant progress was made for designing the holonic LLC architecture using the IEC 61499. This choice was made due to the capabilities offered by the above mentioned function blocks in terms of the "flexibility required in scheduling events, and especially in the case of hard real-time control" (Marik and Pechoucek, 2001). High-level control (HLC) architecture refers to inter-holon interactions and integration of the automation functions into holonic architecture. The collection of architectural standards for software agent systems developed under the Foundation for Intelligent Physical Agents (FIPA) are proposed in several papers as the solution to be used for designing the HLC architecture of holonic systems (Marik *et al.,* 2003).

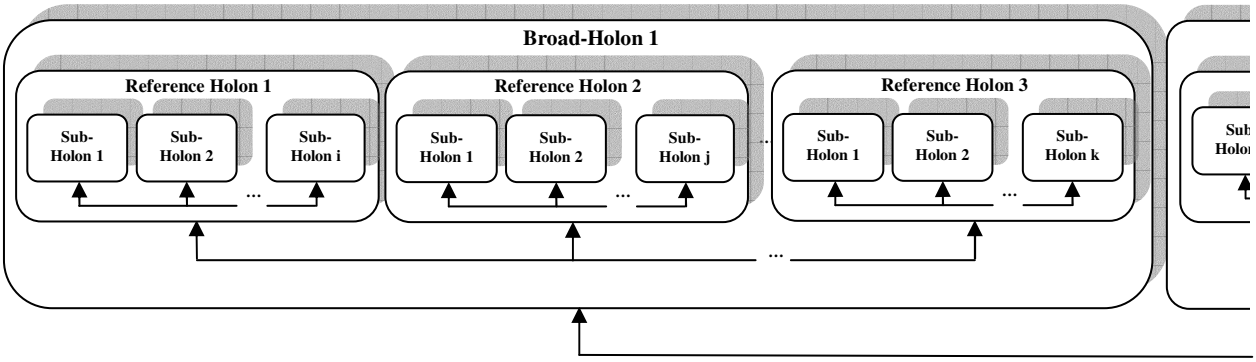

Fig. 6.1. The reference holon in a holonic architecture.

When a manufacturing resource unit is present, such as a processing machine or a MH resource, it needs to have a control unit associated with it to establish and follow the processing unit objectives based on its internal set of goals. Christensen (1994) developed a broader model of a holon which includes also a human unit functioning as a resource in the same way as the physical processing unit, but in the same time, it exchanges information with the environment and can act on the physical processing unit just like the software control unit. In this model, the raw materials and information entering the holon are altered by the processing system, and then transmitted to other processing units, part of other holons.

A more detailed view of a holon developed also under the HMS Consortium program was

presented by Fletcher *et al.* (2000). The software control unit, called intelligent control system (ICS) is further detailed to include all its building blocks, such as controllers and communication interfaces with other components of the holon. Based on the IEC 61499 function block standard the components of the ICS, their functions and interactions are presented in detail. The layered structure of the IEC 61499 function block provides the holons in the architecture the software portability, configurability and interoperability needed to act as autonomous and cooperative entities. The same approach of using function blocks to model the LLC architecture of holonic system appears in several other papers such as Wang *et al.* (1998b), Blasubramanian *et al.* (2000), Zhang *et al.* (2001), Christensen (2003), Deen (2003), and Neligwa and Fletcher (2003).

Table 6.1. Classification of the papers reviewed considering their holonic developmental characteristics and their broad area of application.

	Manufacturing Enterprises	*Shop Floor Control Systems*
Holonic System Design	Christensen (1994); Mathews (1995); Gou *et al.* (1998); Toh *et al.* (1998); Chirn and McFarlane (2000a); Chirn and McFarlane (2000b); Rahimifard *et al.* (1999); Toh *et al.* (1999); Leitao and Restivo (2000); Ulieru *et al.* (2000); Cheng *et al.* (2001); Fletcher and Deen (2001); Ulieru (2001a); Ulieru (2001b); Ulieru and Norrie (2001); Ulieru *et al.* (2001); Huang *et al.* (2002); Christensen (2003); Monostori (2003).	Gou *et al.* (1994); Ramos (1996); Bongaerts *et al.* (1997); Bruckner *et al.* (1998); Gayed *et al.* (1998); Gou *et al.* (1998); Rannanjarvi and Heikkila (1998); Sousa and Ramos (1998); Valckenaers *et al.* (1998); Van Brussel *et al.* (1998); Chirn and McFarlane (2000a); Chirn and McFarlane (2000b); Monostori and Kadar (1999); Silva and Ramos (1999); Sousa and Ramos (1999a); Sousa and Ramos (1999b); Zhang and Norrie (1999); Balasubramanian *et al.* (2000); Bongaerets *et al.* (2000); Cheung *et al.* (2000); Fletcher *et al.* (2000); Hammerle *et al.* (2000); Liu *et al.* (2000); Lun and Chen (2000); Shu *et al.* (2000); Arai *et al.* (2001); Balasubramanian *et al.* (2001); Brennan and Norrie (2001); Brennan *et al.* (2001); Bussmann and Sieverding (2001); Fletcher *et al.* (2001); Giebels *et al.* (2001); Heikkila *et al.* (2001); Sugimura *et al.* (2001); Vrba and Hrdonka (2001); Wang (2001); Heragu *et al.* (2002); Hsieh (2002); Wullink *et al.* (2002); Deen (2003); Heikkila *et al.* (2003); Johnson (2003); Marik *et al.* (2003); McFarlane and Bussmann (2003); Monostori (2003); Neligwa and Fletcher (2003); Ritter *et al.* (2003); Tamura *et al.* (2003); Bussman *et al.* (2004); Hsieh (2004).
Fault-Tolerance	Fletcher and Deen (2001); Ulieru and Norrie (2001); Heragu *et al.* (2002); Christensen (2003).	Jarvis and Jarvis (2003); Johnson (2003); Neligwa and Fletcher (2003).
Real-Time Control	Christensen (2003).	Balasubramanian *et al.* (2000); Balasubramanian *et al.* (2001); Brennan *et al.* (2001); Zhang *et al.* (2000).
Production Planning and Scheduling	Gou *et al.* (1998); Toh *et al.* (1998); Chirn and McFarlane (2000b); Toh *et al.* (1999); Fletcher and Deen (2001); Ulieru and Norrie (2001); Ulieru *et al.* (2001).	Gou *et al.* (1994); Overmars and Toncich (1996); Ramos (1996); Bruckner *et al.* (1998); Gou *et al.* (1998); Rannanjarvi and Heikkila (1998); Sousa and Ramos (1998); Chirn and McFarlane (2000b); Silva and Ramos (1999); Sousa and Ramos (1999a); Sousa and Ramos (1999b); Zhang and Norrie (1999); Bongaerets *et al.* (2000); Cheung *et al.* (2000); Lun and Chen (2000); McFarlane and Bussmann (2000); Shu *et al.* (2000); Arai *et al.* (2001); Brennan and Norrie (2001); Bussmann and Sieverding (2001); Heikkila *et al.* (2001); Schoop *et al.* (2001); Sugimura *et al.* (2001); Hsieh (2002); Deen (2003); McFarlane and Bussmann (2003); Neligwa and Fletcher (2003); Tamura *et al.* (2003); Bussman *et al.* (2004).

6.1.1. Internal Architecture of Holons

As stated above all holons part of a distributed architecture have a control unit which is responsible for guiding holons towards accomplishing individual or group objectives. Based on the information and algorithms included in the control unit, a holon must be able to assess its status, react to changes in the environment, and decide the most appropriate actions to be performed which enhance its internal performance measure and move the holon closer to its objective. An internal database is used to store knowledge related to the holonic architecture, tasks to be performed, and manufacturing environment, needed to correctly evaluate and execute potential tasks. A schematic representation of the internal architecture of a holon in presented in Figure 6.2.

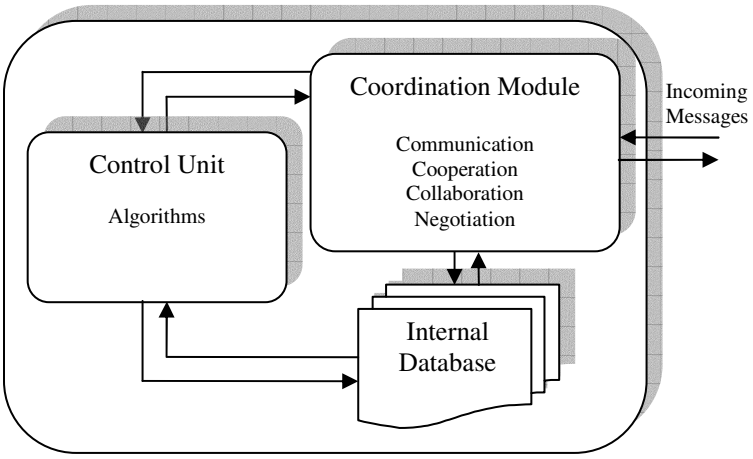

Fig. 6.2. Internal architecture of a holon.

6.1.2. Autonomy and Cooperation

The HMS Consortium definition considers autonomy and cooperation the two most important characteristics of holons. Glanzer *et al.* (2001) stated that "when consider the intelligent and cooperative part of a holon we find properties similar to software agents," thus the basic unit comprising MAS is a good choice to be used in the design of holonic systems. Moreover, Ulieru *et al.* (2001) noted that, "a system decomposition and analysis based on

holonic principles naturally suggests a distributed software implementation, with autonomously executing cooperative entities as building blocks."

Autonomy allows holons to decide the actions needed to be taken such that their individual objectives are accomplished without consulting any supervisory entity. Cooperation is the characteristic that permits holons to agree on common plans and mutually execute them. It also aids holons to seek help in the case of a malfunction appeared after the start of the execution of the common plans.

6.1.3. Relationships between Entities in Holonic Systems

The holons forming the holonic architecture need protocols and methodologies for exchanging information and coordinating their actions to accomplish individual or system wide objectives. Several terms are defined in MAS literature related to the potential relationships among entities in agent-based architectures, from which the most used are: coordination, communication, cooperation, collaboration, and negotiation. The definitions and the exact meanings of these five characteristics may differ from one work to another. Moreover, there is a clearly overlapping of these definitions as more than one paper is considered. As some of these relationships may need the accomplishment of others in order to take place, there could be also an overlapping of these relationships in the operation of MAS. A Venn diagram presenting these relationships and their overlapping characteristics is depicted in Figure 6.3. The definitions and characteristics of these relationships presented below are formulated after a comprehensive review of the available literature.

6.1.3.1. Coordination

Even a global optimal objective may not be achieved in a decentralized structure there might be several constraints which must be satisfied in order for the system to deliver a feasible solution. High-quality coordination moves the system towards satisfying these constraints. Thus, coordination primarily involves actions of the individual holons. If the overall objective function is time-dependent, the actions needed to be performed to achieve coordination become timed-actions. As d'Inverno and Luck (2001) stated, "coordination in MAS is necessary to avoid

duplication of effort and obstruction in the actions of achieving common goals" and includes communication for exchanging information, cooperation between two or more holons for the purpose of achieving a common goal, and negotiation in the case of conflicting interests, though coordination may result as a form of emergent behavior, without any of these characteristics. Methods to achieve coordination in MAS that can be used when developing holonic systems include the use of entities having a global view of the system, use of direct supervisors having a much larger view of the system, mutual adjustments among agents in the architecture, or mediated coordination (Shen *et al.,* 2001).

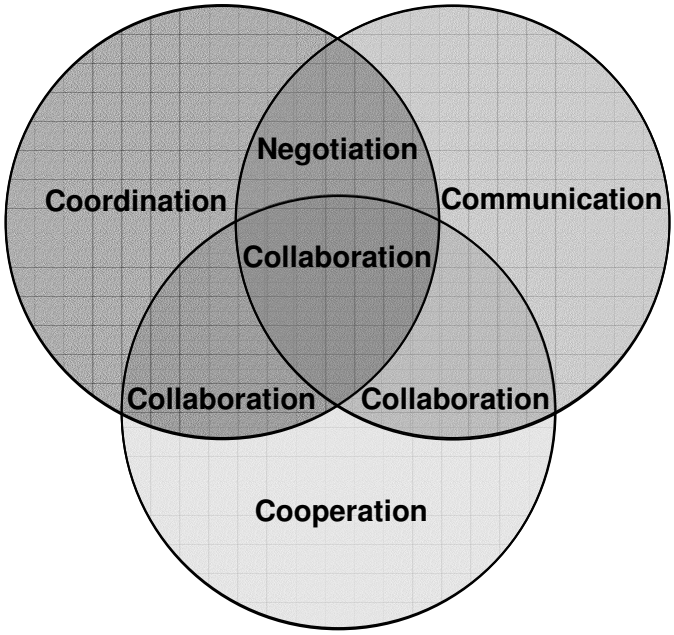

Fig. 6.3. Venn diagram showing the relationships among entities in a holonic system.

Using a global view entity presents the advantage of the global perspective offered, but in most cases it is impractical because of the complexity of the problems needed to be solved and the larger amount of communication that need to be passed through it. However, many holonic and agent-based systems developed have such an entity in their architecture but it is usually used for providing recommendations rather than as a decision-maker.

6.1.3.2. Communication

Due to the decentralized architecture of the holonic systems, entities that comprise the architecture need to exchange information in order to satisfy system constraints and achieve system level goals. Communication is the principal means for exchanging information in MAS, and it mainly involves data. Communication enables coordination and cooperation, though both these processes can occur without communication, and it is a must in the negotiation process. Because MAS is the tool to implement holonic systems, communication between entities in holonic systems is based on the agent communication methodologies and protocols used in MAS. Methods of communication in MAS include message or plan passing, information exchanges through a shared data repository, or high-level communication (Shen *et al.,* 2001). Message passing methodology is based on exchange of messages among the entities in the architecture and is the most used approach for communication in MAS. Several protocols were developed for message passing from which the most known is the contract net protocol (CNP) developed by Smith (1980) as part of his research in distributed problem solver area. CNP gives procedures for most communication needs within distributed agent architectures, such as task announcement and receiving, bidding mechanism, contract awarding and processing, and negotiations trade-offs (McFarlane and Bussmann, 2000).

6.1.3.3. Cooperation

Cooperation is the process, voluntarily or not, which results in common actions that move the entities involved in this process closer to a common goal. From this point of view, it can be considered that cooperation involves, primarily, common goal states. Doran *et al.* (1997) stated that cooperation in MAS means "to act with another or others for a common purpose or for common benefits." Cooperation in MAS does not require agents to voluntarily cooperate for achieving a common goal, though voluntarily cooperation is in many cases desired. When there is no intention of cooperation, and the agents are only following their individual goals, their actions can still lead to achieving a system level goal. Thus, cooperation in MAS can result also as a form of emergent behavior. Tools to achieve cooperation, as presented in the DAI literature, include agent coalition and clustering, communication, sharing tasks and resources, and conflict

resolution through negotiation (Shen *et al.,* 2001). When there is a voluntarily intent of exchanging data for the purpose of achieving a common goal, cooperation is sometimes referred as *collaboration* in the literature. But, basically, collaboration is a form of cooperation in which agents in MAS are voluntarily exchanging information for the purpose of achieving a common goal. There is no emergent behavior in collaboration since the actions of the agents are directed towards achieving common goals. Talukdar (1999) presented several collaboration rules among autonomous software agents, where collaboration is defined as "the exchange of data among information-processing agents, regardless of whether the exchange is productive or not," and a series of guidelines of how this rules can be extended in real-world real-time systems.

6.1.3.4. Negotiation

As the name implies, negotiation is the process of establishing an agreed set of actions that move a system or a part of it, from a conflicting situation towards a common goal. It principally involves intermediate or final system states. Depending on the architecture and the objective of the MAS, the negotiation process can be necessary between any types of entities in the architecture. In the holonic systems literature, negotiation is in some cases a term associated with task allocation. In this literature review, negotiation refers strictly to conflict resolution, so negotiations protocols come into play when there is something to bargain about. Conflicts between holons can result from several reasons such as different objectives, different way of performing specific actions, or incorrect assessment of manufacturing resources capabilities. The conflict resolution techniques used for holonic systems are those used in MAS and include compromise negotiation where a solution is finally agreed by changing values along some dimension until a common point is achieved, integrative negotiation where a solution is found by identifying the common most important objectives of all agents, and game-theory based negotiations (Shen *et al.,* 2001).

6.2. Task Allocation in Holonic Systems

The task allocation in holonic systems is the process of exchanging information among constituent entities and using it in internal algorithms and utility functions for the purpose of

distributing tasks to the available resources. The most used task allocation approach for holonic systems as reported in the literature is the CNP, which uses a bidding process to assign each new task entering the system to the most appropriate resource. Figure 6.4 presents the main steps of the CNP task allocation process.

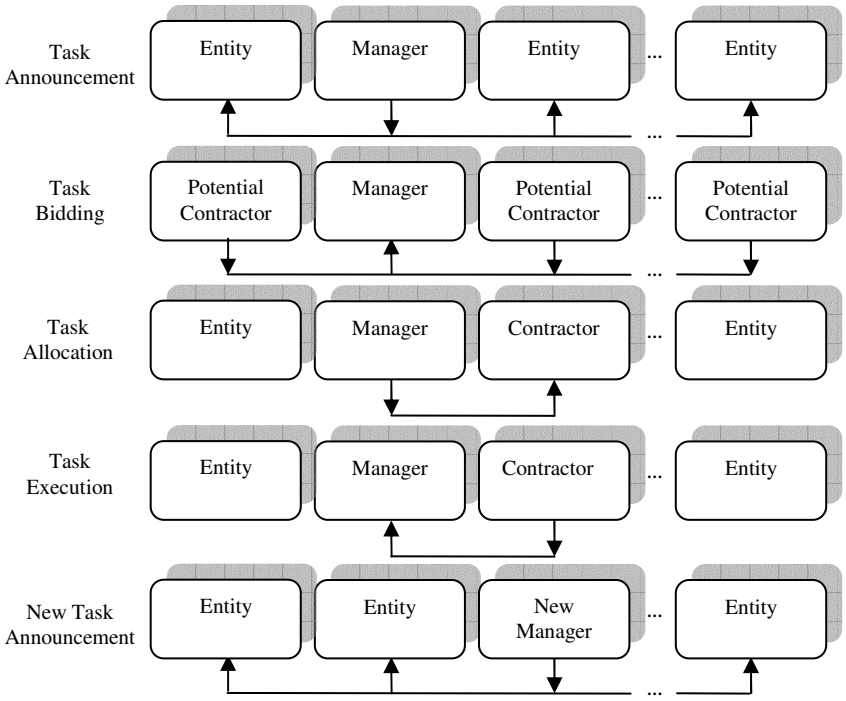

Fig. 6.4. The contract net protocol task allocation process.

As Smith (1980) stated, CNP is “a high-level protocol for communication among nodes in a distributed problem solver; it facilitates distributed control of cooperative task execution with efficient inter-node communication.” The output of this methodology is a contract between the agent that is acting towards its goal, called manager agent, and the other agent, called contractor agent, which based on its internal configuration and set of goals is performing the actions requested by the manager agent. CNP was developed initially for pure heterarchical architectures, where an agent can be both manager and executor of a particular order. Basically, when an agent needs to perform a task, it broadcasts the task offer, and waits a predefined period to receive bids from the participating agents in the architecture. The other agents could respond

to the broadcasted offer by submitting a bid, or take no action, depending on their internal objectives. The manager of the particular tasks, after receiving the individual bids, awards the task execution to the best subcontractor in concordance with its own evaluation algorithm. The winner of the bidding process becomes automatically the new manager for the next task. These main steps of CNP, as viewed by d'Inverno and Luck (2001), are presented in Figure 6.5.

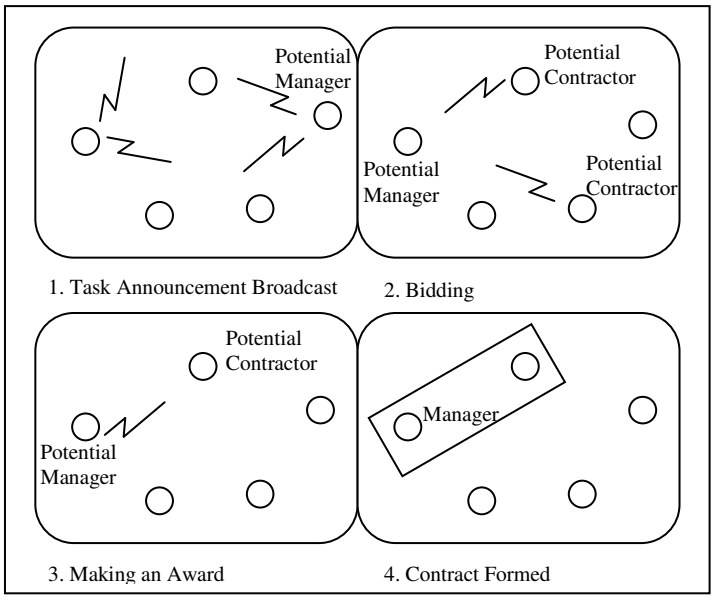

Figure 6.5. The main four steps of CNP. Adapted from d'Inverno and Luck (2001) with the kind permission of Springer-Verlag GmbH[7].

Probably the first to use CNP for task allocation in manufacturing was Parunak (1987) in his control architecture called YAMS (Yet Another Manufacturing System). The literature review identified several papers that employ CNP for distributing tasks in manufacturing applications in both holonic and agent-based systems (Gou *et al.,* 1994, Saad *et al.,* 1995, Ramos, 1996, Cantamessa, 1997, Shen *et al.,* 1997, Rannanjarvi and Heikkila, 1998, Sousa and Ramos, 1998, Ouelhadj *et al.,* 1999, Sousa and Ramos, 1999a, Sousa and Ramos, 1999b, Cardon

[7]Originally published by Springer-Verlag Berlin Heidelberg in *Understanding agent systems*, chapter 6, figure 6.1 at page 126. The reference for the original material is as follows: d'Inverno, M. and Luck, M., *Understanding agent systems*, Springer-Verlag, Berlin, 2001.

et al., 2000, Leitao and Restivo, 2000, Liu *et al.,* 2000, Unver and Anlagan, 2000, Usher and Wang, 2000, Vancza and Markus, 2000, Brennan and Norrie, 2001, Fletcher and Deen, 2001, Heikkila *et al.,* 2001, Mitidieri *et al.,* 2002, Hsieh, 2002, Neligwa and Fletcher, 2003, Tamura *et al.*, 2003, Hsieh, 2004).

6.3. Fault-Tolerance in Holonic Systems

When considering the design of manufacturing systems, Duffie (1990) noted that "one of the most difficult issues in the design of complex manufacturing systems is achieving fault-tolerance." This includes automatic detection of failure situations, diagnosis of the cause of failure, and determination and implementation of appropriate recovery actions. Writing algorithms for every possible scenario is practically impossible. Some failure scenarios can be anticipated, and thus algorithms that take into account the status of the system after that particular failure can be written. But for complex systems, it is not possible to consider all the potential failure scenarios. Hatvany (1985) noted that "no algorithm can be written which foresee every possible failure mode of a highly complex system, nor can remedial strategies be deterministically designed for every situation."

Fault-recovery in HMS is addressed by a number of researchers using different methods. Ulieru and Norrie (2000) applied fuzzy modeling techniques to study the capabilities of holonic systems to recover in the event of occurring faults. The implementation of the proposed approach is capable of accomplishing fault-recovery by re-distribution of tasks in the case of resource failures among existing resources. Fuzzy logic techniques are applied also to study the self-organizing capabilities of holonic systems developed based on intelligent agents as presented in Ulieru *et al.* (2000). In the fault-tolerant HMS developed by Fletcher and Deen (2001), multiple entities use a cooperation framework for rescheduling affected tasks and recover the system from the effects of the unexpected event. Fault-tolerance is achieved by using error generated information, a holon rescheduling mechanism, and a functional component failure recovery mechanism.

There are other papers in the holonic manufacturing area (Christensen, 2003, Jarvis and Jarvis, 2003, Johnson, 2003, Neligwa and Fletcher, 2003) that consider the potential malfunctions that could appear during the operation of a manufacturing system. Such, Jarvis and

Jarvis (2003) presented a model-based holonic diagnostic system for an automotive assembly line. Generally, in diagnostic systems the measures evaluated are the percent of the faults identified, also called fault coverage, and the execution time to identify and generate the diagnosis. By using heuristics to analyze the fault space, the fault coverage for the vehicle assembly station tested was reported at 95% with an execution time of less than one minute for each appeared fault.

6.4. Software Development

At the beginning of the software industry and until a decade ago, software developers used a programming technique, called structural programming, having a pure hierarchical structure to develop computer programs in which master programs are calling up slave routines or subroutines (Mathews, 1995). More recently object-oriented programming (OOP) concept, instead of using a master-slave relationship to return data to the main program, uses objects that contain both data and functions not accessible from outside the object. With the aid of a coordinating program, the objects perform the required tasks (Mathews, 1996). There are many similarities between the OOP theory and the holonic approach for modeling manufacturing systems. Similar to the autonomy concept in holonic systems, the encapsulation concept in OOP provides autonomy for the objects defined within the software program. In OOP, there is no object or sub-program that can access all the data and variables within the classes. This is similar to the holonic architecture where there is no entity with a global view of the whole system that can have decision power over other entities in the architecture. The inheritance concept in OOP assures that objects in a class share common features and have access to the same external objects. In the holonic architecture, common features are specific for holons which are part of the same broader holon and all the holons part of the same broader holon can communicate with the others.

Even that entities in holonic systems can be modeled as objects, they are best portrayed using intelligent agents. Agents are a special type of software objects which have their own internal algorithms, use a common language for communication, and in contrast with objects, have the possibility to reason, interpret incoming messages, and take decisions according to its specific beliefs and objectives (Shen *et al.*¸ 2001). Just like OOP and software agent technology

changed the way the software programs are developed and left behind the hierarchically structured programming, the holonic-based approach to controlling the shop floor has the potential to become the operational standard for the next generation manufacturing systems.

As software development is a continuously developing area, in the last years another software architecture concept emerged. The newly arrived concept of component-based development (CBD) tries to add reusability and reconfigurability into software modules. Using CBD concept and adapting it to the holonic systems theory, Chirn and McFarlane (2000a) presented an architecture called Holonic Component-Based Architecture that has the goal to cope with rapid changes in manufacturing environments.

Some of the software tools most used in the development and implementation of holonic systems are reported below. Unified Modeling Language (UML) is widely used to model individual holons and the interactions among them, examples of using UML can be found in Bruckner *et al.* (1998), Rannanjarvi and Heikkila (1998), Valckenaers *et al.* (1998), Van Brussel *et al.* (1998), Fletcher *et al.* (2000), Hammerle *et al.* (2000), Shu *et al.* (2000), Heikkila *et al.* (2001), Depke *et al.* (2002), and Huang *et al.* (2002). The Common Object Request Broker Architecture (CORBA) architecture, a vendor-independent architecture that computer applications use to work together over networks appears also as a suitable modeling tool for holonic systems in papers such as that of Cheng *et al.* (2001). More recently a Java execution environment, JDPS, was adopted as a platform for developing holonic systems (Tamura *et al.*, 2003).

Coming from the collection of FIPA standards, the ACL is becoming increasingly used as the communication means in distributed architectures (Ritter *et al.*, 2003, Marik *et al.,* 2003). Besides using ACL, communication among entities in the holonic architecture is achieved also by using TCP/IP protocols (Shen *et al.,* 1997, Maturana *et al.,* 1999, Rannanjarvi and Heikkila, 1998, Arai *et al.,* 2001, Heikkila *et al.,* 2001, Heikkila *et al.,* 2003), Microsoft's COM/DCOM technology, which gives solutions for building entities that can communicate with each other (Brennan and O, 2000, Shu *et al.,* 2000, Unver and Anlagan, 2000, Schoop *et al.,* 2001), Knowledge Query and Manipulation Language (KQML), an agent communication language and protocol which has many similarities with the FIPA-ACL (Shen *et al.,* 1997, Maturana *et al.,* 1999, Leitao and Restivo, 2000, Heikkila *et al.,* 2001, Wang, 2001), or in the case of research systems, communication can be simulated using software objects residing in a single computer.

6.5. Real-Time Issues in Holonic Systems

According to Buttazzo (2002), real-time systems are defined as "systems that must react within precise time constraints to events in the environment." Thus, in real-time systems, not only the logical output has to be valid, but also the timing issues impose specific constraints. The characteristic that differentiate a real-time system from non-real-time systems is that the former have a pre-established deadline in which they are supposed to finish their tasks. Real-time tasks can be hard or soft. A real-time task is said to be hard when "missing its deadline may cause catastrophic consequences on the environment under control," whilst a real-time task is said to be soft when "meeting its deadline is desirable for performance reason, but missing its deadline does not cause serious damage to the environment and does not jeopardize correct system behavior" (Buttazzo, 2002). Both hard and soft real-time tasks appear in manufacturing systems.

At the equipment level controllers, tasks such as positioning and transporting materials can be considered hard real-time since missing timing constraints could lead to damaging the entire system. The real-time tasks we deal in production planning and scheduling area are soft real-time tasks, since nothing catastrophic will occur by missing a deadline. Meeting deadlines for task scheduling in manufacturing systems is desirable and missing them only results in reduced solution quality. Real-time control requires an effective method for measuring time. Since a global clock is not much of a help in a distributed computing environment, such as a holonic architecture, a more appropriate solution is the use of independent clocks for each of the holons, which can be reset every time when a timed request is received. Based on these considerations, Jennings *et al.* (1999) present a description of three time transfer protocols for clock synchronization and a characterization of distributed real-time systems.

A comprehensive discussion on the real-time requirements imposed on holonic systems can be found in Balasubramanian *et al.* (2001). Hard and soft real-time requirements are combined with other needed characteristics such as distributed, event-driven and intelligent control to develop a real-time distributed LLC system. Holonic systems are distributed by their nature, intelligent due to their software units, and event-driven as they react in response to changing environment, so these characteristics are appropriate to be included in the development of real-time holonic control architecture. Detailed design characteristics of the real-time distributed LLC system developed based on the IEC 61499 standards and its implementation can

be found in Zhang *et al.* (2000).

Based on the OOP and agent-based approaches, another research undertaken by Brennan *et al.* (2001) considers the real-time issues in HMS and aims at developing reconfigurable distributed control systems capable of delivering real-time response to any service request coming to the system. Deadline control as an important evaluation of real-time capabilities is studied by Fletcher *et al.* (2001). They developed a holonic architecture with capabilities for meeting established deadlines and presented several reconfiguration considerations for the design of manufacturing systems. A real-time control architecture viewed from the system, software, and functional architectures points of view is presented by Wang *et al.* (1998a), followed by an event driven real-time distributed control system developed via using a combination of intelligent agents and IEC 61499 function blocks.

6.6. Production Planning and Scheduling in Holonic Systems

In holonic control systems, the computations needed for manufacturing scheduling applications are distributed among the control units of the holons comprising the architecture. This is in contrast to traditional manufacturing control where a single central processing unit (CPU) is performing all the calculations needed for a specific planning or scheduling application. Therefore, for complex problems, an improvement in the time necessary to obtain a valid schedule is obtained, due to the fact that the distributed holonic controllers are required to solve smaller problems compared to the complex problem needed to be solved by the single CPU. Consequently, the holonic architecture allows for the possibility to attack large and complex problems which otherwise would be intractable in the traditional way. A valid solution is needed in many cases in real-time, a time-frame difficult to achieve when complex problems have to be solved on a single CPU. Even the solution obtained might not be optimal, by distributing the calculations over a number of controllers, real-time response can be achieved and complex problems become tractable.

Real-time scheduling is necessary in many industrial environments. The big difference between scheduling theory and practice is coming from the way the scheduling environment is perceived. While the scheduling theory considers the environment as static, real-world scheduling environment is definitely dynamic. So the schedules have to be frequently revised in

response to the changes made in the environment by the unexpected disturbances. There are two types of real-time scheduling. First uses the traditional scheduling theory and develops static schedules for all the jobs that need to be processed in a period. Then by using real-time information as it becomes available, works as a reactive system and revise the schedules to include the newly arrived information (Cowling and Johansson, 2002). The second approach is the application of software agent technology which constructs schedules using the existing resources whenever the status of the system changes. HMS is using this second approach for real-time allocation and scheduling of tasks.

A detailed description of a production planning and scheduling approach based on the CNP is presented by Vancza and Markus (2000). Starting with the order processing and task announcement, continuing with bidding mechanisms, task assignment and incentive-based mechanisms, and finishing with the final schedule generation and dispatching, all the steps of production scheduling are presented in detail. Many other papers (e.g., Gou *et al.,*1994, Ramos, 1996, Bruckner *et al.,*1998; Gou *et al.,* 1998; Rannanjarvi and Heikkila, 1998, Sousa and Ramos, 1998, Silva and Ramos, 1999, Sousa and Ramos, 1999a, Sousa and Ramos, 1999b, Zhang and Norrie, 1999, Bongaerets *et al.,* 2000, Cheung *et al.,* 2000, McFarlane and Bussmann, 2000, Shu *et al.,* 2000, Arai *et al.,* 2001, Brennan and Norrie, 2001, Heikkila *et al.,* 2001, Schoop *et al.,* 2001, Sugimura *et al.,* 2001, Hsieh, 2002, Huang *et al.,* 2002, Deen, 2003, McFarlane and Bussmann, 2003, Neligwa and Fletcher, 2003, Tamura *et al.,* 2003), considered the holonic approach to production planning and scheduling in manufacturing systems. Several of these papers are presented in the next chapter in the modeling of holonic enterprises and shop floor control systems discussion.

7. Applications and Implementations of Holonic Manufacturing Systems Concept

As holonic concept is considered a solution for next generation manufacturing systems by many researchers in both academia and industry, there is a vast number of applications and implementations of the holonic concept in manufacturing systems domain. The most relevant ones as identified in the literature review are the focus of this chapter. Table 7.1 gives a classification of the papers reviewed based on the area of application and implementation of the holonic manufacturing concept.

Table 7.1. Classification of the papers reviewed based on the area of application of the holonic manufacturing concept.

Applications of the Holonic Concept to Existing Manufacturing Systems	*Holonic Modeling of Manufacturing Enterprises*	*Holonic Modeling of Shop Floor Control Systems*	*Holonic Modeling of Material Handling and Logistics Systems*
Overmars and Toncich (1996); Gayed *et al.* (1998); Monostori and Kadar (1999); Chirn and McFarlane (2000a); Chirn and McFarlane (2000b); Lun and Chen (2000); Schoop *et al.* (2001); Jarvis *et al.* (2003); Monostori (2003).	Christensen (1994); Mathews (1995); Gou *et al.* (1998); Toh *et al.* (1998); Rahimifard *et al.* (1999); Toh *et al.* (1999); Leitao and Restivo (2000); Ulieru *et al.* (2000); Cheng *et al.* (2001); Fletcher and Deen (2001); Ulieru (2001a) ; Ulieru (2001b); Ulieru and Norrie (2001); Ulieru *et al.* (2001); Huang *et al.* (2002); Christensen (2003); Monostori (2003).	Gou *et al.* (1994); Ramos (1996) Bongaerts *et al.* (1997); Bruckner *et al.* (1998); Gayed *et al.* (1998) Gou *et al.* (1998); Rannanjarvi and Heikkila (1998); Sousa and Ramos (1998); Valckenaers *et al.* (1998); Van Brussel *et al.* (1998); Sousa and Ramos (1998); Silva and Ramos (1999); Sousa and Ramos (1999a); Sousa and Ramos (1999b); Zhang and Norrie (1999); Balasubramanian *et al.* (2000); Bongaerets *et al.* (2000); Cheung *et al.* (2000); Chirn and McFarlane (2000b); Fletcher *et al.* (2000); Hammerle *et al.* (2000); McFarlane and Bussmann (2000); Shu *et al.* (2000); Zhang *et al.* (2000); Arai *et al.* (2001); Balasubramanian *et al.* (2001); Brennan and Norrie (2001); Brennan *et al.* (2001); Cholski and McFarlane (2001); Fletcher *et al.* (2001); Giebels *et al.* (2001); Heikkila *et al.* (2001); Sugimura *et al.* (2001); Wang (2001); Heragu *et al.* (2002); Hsieh (2002); Wullink *et al.* (2002); Deen (2003); Heikkila *et al.* (2003); Jarvis and Jarvis (2003); McFarlane and Bussmann (2003); Neligwa and Fletcher (2003); Tamura *et al.* (2003); Hsieh (2004).	Cselenyi and Toth (1998); Liu *et al.* (2000); Bussmann and Sieverding (2001); Vrba and Hrdonka (2001); Heragu *et al.* (2002); Marik *et al.* (2003); Ritter *et al.* (2003); Bussman *et al.* (2004).

7.1. Applications of the Holonic Concept to Existing Manufacturing Systems

Several attempts to apply the holonic principles to existing manufacturing systems, such as FMS, to improve their responsiveness to unexpected events and achieve fault-tolerance are presented in the literature. Overmars and Toncich (1996) suggested a method to apply the holonic approach to scheduling in FMS by dynamically selecting the appropriate manufacturing resources for any given workpiece coming into the system. The same research topic, applying the holonic concept for FMS scheduling, is considered by Cheung *et al.* (2000) and by Lun and Chen (2000). In the first paper a series of prototype holons are implemented for real-time scheduling tasks in an existing FMS, while in the second paper, the conceptual design developed based on the holonic approach is compared to the traditional FMS scheduling approach. Different disturbance scenarios are simulated and improvement results are reported. Jarvis *et al.* (2003) presented a migration process from an existing control system to an intermediate stage for a manufacturer of automotive engines and engine components. In the intermediate stage, the holonic framework and a part-oriented concept coexist with the existing infrastructure. The results obtained so far in implementing the intermediate stage show that more research is necessary in order to have all the tools needed for the migration to a true holonic system, the final objective of the research.

Various other strategies for migration of existing systems towards holonic systems are considered as a first and easier to implement solution and are presented in several papers. These strategies include the use of diagnostic capabilities and limited reconfiguration associated with existing systems (Gayed *et al.,* 1998), transitions of existing CIM systems to holonic systems using mapping of holons to existing CIM architecture (Chirn and McFarlane, 2000a), holonification of existing resources by incorporating control units for each resource (Monostori and Kadar, 1999, Monostori, 2003), or transitions of existing manufacturing cells to holonic cells (Chirn and McFarlane, 2000b).

7.2. Holonic Modeling of Manufacturing Enterprises

Considering the definition of the holonic systems as holarchies in which holons are at the same time wholes and sub-wholes and that at any time a holon may be part of another broader

holon, several papers in the literature cite the possibility to model the entire structure of an enterprise as a holonic system. There are also papers that present models of a chain of enterprises structured as a holarchy. This aspect can lead to the possibility to model the entire supply chain of an enterprise based on the holonic approach, for the purpose of developing holonic-based supply chain management systems.

Studies such as those of Ulieru (2001a) and Ulieru (2001b) looked at the whole picture and illustrated manufacturing organizations as a network of enterprises grouped in a larger holonic enterprise. A shop floor is only a part of one of the enterprises involved in the larger holonic organization. This is a much larger view of the manufacturing picture and gives a good idea about the different levels of the holonic system. Even it looks like the holons are organized into levels of hierarchy the general master-slave relationship existing in proper hierarchies does not apply to these holonic organizations. Leitao and Restivo (2000) discussed the multi-enterprise model and present it in a structure similar to the general holonic architecture. The holon characteristic of being a part of another holon (i.e., a holon can be then broken into several other holons, which can be also broken into several other holons) allows for reduced system complexity, and consequently a multi-enterprise organization can be detailed in successive layers down to the equipment level of a single enterprise.

Toh *et al.* (1998) presented a solution to model small to medium enterprises (SME) using the holonic approach. Particularly, their model refers to a small metal-working company, and it included three main holons, the executive holon responsible for the most important decisions within the company, the business holon responsible for customer orders, inventory and other business related issues within the plant, and the manufacturing holon comprising the production resources and the control units associated with them and responsible for delivering goods to the shipping department. All these holons communicate and cooperate through the aid of the holonic information system support. Moreover, the main three holons are then composed of multiple sub-holons each of them having a well defined responsibility in its area. A more detailed modeling of this particular enterprise functionality and behavior is presented by Toh *et al.* (1999) where holonic principles are applied for the small enterprise and also for the detailed model of three holons part of the holonic enterprise, the production planning, machining, and purchasing holons.

Another model for a holonic SME is presented by Rahimifard *et al.* (1999). Executive, inventory, scheduling and manufacturing holons together with orders and manufacturing

databases form the holonic architecture of the SME. Additionally, a production planning and scheduling method and its software implementation called "Distributed autonomous real-time (DART) planning and control" is presented in the paper. Other solution for modeling enterprises as holonic organization include the formation of dynamic clusters of software and hardware entities created each time new tasks need to be executed within the enterprise (Ulieru *et al.*, 2001). "Patterns of holonic collaboration" are studied starting with the inter-enterprise level, going through the intra-enterprise level, and down to the physical machine level of the holonic architecture. Mechanisms for holonic collaboration are presented for inter-enterprise level where the focus is to move from the actual closed supply chain model to a more open system that allows dynamic collaboration among entities. Intra-enterprise level focuses on task allocation using reconfigurable software, and achieves fault-tolerance, while at the machine level the focus is on using distributed intelligent control technologies.

Table 7.2. Comparison of holonic frameworks for modeling manufacturing enterprises.

Holonic Framework / *Holonic Characteristic*	*Holonic Enterprise Model developed at the University of Calgary (Ulieru and Norrie, 2000, Ulieru et al., 2000, Ulieru, 2001a, Ulieru, 2001b, Ulieru et al., 2001)*	*Holonic Enterprise Model developed for Small to Medium Enterprises at Loughborough University, UK (Toh et al., 1998, Toh et al., 1999)*	*Holonic Enterprise Model developed by Huang et al. (2002)*
Scope	Inter-enterprise and enterprise-wide control	Enterprise-wide control	Inter-enterprise control
Reference Holons (Sub-Holons) Types	Enterprise (Resource Knowledge, Design, Product Model, Order, Resource Scheduling, Resource Control)	Executive Business (Purchasing, Production Planning) Manufacturing (Machine)	Virtual Enterprise (Member Enterprise (Planning, Task, Scheduling, Resource))
Holonic System Architecture	Three levels: Inter-enterprise, Intra-enterprise, Physical	Holarchy: Reference holons contain sub-holons	Holarchy: Reference holons contain sub-holons
Decision-Making	Mediator holons included in each of the levels	Individual holons or sub-holons based on their functions	Member Enterprise holons with mediation from Virtual Enterprise holons

A framework for modeling virtual enterprises (VE) using the holonic approach is presented by Huang *et al.* (2002), where the VE is formed by a group of distributed, autonomous and cooperative member enterprises (ME). Each ME holon has four corresponding sub-holons, planning, scheduling, task and resource holons. These holons can be further detailed in smaller holons, if needed. For example the resource holon is further comprising factory, shop floor,

workstation and cell holons. UML is used to present the interactions between holons in the proposed framework. Just like in the holonic shop floor resource allocation, where scheduling is performed among autonomous and cooperative holons with advices received from a global view entity, at the virtual enterprise level, resource sharing is performed among autonomous and cooperative ME holons with assistance from the VE holon. Table 7.2, above, presents a comparison of three of the holonic frameworks for modeling manufacturing enterprises presented in this section.

7.3. Holonic Modeling of Shop Floor Control Systems

In a manufacturing facility the shop floor includes all the equipment, devices and controllers that aid in transforming raw materials into finished products. To efficiently control the activities on the shop floor, the shop floor control system (SFCS) needs to take into account all the orders and corresponding process plans received from production planning department and execute them according to predefined objectives which can include: an acceptable level of quality, timing constraints, pre-established throughput levels, or resources utilization constraints. A traditional hierarchical SFCS cannot change the schedules it delivers when some predefined system status is modified. A holonic SFCS, on the other hand, can cope with these changes, by having the possibility to add and delete the holons in the architecture as imposed by the status of the system at each particular time.

The characteristics of the holons as potential part of two or more holons simultaneously is considered in the holonic shop floor control architecture developed at the Instituto Superior de Engenharia do Porto by Ramos (1996), and Sousa and Ramos (1998, 1999a). The resource holons are simultaneously part of three holons, the production planning, scheduling, and process planning holons, and the task holons are simultaneously part of other two holons, the production planning and the scheduling holons. The same idea appears in Sousa and Ramos (1999b) where several functional units of the shop floor are modeled as holons and there are multiple intersections across the functional units in the architecture. Silva and Ramos (1999) modeled a dynamic scheduling manufacturing system using the same approach with scheduling, production planning and process planning holons comprising common elements for more than one holon. All these papers make use of two important properties of holons, first a holon can be formed of

other holons, and second, that a holon can be part of several larger holons.

PROSA (Product Resource Order Staff Architecture) is a HMS architecture developed at PMA-KULeven (Van Brussel *et al.,* 1998) and aims to be used as reference for future HMS implementations. It gives the basis of the architecture design, defines the terminology within the architecture, presents the component entities and their responsibilities within the architecture, and also gives examples of test-bed implementations. Three basic holons are the main components of PROSA reference architecture namely, product, resource and order holons. Their responsibilities are covering the main aspects of manufacturing control, product and process related planning, shop floor resources, and manufacturing tasks respectively, making it a good modeling tool in the design of SFCS. The other type of holon defined in PROSA, the staff holon, aids the basic holons with its global view of the system, and can deliver optimal plans for particular manufacturing situations.

Valckenaers *et al.* (1998) argued that PROSA, aiming at being reference architecture, is a good "starting point for the design and development of holonic manufacturing control systems." And, as suggested by authors, PROSA was applied by other researchers to develop holonic manufacturing control architectures and implement them in test-bed systems. As an example, Liu *et al.* (2000) used PROSA to develop the control architecture for an AGV system which will be discussed in one of the next sections. Another PROSA-based system was developed under the MASCADA (Manufacturing Control Systems Capable of Managing Production Change and Disturbances) project for a car body painting at a German plant (Bruckner *et al.*, 1998). In the MASCADA project, the PROSA reference architecture is the starting point for a more detailed system which includes all the specific aspects of the car body paint facility. As the holon and agent terms are used in many situations interchangeably in the distributed manufacturing control architecture literature, the paper adopts the term agent, instead of holon for the basic entities in PROSA. Related to the MASCADA project, the ManAge manufacturing control architecture (Heikkila *et al.,* 2001) uses agent technology to develop a distributed control architecture and provides a clear differentiation of agents and their functions in the architecture, enables scalability, and gives a detailed framework of inter-agent and inter-process communication. Specific to ManAge is the implementation of a belief database which is shared by the agents in the architecture, giving it an improved reaction to changing environments.

EtoPlan (Engineer-to-order Planning) is a holonic architecture for manufacturing

planning and control developed at the University of Twente in Netherlands designed to work within a manufacture-to-order environment (Wullink *et al.,* (2002). The EtoPlan architecture is based on multiple and temporary holarchies of applicable resources, called applicability groups (AG). Each activity on the shop floor has an AG associated with it, which groups all the resources needed to perform that specific activity. EtoPlan aims at dealing with large amounts of uncertainty, caused by incomplete and unreliable information. A prototype software implementation and more information about EtoPlan can be found in Giebels *et al.* (2001). Table 7.3 summarizes the main characteristics of the EtoPlan, PROSA, and the holonic framework developed at Instituto Superior de Engenharia do Porto.

Table 7.3. Comparison of holonic frameworks for modeling shop floor control systems.

Holonic Framework / *Holonic Characteristic*	*The Holonic Framework developed at Instituto Superior de Engenharia do Porto, Portugal (Ramos, 1996, Sousa and Ramos, 1998, Sousa and Ramos, 1999a, Sousa and Ramos, 1999b)*	*The Holonic Reference Architecture (PROSA) developed at Katholieke Universiteit Leuwen, Belgium (van Brussel et al., 1998, Valckenaers et al., 1998)*	*The Holonic Architecture (EtoPlan) developed at University of Twente, The Netherlands (Giebels, M. M. T. et al., 2001, Wullink, G. et al., 2002)*
Scope	Production Planning and Scheduling	Entire Manufacturing System	Production Planning and Control
Reference Holons (sub-Holons) Types	Task Manager Production Planning Task Resource	Basic: Product Resource Order Assistant: Staff	Applicability Group (AG): group of resources that are applicable for the execution of an order
Holonic System Architecture	Holarchy: Reference holons include sub-holons; A holon can be part of several broader holons	Holarchy: Reference holons include sub-holons	Temporary Holarchy: AG, Parent-AG, Peer-AG, Resource, Child-AG. Resource can be member of multiple AGs
Decision-Making	Task holons Task manager holon deals with uncertainties	Basic holons Assistant holon provides advice only	AG controller

Shop floor control is usually the first stage for the application and implementation of the holonic concept to industrial situations. Cholski and McFarlane (2001) presented their vision of implementing the holonic approach in the chemical process industries. The paper presents both the particular characteristics of this industry and specific ways to implement the holonic principles. Rannanjarvi and Heikkila (1998) applied the holonic concept in the development of

the control software for surface treatment robots. They presented a very detailed software development process and include models and tools used in analysis of the requirements for holonic systems, models for system design, models for structure design, models for design of behavior, and implementation models. Chirn and McFarlane (2000b) developed an architecture structured on a modular mix of standardized, autonomous, cooperative and intelligent components capable to cope with rapidly changing environments. They implemented the architecture for a robot assembly cell and report an encouraging preliminary performance evaluation, which include system integration and communication infrastructure.

7.4. Holonic Modeling of Material Handling and Logistics Systems

Most of the research in the HMS area did not focus specifically on the MH task, so material movement within a manufacturing system is not much presented as potential implementation for holonic-based systems. However, several examples that consider the MH aspect are presented here. The agent-based implementation for a MH system presented in Vrba and Hrdonka (2001), Marik and Pechoucek (2001) and Marik *et al.*, (2003) is formed of a sorting system that includes several conveyors, diverters and storage units and supports two different types of dynamic reconfigurations. First one, called light-weight reconfiguration, when a resource failure is detected, and the second one, called heavy-weight reconfiguration, when the architecture of the systems is changing by adding or removing some of its physical elements.

Based on PROSA, the holonic reference architecture discussed above, a number of researchers developed holonic control frameworks to operate different parts of manufacturing systems, but only a few considered the MH system specifically. Liu *et al.* (2000) developed a holonic control architecture for an AGV system by further detailing PROSA to incorporate the AGV system, capable of being robust in the presence of disturbances. The architecture developed uses CNP as the communication methodology, and simulation tools for performance evaluation. The reported results show that the holonic system can offer considerable improvement compared to the traditional hierarchical control.

In the hybrid manufacturing control architecture, placed between hierarchical and totally decentralized architectures, developed at Rensselaer Polytechnic Institute (Heragu *et al.*, 2002, Kim *et al.*, 2004) the entities comprising the architecture are referred as both agents and holons

interchangeably. The particularity of this architecture resides in the fact that it has an intermediate agent responsible for the decision making in the control architecture. A higher level global view agent has the ability to deliver optimal schedules, while the lower level agents are responsible for preparing the individual schedules, but they need to get permission from the intermediate level agent. The intermediate level agent, called middle level guide agent, has a larger system view than lower level agents, so it acts as a coordinator for the actions of lower level controllers. The authors argued that this architecture is more suited to satisfy the new requirements imposed on manufacturing system operations since it enables communication and decision-making in both horizontal (as in a heterarchical structure) and vertical (as in a hierarchical structure) directions among the entities in the architecture. The architecture was used to model a gantry robot system for order picking operations. A comparison of the basic characteristics of this holonic framework developed by Marik and Heragu, mentioned above, is presented in Table 7.4.

Table 7.4. Comparison of holonic frameworks for modeling material handling systems.

Holonic Framework / *Holonic Characteristic*	*The FIPA-Compliant System for Material Handling Control developed at Czech Technical University and the Rockwell Automation Research Center, The Czech Republic (Marik et al., 2003, Marik and Pechoucek, 2001, Vrba and Hrdonka, 2001)*	*The Hybrid Manufacturing Control Architecture developed at Rensselaer Polytechnic Institute, USA (Heragu et al., 2002, Kim et al., 2004)*
Scope	Material Handling Control	Production Planning, Control and Scheduling
Reference Holons (Sub-holons) Types	Workplace Agent (Node, Storage, Machine) Crossing Agent (Diverter, Intersection) MH System Agent (Conveyor)	Order/Part Machine/Material Handling Device Middle level Cell/System
Holonic System Architecture	Decentralized, composed of autonomous and cooperative agents	Low level: Machine, Material Handling Device Middle level: Guide Higher level: Cell/System
Decision-Making	Crossing Agent	Order holons with permission from middle level holons

Within the HMS Consortium project, the Holomobiles work-package (Bussmann *et al.*, 2004) analyzed the limitations of the material flow system in existing engine assembly systems.

The aim of the Holomobiles work-package was to optimize the material flow in industrial manufacturing systems served by AGV systems. Using the holonic concept a new material flow design was proposed. The new design demonstrated improvements in the robustness and volume flexibility of the assembly process. Cselenyi and Toth (1998) presented a discussion of the logistics related issues in holonic systems. The paper gives the necessary features for holonic systems and the requirements that need to be satisfied from the logistics point of view. Collecting the transport demands, determining the delivery deadlines and sequences, and providing the transportation means are only a few of the capabilities of the system proposed. In the holonic transport system presented in Ritter *et al.*, (2003), the holonic AGVs are capable of deriving a demand for orders individually and at the system level the cooperation mechanism prevents and resolves the potential deadlock conflicts.

7.5. Industrial Applications of Holonic Systems

Presently, the industrial acceptance of holonic systems is relatively low due to reasons inherent to the holonic systems development process and as a consequence of companies' business strategies. So, to date, there are only a few real-world implementations of holonic systems. The holonic shot-blasting system presented in Heikkila *et al.* (2003) is a demonstration of how separate robots can both act autonomously and cooperate with each other to complete the needed tasks. The Holomobiles project mentioned in the previous section and developed for existing engine assembly systems in the automotive industry offered robustness and scalability as additional resources can be added easily to the system, unprecedented achievements in existing assembly systems (Bussmann and Sieverding, 2001). Two other holonic industrial implementations reported in literature were mentioned briefly in the previous sections. The first one is the model-based holonic diagnosis developed for an Australian automotive assembly plant (Jarvis and Jarvis, 2003), and the second one is the industrial automated warehousing system developed by Heragu *et al.* (2002) to which an intelligent holonic scheduling and control framework is applied for order picking and replenishing.

PART III

PROPOSED HOLONIC-BASED CONTROL SYSTEM

8. General Holonic Control Framework

In the holonic manufacturing literature, holons that include a manufacturing resource are usually called resource holons. However, when a more detailed image is necessary, the resource holons are divided in categories corresponding to the manufacturing resources involved, such as machine holons, MH holons, AGV holons, etc. Depending on the holonic architecture developed, the literature presents other types of holons that are part of the holonic architecture, for example, part holons, order holons, scheduling holons, shop floor holons, etc. To help in achieving coordination many holonic systems also include an entity having a global view of the system, which resembles the control unit of the centralized architecture or the higher level controller of the hierarchical architecture.

Based on the above considerations this chapter presents an innovative holonic control framework that takes into account all the categories of hardware and software entities and modules existing in a manufacturing system. The proposed architecture can be used for task allocation and scheduling of all activities needed to be performed within a manufacturing system.

8.1. Schematic Holonic Control Framework

Just like the FMC installed in Virginia Tech's FMS Lab (Figure 3.2), a small automated manufacturing cell or system contains one control unit, processing machines, MH resources and other pieces of equipment that work together to perform the required operations on the incoming workpieces. By grouping the hardware and software parts of the system based on their functions and applying both the holonic concepts and the decentralized control approach, a holonic architecture to control manufacturing cells or systems can be developed. Using the same notations for the control units and manufacturing resources as stated previously, this holonic control architecture is depicted in Figure 8.1.

Having more than one control unit, this architecture is different from the traditional centralized control architectures used for small manufacturing systems or cells which have only one control unit for the whole manufacturing system. It can be seen that the flow of information between control units is always bi-directional due to the decentralized nature of the architecture and the application of the holonic concept of cooperation.

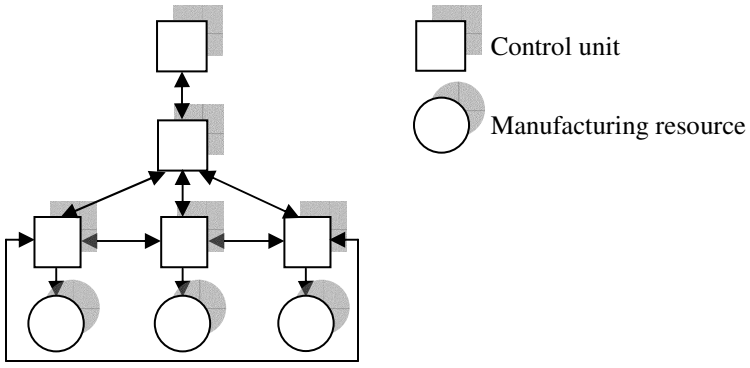

Figure 8.1. Schematic representation of the proposed holonic architecture.

8.2. Detailed Holonic Control Framework

The proposed holonic architecture is formed by five types of entities: Order Holons, three types of Resource Holons (Machine, Material Handling and Equipment Holons), and another entity called Global Scheduler (GS) that holds a general image of the entire system. This more detailed holonic architecture is depicted in Figure 8.2. In the holonic system, an Order Holon (OH) represents a job and all its associated information embedded in one control unit, while the Resource Holons (RH) are represented by the physical manufacturing resources, each of them having its own control unit.

The Machine Holon (McH) is formed by all the processing machines existing in the system together with their associated control units, and is responsible for the physical transformation of the workpieces from raw material into finished products. The Material Handling Holon (MHH) is formed by all transportation systems, such as robotics and AGV systems, together with their control units, and is responsible for moving the raw materials, work-in-process, and finished parts in the manufacturing cell. The Equipment Holon (EH) contains all pieces of equipment existing in the manufacturing system other than processing machines and the moving equipment represented by the Material Handling Holon. Every manufacturing system has a basic transportation system such as a robotics system or an AGV system. All other pieces of equipment that are part of the manufacturing system form the Equipment Holon. Examples of Equipment Holons include: measuring systems, intermediate buffers, quality control system (e.g., vision systems), inbound and outbound conveyors that move the product within the reach

of the robotics or AGV systems or out of the system after processing is completed.

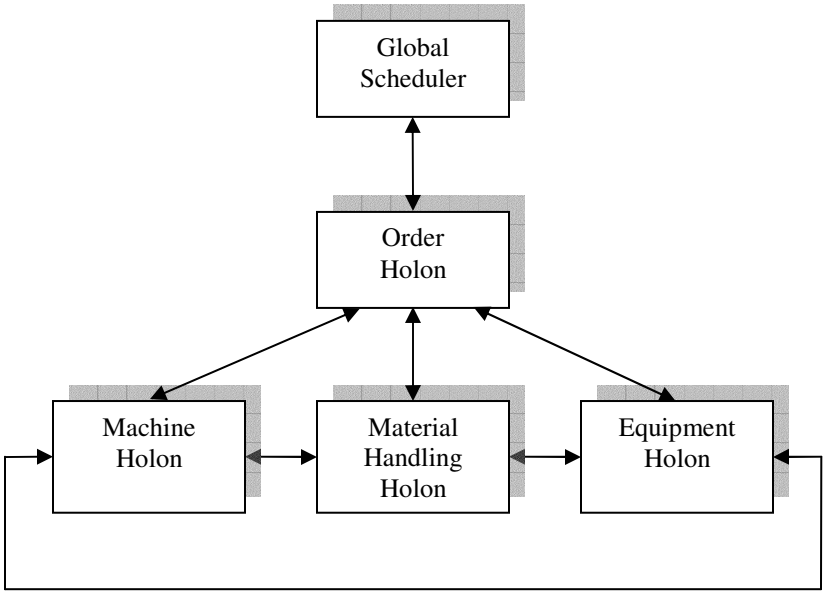

Figure 8.2. Proposed holonic control architecture.

These systems do not need special algorithms for scheduling the arrival of parts to their working area. They only need to perform their predefined actions on the parts that are entering in their work area. However, they also need some kind of intelligent control unit to exchange information about breakdowns, and in the case of the intermediate buffers, to reject any new coming part when their maximum capacity is reached. Measuring devices need to compare the raw materials or finished product dimensions with the information received from the Order Holons. The same operations are performed by the vision system, which based on its form recognition software is accepting or rejecting finished parts, and sends back this information to the Order Holons, such that they acknowledge the number of parts rejected, and thus necessary to be replaced.

To monitor the number of jobs and the availability of resources in the system, and to keep track of the already executed jobs, another entity called System Monitoring and Database (SMD) is introduced in the general holonic architecture. Considering a centralized manufacturing system, like the one installed in Virginia Tech's FMS Lab, all its components are represented in the proposed holonic architecture. This general holonic architecture, presented in Figure 8.3, is

further discussed in the next chapter with emphasis on the MH resources as manufacturing resources.

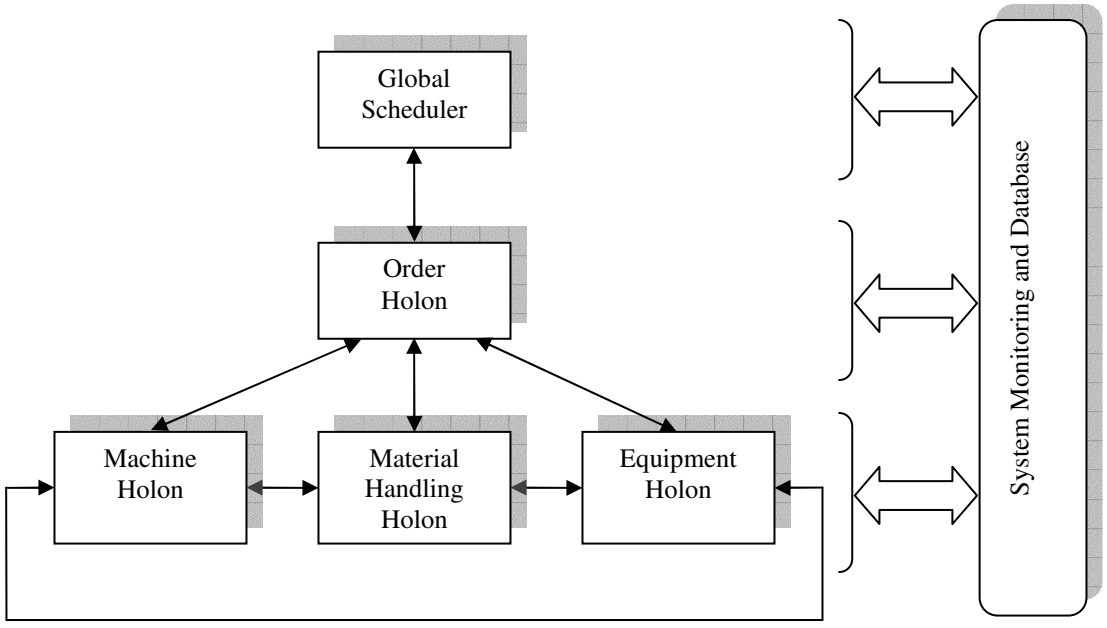

Figure 8.3. Entities in the general holonic control architecture.

8.3. Holonic Control Framework at the Shop Floor and Enterprise Levels

Just like in Koestler's definition of holons and holarchies, in the proposed general holonic control framework, the holons in the architecture may be formed by a number of sub-holons. During operations, there could be more than one job to be processed at a time, and definitely, there are more than one machine and one MH resource, and, possibly, more than one auxiliary equipment in every system. Using Koestler's concept of OEH presented in a previous chapter, the entities in a holonic manufacturing cell are represented as depicted in Figure 8.4.

When viewed from the cell level the proposed architecture is a combination between Koestler's holonic concept, and the decentralized systems theory. Holons are allowed to exchange commands and information among them both at the same level and with holons of higher or lower architecture levels. Because of the large number of relationships possible in the system, the arrows indicating them are not represented in the figure. Basically, any entity can

have relationships with any other entity in the architecture.

Looking further to the shop floor and enterprise levels the holonic architecture will contain more than one cell holon. The relationships between the holonic entities in this architecture are very complex and change frequently as more jobs are coming into the system or their execution completes. Temporary holarchies are formed based on the needs of the specific jobs existing in the system. Applying the OEH concept the holonic architecture at the enterprise level is presented in Figure 8.5 and resembles the structural organization of an army mentioned in a previous chapter. Once again, no relationships are represented in the figure, due to their large number and the possible associations between holons in different temporary holarchies.

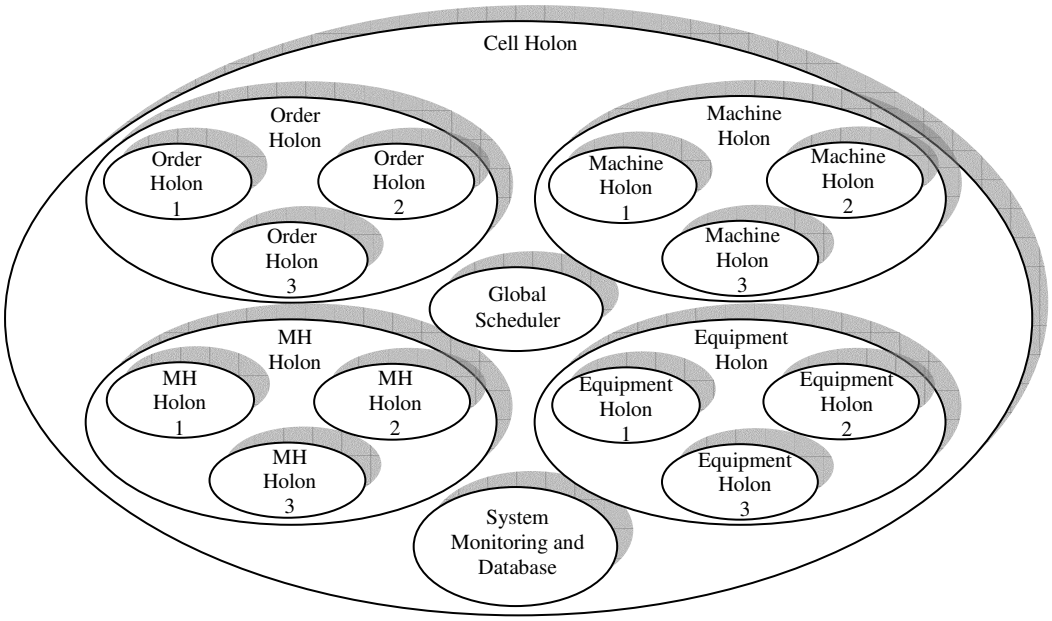

Figure 8.4. The holonic architecture viewed from the cell level.

These structures constructed on levels of OEH make the cell and enterprise architectures easier to understand, and also give the possibility to model complex organizations having a large number of departments and resources. While the OEH concept makes the architecture easier to understand, its decentralized structure makes it easier to react to any kind of changes that might occur. The direct exchange of information between holons at the same OEH level, or between holons at different OEH levels, does not mean necessarily for example that a Material Handling

Holon will have direct communication with the shipping department, the actual exchanges of information are dictated by the specific processing needs for the jobs existing in the system.

If, either the cell or the enterprise holonic architectures described above, needs to capture the association of holons based on their functions (OHs, RHs, etc.), then the possible relationships dictated by specific job assignments between OHs and RHs will not be captured. On the other hand, if, either the cell or the enterprise holonic architectures described, needs to capture the association of holons based on their specific jobs to be executed (a particular OH, and several RHs), then the functional organization of holons will not be captured. Thus, in Figures 8.4 and 8.5, no permanent or temporary relationships among the holons in the cell or enterprise organizations are presented. To represent these relationships when modeling cell or enterprise organizations using the holonic framework presented, only the holons involved in performing a specific job are considered. In this way, the number of relationships among them is reduced considerably and can be illustrated within the holonic framework.

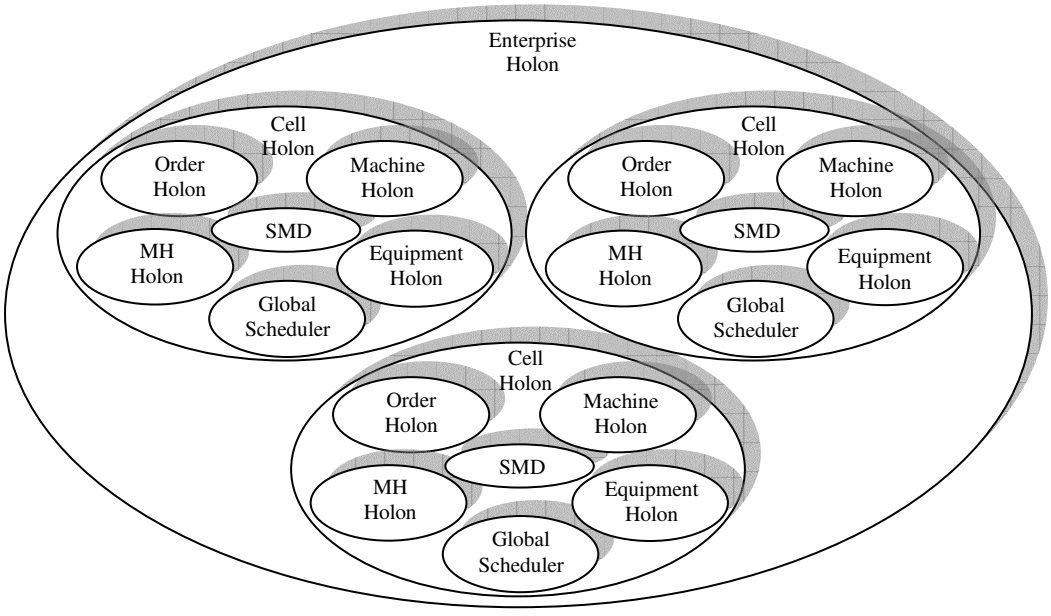

Figure 8.5. The holonic architecture viewed from the enterprise level.

9. Material Handling Holonic Control Architecture

In the general holonic control framework, the MH System is one of the three types of manufacturing resources modeled as autonomous and cooperative holons. Based on this general holonic framework, a holonic control architecture used to perform L/U tasks within a manufacturing system is presented in this chapter.

As the definition of the OEH states that it is unbounded in either direction, and as the concepts of holonic organization are valid on the entire OEH, it then results that even the HMS concept aims at modeling an entire enterprise, when trying to represent only a part of it such as the MH System within a manufacturing cell, the holonic principles do stand, so the kind of control that is applied to this part of the enterprise is also holonic control. Because there are no hard boundaries among holons in the entire architecture they can communicate and cooperate to execute tasks with other holons no matter where they are placed in the architecture.

9.1. General Material Handling Holonic Control Architecture

The objective in controlling MH resources is to find the best possible operation assignments and the right sequences and their associated schedules such that the performance of the manufacturing system is close to optimality. The MH holonic control architecture is derived from the general holonic control framework developed for the entire manufacturing system, presented in the previous section.

Because the interest is to control the resources or equipment that move materials within the manufacturing system, not all the entities in the general holonic control framework need to be considered when modeling only the MH system. Moreover, the processing operations of any job are considered to be already assigned to the machines and their processing sequences and schedules already known before any transport operation is scheduled. Removing the Machine and Equipment Holons from the general holonic architecture and de-aggregating the Material Handling and Order Holons in their number of sub-holons, leads to the creation of a new architecture for addressing only the material movement within the manufacturing system. This architecture, presented in Figure 9.1, contains one or more Order Holons and several Material Handling Holons corresponding to the number of transporting resources in a hypothetical

manufacturing system.

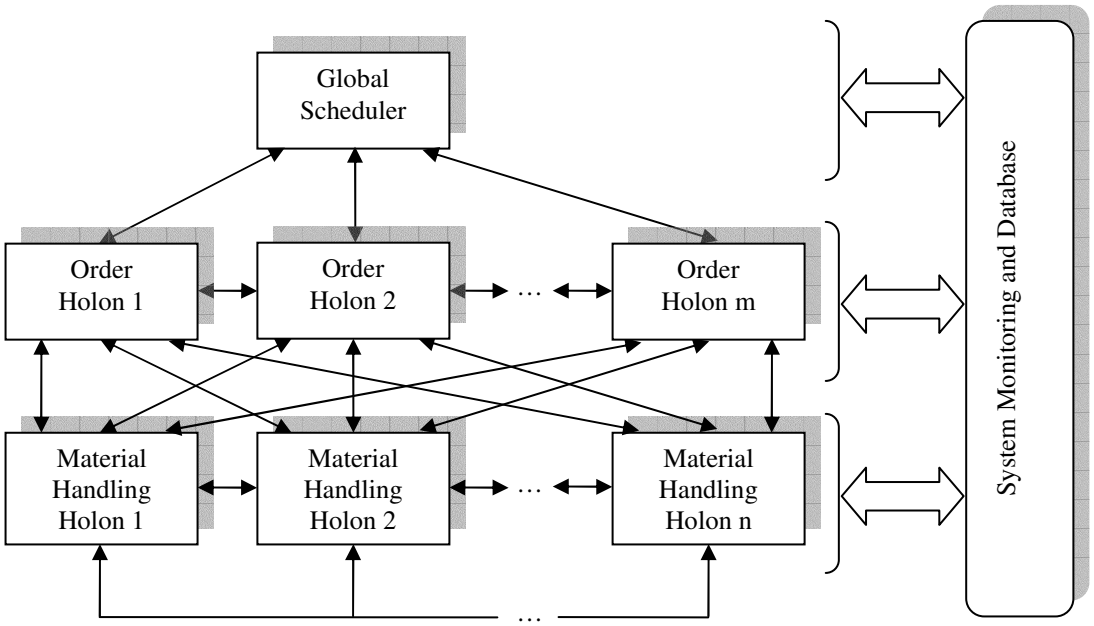

Figure 9.1. Material handling holonic control architecture.

9.2. Entities and their Functions in the Holonic System

All the entities in the MH holonic architecture are involved in preparing the L/U operations in the holonic system. The MH holonic resource allocation process considers information exchanged among all the entities in the system. Except for the System Monitoring and Database module, all other software entities have internal evaluation algorithms embedded in their structure, based on which they make the allocation decisions corresponding to their functions in the holonic architecture. To describe how the holonic system operates and what the role of each entity in the system is, an architecture having *m* Order Holons, and *n* Material Handling Holons is considered (Figure 9.2). The functions of the entities working in the holonic architecture related to the development of the global schedule is presented on both sides of Figure 9.2. As seen from the figure not only there are several types of entities involved in the decision-making process, but an individual decision type is also distributed among entities of the same category.

9.2.1. The Global Scheduler

In the holonic architecture presented in Figure 9.2, the Global Scheduler is basically a control unit that can deliver optimal or close-to-optimal schedules for the MH equipment when the system is operating under normal conditions and the complexity of the scheduling problem permits. Based on its global image of the MH System and the number of jobs in progress at any given time, the Global Scheduler computes the global schedule like the central control unit in the centralized architecture, or the scheduler module in the hierarchical control architecture. Because of the real-time requirements set on the system and the computational complexity of the problems to be solved, the Global Scheduler will not be able, in most situations, to deliver an optimal schedule when a request for a new part-type is received. Rather, a schedule constructed by the Global Scheduler can be used when the system operates under normal conditions (i.e., another order for the same part type has been done before, there exists an optimal schedule for it, and there are no other jobs in the system that require the use of one of the transporters used in the optimal schedule).

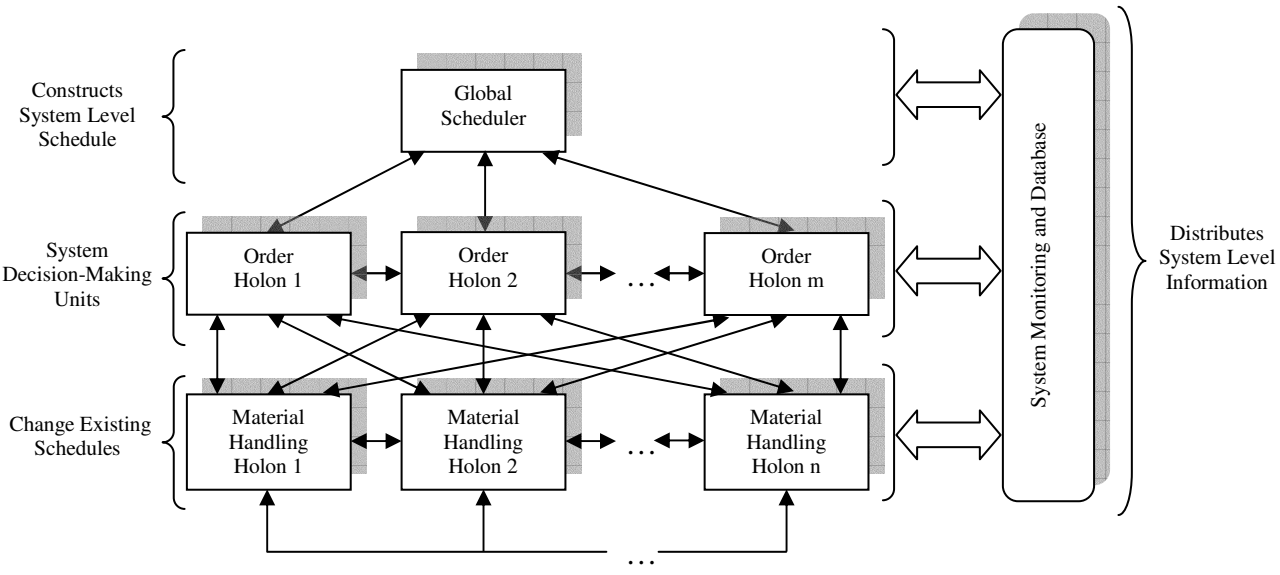

Figure 9.2. Entities and their decision-related functions in the proposed holonic system.

Another important task performed by the Global Scheduler is to use its global view of the system, to construct the resulting System Level Schedule (SLS), by combining the individual

schedules received from the active Order Holons. If the Global Scheduler succeeds in delivering a global schedule in the required time, it then compares it, using the overall system objective function, with the resulting System Level Schedule obtained by combining individual Material Handling Holons schedules. The better schedule is then sent to the System Monitoring and Database module for distribution to all entities in the system.

9.2.2. The Order Holons

The relationships between the holons in the proposed architecture are based on a decentralized approach like those in a heterarchical structure. However, contrary to the heterarchical architecture where there is no dedicated manager for the jobs coming into the system, in the MH holonic architecture presented above, a new dedicated Order Holon is created whenever a new job enters in the holonic system. The existence of dedicated managers for every job and the presence of the Global Scheduler are adding more predictability to the holonic architecture compared to a pure decentralized approach.

Based on internal evaluation algorithms, the Order Holons compare the transport offers received from the Material Handling Holons, and assign the transport jobs by choosing the best offer for each of the transport requests submitted. From this point of view, the Order Holons can be viewed as the system decision-making units. After assigning the jobs to the Material Handling Holons, the Order Holons send the individual Material Handling Holons schedules to the Global Scheduler, which, based on its global view, is combining these individual schedules to obtain the overall System Level Schedule.

9.2.3. The Material Handling Holons

The second type of holons in the architecture, the Material Handling Holons, is formed by the physical MH resources together with their own control units. Their main tasks are to execute all the movement activities in the manufacturing system, and to prepare appropriate offers for every new transport job that is issued by the Order Holons. Having their own control unit, the Material Handling Holons can decide their course of action based on their status and future workload by pricing differently the transport jobs received from the Order Holons. Based

on the autonomy and cooperation, characteristics specific for a holonic system, the Material Handling Holons can make changes in the System Level Schedule when one of them faces unexpected heavy workload, or it requires maintenance. Predefined Material Handling Holons cooperation mechanisms and protocols are used to operate these changes, without any involvement from the Order Holons.

9.2.4. The System Monitoring and Database Module

The System Monitoring and Database module has no decision power; rather it works as a database for already performed jobs and as a monitor for the availability of the MH resources in the system. It communicates with all entities in the system and transmits the MH equipment availability data to the Order Holons and the Global Scheduler. Order Holons need to find out which Material Handling Holons are available in the system such that they know to whom to transmit new transportation requests for offer preparation, while the Global Scheduler needs to know which Material Handling Holons are available for use to compute appropriate operations assignments and schedules. Other functions of the System Monitoring and Database module are to create the Order Holons whenever a new order enters in the system, delete the Order Holon when all the transport tasks associated with this holon are completed, and update the database with the information related to the executed jobs for future reference.

9.3. Temporary Associations among Entities in the Holonic System

One of the specific characteristics of holonic systems is that they can form temporary holarchies (Koestler, 1968). As Material Handling Holons could execute tasks for more than one Order Holon in the same period, there are several temporary associations between Order and Material Handling Holons during the operation of the manufacturing system. Using a model formed by two Order and three Material Handling Holons, two possible temporary associations are presented in Figure 9.3.

The first one contains one Order Holon and two Material Handling Holons, while the second one is formed by one Order and one Material Handling Holons. The third Material Handling Holon is idle in this particular instance of the operation of the system, so it can respond

to help requests received from the other two Material Handling Holons to ease their current workload. The temporary associations showed in the figure apply only for the offer preparation stage. In the subsequent stages, System Level Schedule generation and execution of MH tasks, both Global Scheduler and System Monitoring and Database module are involved, so they should be included in these associations, as well.

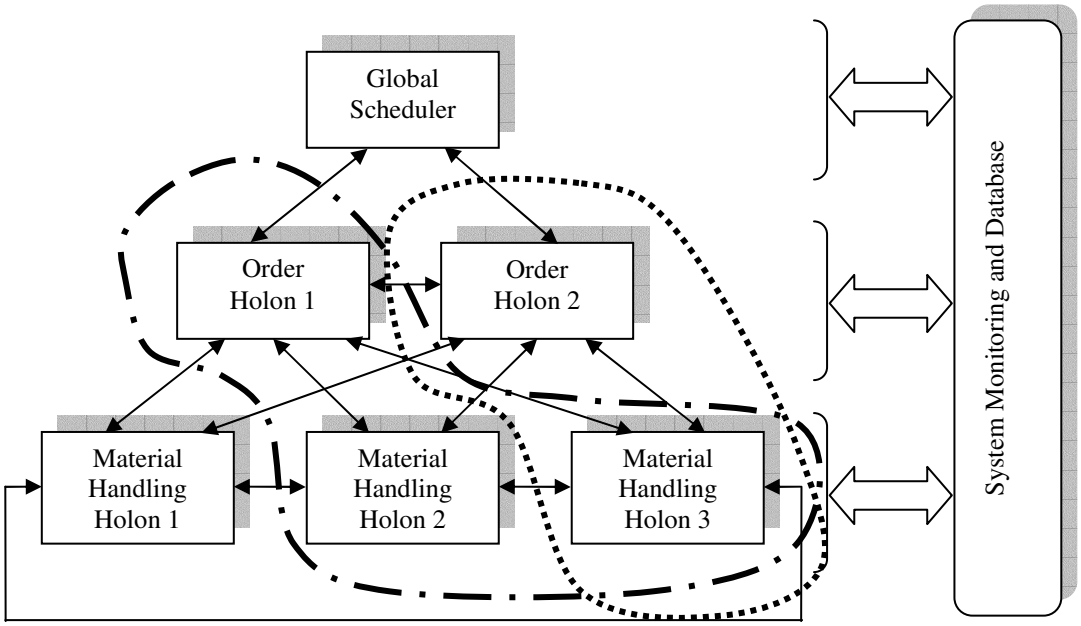

Figure 9.3. Temporary association among the entities in the holonic architecture.

9.4. Internal Architectures of the Entities in the Holonic System

The entities in the proposed holonic architecture have comparable structures in terms of the components forming their architecture, but the internal organization of these components is specific for each type of entity. The four main components included in the structure of the entities in the holonic architecture, Evaluation, Coordination, Internal Database, and Learning are modeled as software modules, and each of them has a specific capability: computational, exchanging information and coordinating actions for the purpose of achieving individual or common goals, storing and retrieving data, and machine learning respectively. Moreover, individual utility functions embedded in the Evaluation Module (EM), and expressed as

objective functions to be optimized, could vary also among the same type of holons. The internal structure of Material Handling Holons is presented in Figure 9.4, that of the Order Holons in Figure 9.5, and the architecture of the Global Scheduler is depicted in Figure 9.6.

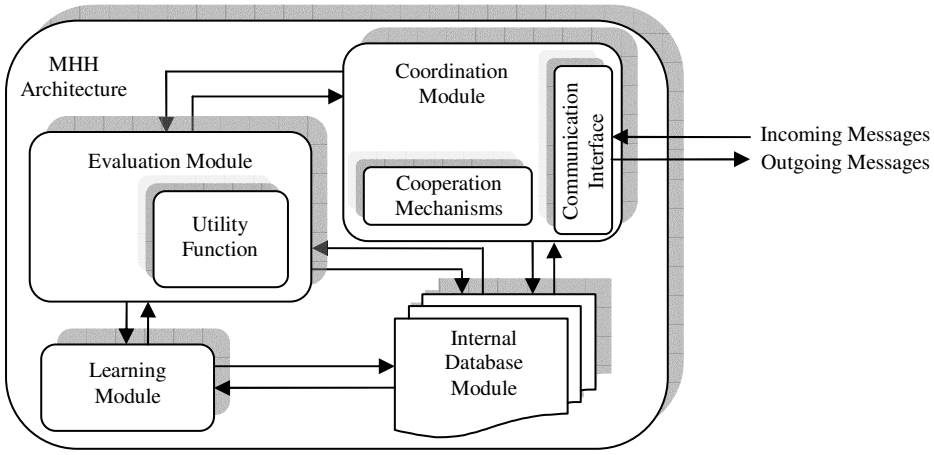

Figure 9.4. Internal architecture of a Material Handling Holon.

9.4.1. The Evaluation Module

The Evaluation Module is the place where all the computations are performed within the holonic structure, and it is formed by the Problem Solver (PS), and the Utility Function (UF). The Problem Solver contains the algorithms used for task evaluation process and is doing all the computations required for any problem needed to be solved. The Utility Function contains the objective function of the holon which needs to be optimized. The Evaluation Module takes the information received from the Coordination Module (CM) and after performing the calculations and individual holon goal optimization process, decides the actions that the holon will take.

For a Material Handling Holon, the internal Evaluation Module is responsible for assessing the value of the incoming service requests in terms of holon objectives and selecting the ones to be executed. Both, the algorithm used for the evaluation process and the internal Utility Function, expressed as objective function to be optimized, could vary among the holons in the architecture. For an Order Holon, the Evaluation Module is performing the initial preparation of transport requests and the subsequent evaluation of the transport offers received.

Because of the lack of global view, the Utility Function that provides the Order Holons objective is always associated with completing the job as soon as possible. It cannot accept other objectives since, any other objective viewed from only the individual job level, translates in minimizing the completion time of the job. In the case of Global Scheduler, the Evaluation Module includes a series of optimal and heuristic scheduling algorithms that attempt to provide a global schedule considering the overall system objective contained in its Utility Function.

9.4.2. The Coordination Module

Coordination in the proposed holonic system can be defined as the exchanging of information and the aggregation of processes performed for the purpose of achieving individual and system level goals. Depending on the entity in the architecture, the Coordination Module includes both general (Communication Interface sub-module (CIm)), and particular sub-modules (Cooperation Mechanisms sub-module (CMm) and Schedule Generation Mechanisms sub-module (SGM)). Communication is the process of exchanging data between any two entities in the holonic architecture performed for the purpose of achieving individual or system level goals. Cooperation is defined as the processes performed between two or more resources without any involvement from the job managers that attempts to improve the value of the system objective function. The Schedule Generation Mechanisms sub-module is designed for constructing the System Level Schedule using algorithms that account for the system objective function.

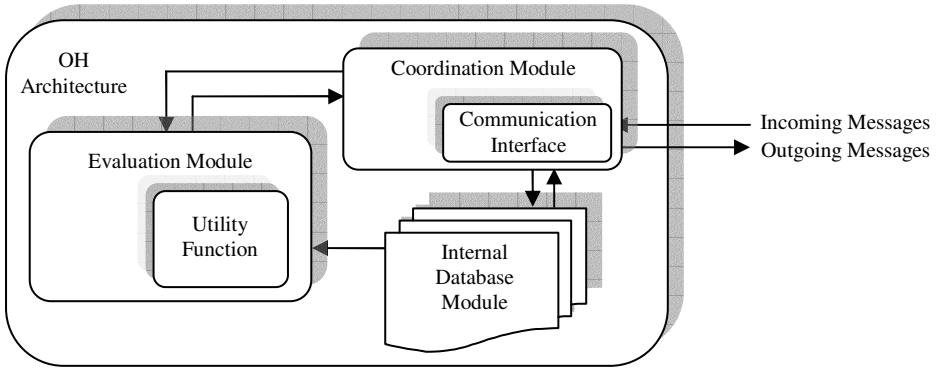

Figure 9.5. Internal architecture of an Order Holon.

For the Material Handling Holons, the Coordination Module manages all the incoming and outgoing messages with the aid of the Communication Interface, and attempts to improve the solutions delivered by using the algorithms embedded in the Cooperation Mechanisms sub-module. From the Coordination Module, the requests for service are sent to the Evaluation Module, and the information associated with them is sent to the Internal Database Module (IDM). After the request evaluation, offers are sent back through the Coordination Module to appropriate addresses. By exchanging information among the holons, and with the use of specific cooperation mechanisms, a sort of emerging coordination among the Material Handling Holons is obtained. In the case of Order Holons, the Coordination Module includes only the Communication Interface sub-module and is reduced only to exchanging messages process presented above. To generate the System Level Schedule, the Coordination Module of the Global Scheduler, includes the Schedule Generation Mechanisms sub-module, which is responsible for accounting for the overall system objective in the System Level Schedule generated.

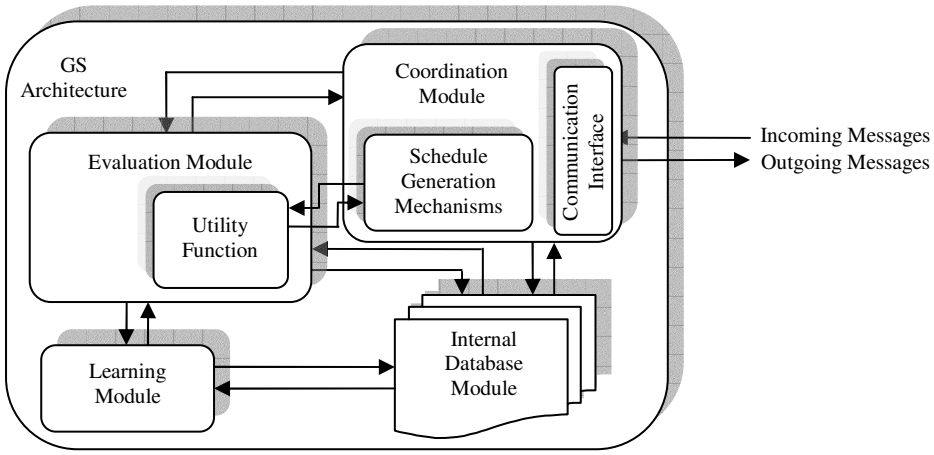

Figure 9.6. Internal architecture of the Global Scheduler.

The Communication Interface uses a common communication language and agreed communication protocols with other entities in the architecture. The software capabilities within the Coordination Modules provides the possibility to arrange messages in queues, in the First-In-First-Out (FIFO) order, such that no messages are lost and there is an arranged order of incoming and outgoing messages. In the proposed system communication is simulated using a kind of

blackboard system where individual entities can read and write messages. C++ language is chosen for both internal algorithms and the coding of the inter-holon communication language. A Communication Interface is also included in the System Monitoring and Database module, and helps in the exchange of information with other entities in the architecture.

9.4.3. The Internal Database Module

Order Holons keep in their internal databases for the duration of their existence all the information related to their associated job. The Internal Database Module in the structure of Material Handling Holons and Global Scheduler is used to store the general manufacturing system information and also specific information related with the incoming transport requests. At the completion of a particular job, the information related with the problem solving is also copied in the database, such that it is made available for future reference. More detailed description of the information needed and stored in their databases by both Material Handling Holons and Global Scheduler is provided in the next chapter.

9.4.4. The Learning Module

Since there are no reported results of improvements made by learning mechanisms associated with MH holonic systems, this research also investigates the addition of a Learning Module (LM) to the internal structure of the Global Scheduler and the Material Handling Holons. In MAS applications, several learning techniques are already adding power to different types of algorithms. Careful selected learning mechanisms embedded in the appropriate entities' software parts could deliver the same improvements in the case of holonic systems.

The stored information related to processed jobs can be used in the event that a similar job needs the same type of processing in the future. Using this information as input for the learning algorithms, at the individual Material Handling Holon level, machine learning mechanisms may result in better and faster problem solution. The Learning Module included in the Global Scheduler architecture, may help in delivering a feasible global schedule by the imposed deadlines in more situations than usual, and thus, may also result in more reliable and faster problem solution.

9.4.5. Autonomy and Cooperation in the Internal Holonic Architecture

The Evaluation Module in the internal Material Handling Holon architecture described above gives the possibility to assess the value of performing the operations for the incoming requests and decide which of the jobs is more appropriate for each individual Material Handling Holon. In the same time, the Coordination Module assures the existence of cooperation among the holons in the architecture. Thus, the internal holon architecture presented, promotes both autonomy and cooperation of holons in the proposed holonic architecture.

10. Order Processing in the Holonic System

The entities involved in the decision-making process for an incoming job are the associated Order Holon, the Material Handling Holons and the Global Scheduler. In the proposed holonic system, the L/U and transport operations are executed by the Material Handling Holons after an algorithmic evaluation of the requests received from the Order Holons. The individual schedules coming from the Material Handling Holons and the Global Scheduler are evaluated by the Order Holon and the MH tasks are awarded by choosing the lowest cost schedule received. The System Monitoring and Database unit also provides useful information in the initial stages of the decision-making process. The final decision is made by the Order Holon after receiving the L/U and transport offers from the entities in the system in the allocated time. When necessary, as stated before, changes in existing schedules are made only by the Material Handling Holons based on specific cooperation methodologies. After any change in the manufacturing conditions, the System Level Schedule is updated by the Global Scheduler and made available to all other entities by the System Monitoring and Database.

10.1. Information Needed in the Job Evaluation Process

To prepare the L/U and transportation proposals, both Material Handling Holons and the Global Scheduler need to have specific information about the incoming jobs and the manufacturing system that will permit them to make accurate offers. The incoming job information is received from the Order Holon together with the necessary L/U and transport operations, while the manufacturing system information is stored in their databases and updated when necessary by the System Monitoring and Database module. The Order Holon needs to have enough information about the incoming job and transmit it to the Material Handling Holons and the Global Scheduler such that the L/U and transport processes are made possible.

10.1.1. Information Needed by the Material Handling Holons

To plan and execute MH tasks a Material Handling Holon needs to have in its database

sufficient information in order to evaluate the incoming requests, make accurate offers, and be able to execute the potential awarded jobs, as shown generically in Table 10.1. The L/U and transport requests for a new part type received from an Order Holon contain detailed part type information needed by the Material Handling Holon during processing of all operations of the respective job. The new part type information includes details about physical properties, shape and dimensional characteristics, as well as important other characteristics related to the number of parts and timing issues. The physical properties such as material and weight, and the shape and dimensions of the new part type are necessary for establishing the gripping characteristics in the case of a robotics-based MH system and specific transport conditions in the case of an AGV system. The information about the number of parts and the timing issues are needed to make accurate offers for the MH operation requests received and includes batch sizes, due dates, task priority, and offer deadline.

Table 10.1. Information needed by the Material Handling Holon.

Information type	*Scope*	*Details*
Part type information	Evaluate L/U and transport requests Make accurate offers	Material, weight, shape, dimensions Batch size Release date/Due date Task priority Offer deadline
Process information	Evaluate L/U and transport requests Make accurate offers Perform awarded tasks	Sequence of operations Processing times Part orientation Loading/unloading times Pick-up/drop-off times
Transport system information	Perform awarded tasks	Number of machines and MH resources Location of pick-up/drop-off points L/U positions of machines Possible tracks between machines Current status of peer holons Communication/cooperation protocols

Along with the part type information described above, the Order Holon transmits to every Material Handling Holon, the manufacturing process information specific to each part type. This information is also needed to make accurate offers for any L/U and transport requests and includes sequence of operations, processing times on machines, part orientation on machines, L/U times, and pick-up and drop-off times. The manufacturing system information that permits the Material Handling Holons to move parts where they are needed, with the accuracy needed,

and in the period of time required include details about the architecture of the manufacturing system, current status of peer holons, and communication and cooperation methodologies and protocols.

The information needed about the manufacturing system architecture consists of the number of MH resources and processing machines existing in the system, the location of the pick-up and drop-off points, the L/U positions in space for the processing machines, and the possible routes to move from one machine to another. Knowing the current status of peer holons, will allow a Material Handling Holon to offer help when other peer Material Handling Holons have a heavy workload, thus decreasing the flow time of parts through the system. The communication protocols allow Material Handling Holons to exchange messages and data within the holonic architecture, while the cooperation protocols allow a Material Handling Holon to exchange information with other peer Material Handling Holons for the purpose of improving current solutions.

10.1.2. Information Needed by the Global Scheduler

The Global Scheduler needs to receive from the Order Holon the required information to be able to build an optimal schedule, if possible. Except for the information regarding the status of peer holons, and the communication and cooperation protocols, the Global Scheduler needs the same part type, process and transport system information similar to that delivered to the Material Handling Holons. Moreover, the Global Scheduler needs to know how many MH resources are available for performing the L/U and transportation tasks (i.e., if there is one of them down for any reason). This information is received from the System Monitoring and Database after any change that occurs in the system.

10.2. Job Processing in the Holonic System

Based on the well-known CNP which was presented in a previous chapter, the task allocation process considers information received from all the entities in the system, the Order and Material Handling Holons, the Global Scheduler, and the System Monitoring and Database module. However, not all these entities have the right to make decisions. Except for the System

Monitoring and Database module, all other entities have internal evaluation algorithms embedded in their structure, based on which they make the job allocation decisions corresponding with their functions in the holonic architecture.

10.2.1. Differences from the Original Version of Contract Net Protocol

Compared to the original version of the CNP used in pure heterarchical systems where the decision is totally distributed among the entities forming the structure of the system and each node can take both the manager and the contractor roles, often simultaneously, in the proposed system, the entities in the architecture have predefined responsibilities consistent with their functions described in the previous chapter. Each individual task allocation process is considered a step towards achieving the overall system goal. For each new job, there are dedicated holons having specific responsibilities, including the right to make the final decision about which holon will execute the new operations. This decision is made by the manager of the new job, the associated Order Holon, based on the proposals received from the Material Handling Holons and the Global Scheduler.

The Material Handling Holons offers have their own decision embedded in them, since they are weighed corresponding with their status and future workload. They can be weighed differently, such that placing a high weight on a specific operation is almost equivalent to rejecting it. Thus, the Material Handling Holons participate actively in the decision-making process. Based on internal algorithms, the Order Holons are evaluating all offers received, and then award the task to the MH resource that offered the lowest cost. In this way the proposed system works as a heterarchical system, with the exception that there exists a dedicated manager for each set of operations. All communications between the Order and Material Handling Holons are performed based on specific, predefined methodologies and protocols.

The job allocation process in the proposed holonic architecture allows all the Material Handling Holons to send offers for the coming jobs, no matter if they are idle or not and, thus, avoid the unwanted situation when there is no offer for a new job because there is no idle MH resource. Moreover, a Material Handling Holon can submit multiple offers for the same job, priced differently. These characteristics are in contrast with the original CNP version where only the idle nodes have the right to submit only one offer for contracting a new job. Delaying the

awards past agreed deadline, and even the possibility to abandon the execution of awarded jobs are other characteristics introduced in the Material Handling Holons' internal structure.

10.2.2. Order Holon Transport Job Announcement

Following the arrows in Figure 10.1, the task allocation process for a new order is presented in detail below. When a new job comes into the system, the first entity to acknowledge it is the System Monitoring and Database which creates a corresponding Order Holon. Once created, the Order Holon announces the new L/U and transportation tasks to the Material Handling Holons and the Global Scheduler, sends all the information related to the new part type, and sets a deadline for receiving the L/U and transport offers from the Material Handling Holons. The same deadline bounds the timeframe in which the Global Scheduler attempts to deliver a global schedule.

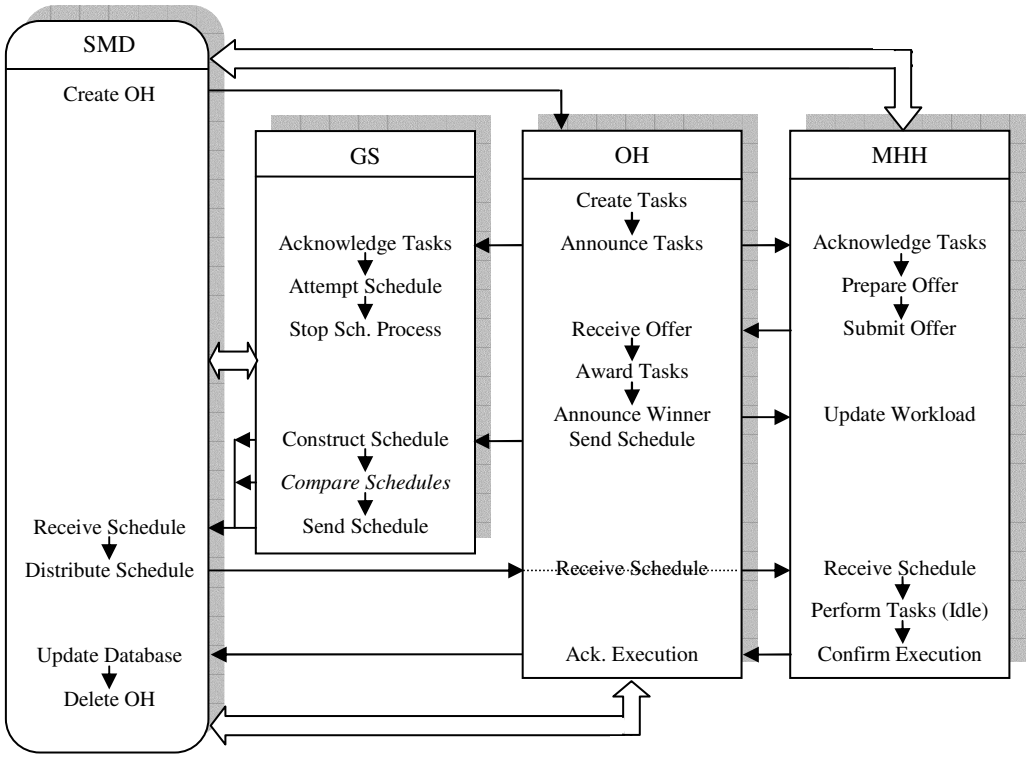

Figure 10.1. Job evaluation, allocation and execution processes in the holonic system (* the entry in *italics* not necessary to be executed).

In most cases, because of the complexity of the combinatorial problems involved, the Global Scheduler will not be able to deliver an optimal schedule by the stated deadline, so the Order Holon will receive only the offers coming from Material Handling Holons. They can construct their L/U and transport offers in time because of the much reduced complexity in the combinatorial problems needed to be solved, compared to that of the Global Scheduler.

10.2.3. Material Handling Holons Offer Preparation and Submission

After receiving the L/U and transport request, the Material Handling Holons generate their offers based on their present and future workload, and send them back to the Order Holons. Besides the present and future workload, another factor that can influence the individual transport offers is the Material Handling Holons objective, if any. Generally, Material Handling Holons can have different objectives based on which they construct their L/U and transport offers. Performing the L/U operations and moving the parts through the system in the shortest possible amount of time, is usually the most widely used, but other objectives could be used as well, such as performing all the MH operations needed for certain types of parts having particular requirements more often than others, or transferring first the jobs that require the least processing time.

10.2.4. Awarding Jobs and Schedule Generation

The decision on which MH resource will perform the operations is made after the deadline to receive the L/U and transport offers has passed, and is based on evaluating the individual offers received and choosing the best of them for each specific operation. Then, the Order Holon announces the winning resource and informs all other Material Handling Holons the winner. In the same time, the Order Holon sends the individual Material Handling Holon schedule to the Global Scheduler for constructing or updating the System Level Schedule. In the case that during the construction of the System Level Schedule there is a conflict situation in which a Material Handling Holon is supposed to perform two or more operations in the same time, the resulting System Level Schedule will delay one or more operations based on a pre-defined allocation algorithm.

Once constructed, this schedule is sent to the System Monitoring and Database module for distribution to other entities in the holonic architecture. Material Handling Holons need to receive the System Level Schedule to find out the exact times for performing the L/U and transport operations. For a Material Handling Holon, the starting times of its operations are different in the System Level Schedule compared to its initial individual schedule because of the delays introduced in the schedule by the operations performed by the other Material Handling Holons. The System Level Schedule is needed by the Order Holons to know when to check with the contracted Material Handling Holons for the completion of the transport jobs. If the Global Scheduler succeeds in its attempt to deliver a global schedule by the imposed deadline, it compares its schedule with the constructed System Level Schedule. If Global Scheduler's schedule is better in terms of the overall system objective, it becomes the new System Level Schedule.

10.2.5. Job Execution and Updating the Database

The selected Material Handling Holon will have to perform the L/U and transportation operations associated with the award, at the times specified in the System Level Schedule, while others, not selected, will finish their existing tasks or stay idle if no other job is in progress. Changes in the System Level Schedule can be performed by using the cooperation mechanisms and protocols embedded in the internal Coordination Module of the holonic structure. If a Material Handling Holon has a series of two or more operations to be performed on two different idle machines, it can ask help from the other Material Handling Holons in the architecture to perform the operation(s) on one of the two machines. Thus, the performance of the resultant System Level Schedule is improving. When an agreement is reached, the contractor Material Handling Holon is responsible for notifying the Global Scheduler for the changes made. The System Monitoring and Database module will, then, distribute the updated System Level Schedule.

After completing the tasks, the selected Material Handling Holon is confirming the task execution by sending a message to the Order Holon. The Order Holon acknowledges the task execution and sends all the information to the System Monitoring and Database for storing for future reference. During the lifetime of an Order Holon, if the Global Scheduler succeeds in

finding an optimal schedule, it sends it to the Order Holon, and using this approach, it can be available for future reference if it is stored in the system database. This optimal schedule can be useful if an order for the same part type occurs in the future and the system can operate without any disturbances. After the database is updated with the information about the recently finished job, the System Monitoring and Database will delete the Order Holon associated with this job. Under normal operation execution the holonic system has the possibility to work following a predefined schedule like a hierarchical system, while in the presence of disturbances, the autonomy of the holons permits them to work under a decentralize heterarchical architecture and therefore handle the unexpected changes much more smoothly.

10.3. Communication in the Holonic Architecture

To make the job allocation process possible, entities in the holonic architecture need to exchange information, in the form of messages, using a common communication language and agreed communication protocols. To be understood every message sent or received includes five different categories of information, such that the recipients know how to interpret it and what actions to take. The use of specific identification fields assures that the messages are sent only to the intended recipients.

The first and second fields in every message correspond to the Recipients (RE) and Sender (SE) of the messages. The third one, called Identifier (ID) is a two character code which defines the type of the message, while the last two fields contain information about the Job Characteristics (JC) and Scheduling (evaluation and allocation) Process Characteristics (SC), respectively. The use of communication protocols provides the mechanisms for exchanging information within the holonic architecture, and assures that the messages are sent only to the intended recipients. Five categories of messages: initial job allocation, sub-contracting operations, resulting System Level Schedule, job execution, and changing manufacturing conditions related messages are flowing within the holonic architecture.

10.3.1. Job Allocation Related Messages

The job allocation related messages are sent for the purpose of initial assignment of jobs

to the Material Handling Holons. They include the announcement of MH jobs and subsequent offer submission, and the winner announcement and rejection of all other offers. These job allocation related messages are presented in Table 10.2, below:

Table 10.2. Job allocation related messages.

Message Type	*RE*	*SE*	*ID*	*JC*	*SC*
Announce Job	MHH_j GS	OH_i	A1	Job_i Release Date Weight Due Date Sequence of Op. Processing Times Other Info	OH-Exp. Compl. OH-Price Offer Deadline
Submit Offer	OH_i	MHH_j	A2	Job_i	MHH-Exp. Compl. MHH-Price
Announce Winner	MHH_j	OH_i	A3	Job_i	
Reject Offer	MHH_j	OH_i	A4	Job_i	

where, $i = 1, \ldots, m$ is the number of jobs in the system; $j = 1, \ldots, n$ is the number of MH resources in the system; Other Info includes the part-type information from Table 10.1 not specifically included in the messages' entries; and, OH Expected Completion Time, OH Price, MHH Expected Completion Time and MHH Price entries will be discussed in the next sections.

10.3.2. Sub-Contracting Operations Related Messages

The sub-contracting related messages come into play when one of the Material Handling Holons asks for help in performing one or more operations from an already assigned job. These messages, presented in Table 10.3, include the help request and response to the request, winner announcement, and operation(s) completion information exchange.

10.3.3. System Level Schedule Related Messages

Once developed, the individual schedules of Material Handling Holons are combined together by the Global Scheduler to obtain the System Level Schedule. This process is performed after any change that occurred in the system and any subsequent update of the System Level Schedule occurred after an operation sub-contracting process. The information exchange

associated with the development and distribution of the System Level Schedule is presented in Table 10.4.

Table 10.3. Sub-contracting operations related messages.

Message Type	*RE*	*SE*	*ID*	*JC*	*SC*
Help Requests	MHH_j	MHH_j	H1	Job_i Op_k Weight Op_k Due Date Sequence of Op. Processing Time	MHH Price Offer Deadline
Help Offer	MHH_j	MHH_j	H2	Job_i Op_k	MHH Price
Announce Winner	MHH_j	MHH_j	H3	Job_i Op_k	
Finish Sub-contracted Operations	MHH_j	MHH_j	H4	Job_i Op_k	
Ask Completion Status	MHH_j	MHH_j	H5	Job_i Op_k	

where, $i = 1, \ldots, m$ is the number of jobs in the system; $j = 1, \ldots, n$ is the number of MH resources in the system; k is the operation to be sub-contracted; and, MHH Price are the amounts offered and requested by the sub-contracting and sub-contractor entities respectively, for performing the specified operations.

Table 10.4. System Level Schedule related messages.

Message Type	*RE*	*SE*	*ID*	*JC*	*SC*
Individual Schedule	GS	OH_i	S1	Job_i	Ind. Schedule
Resulting System Level Schedule	SMD	GS	S2		SLS
Distribute System Level Schedule	OH_i MHH_j	SMD	S3		SLS
Change in System Level Schedule	GS	MHH_j	S4	Job_i Op_k	Completion Time

where, $i = 1, \ldots, m$ is the number of jobs in the system; $j = 1, \ldots, n$ is the number of MH resources in the system; and, k is the operation to be sub-contracted.

10.3.4. Job Execution Related Messages

These messages are exchanged when a Material Handling Holon finishes the execution of

an awarded job. They include the Order Holon job manager inquiry about job completion, Material Handling Holon acknowledgement of job execution, and the contractor and sub-contractor(s) account updating, and are presented in Table 10.5.

Table 10.5. Job execution related messages.

Message Type	*RE*	*SE*	*ID*	*JC*	*SC*
Finish Job	OH_i	MHH_j	E1	Job_i	
Ask Completion Status	MHH_j	OH_i	E2	Job_i	
Update Contractor Account	SMD	OH_i	E3		MHH_j Account Amount
Update Sub-Contractor Account	SMD	MHH_j	E4		MHH_j Account Amount

where, $i = 1, \ldots, m$ is the number of jobs in the system; $j = 1, \ldots, n$ is the number of MH resources in the system; and, MHH_j account amount is the value with which the Material Handling Holons accounts are updated at the completion of the job, and will be discussed in the next sections.

10.3.5. Changing Manufacturing Conditions Related Messages

The communications associated with these types of disturbances are presented in Table 10.6 below:

Table 10.6. Changing manufacturing conditions related messages.

Message Type	*RE*	*SE*	*ID*	*JC*	*SC*
New Order Holon	*GS* MHH_j	*SMD*	C1		
Add Resource	OH_i MHH_j GS	SMD	C2		MHH_j
Remove Resource	OH_i MHH_j GS	SMD	C3	Job_i	MHH_j
Resource Breakdown	SMD	MHH_j	C4		
Recovering from Breakdown	SMD	MHH_j	C5		
Changing Due Dates	MHH_j	OH_i	C6	Job_i New Due Date	OH Price
Answer to New Due Date Request	OH_i	MHH_j	C7	Job_i	MHH Exp. Compl. MHH Price

where, $i = 1, \ldots, m$ is the number of jobs in the system; $j = 1, \ldots, n$ is the number of MH resources in the system; and, MHH Expected Completion Time and MHH Price entries will be discussed in the next sections.

New jobs entering the system, adding and removing MH resources, changing due dates for a job already in processing and resource breakdowns are of most importance because of the real-time response, fault-tolerance and hardware reconfigurability characteristics required for the next generation manufacturing systems.

10.3.6. Message Exchange during Operations

Figure 10.2 presents a short instance, in the form of a timeline, of the holonic system operation, that illustrates most of the message exchanges within the holonic architecture. A job *(j-1)* already in process and assigned to MHH_2, and the two jobs *(j* and *j+1)* entering the system in the period *[i, i+29]* have the corresponding Order Holons, OH_{j-1}, OH_j and OH_{j+1}, created by the System Monitoring and Database module. Based on the message exchange process between the Order and Material Handling Holons, the two new jobs are allocated to MHH_3 and MHH_1, respectively.

There are several distributions of the System Level Schedule corresponding to each change in the system. The process of Inter-Material Handling Holons cooperation is illustrated by the help request of MHH_3, the k^{th} operation of job *j*, being awarded to MHH_2. A resource breakdown announcement is illustrated, followed by the announcement of removing one resource (MHH_2) from the architecture. The operations of job *(j-1)* left unfinished by MHH_2 are re-announced to the remaining resources by the corresponding OH_{j-1}. In the same time, MHH_2 finishes all the operations for job *(j-1)* and confirms job completion to the corresponding manager, OH_{j-1}, which then updates the database. The other two operations are then completed in time, the database is updated, and the system waits for new orders.

As it can be seen from Figure 10.2, the amount of communication in the holonic architecture can be considerable. Therefore communication load is an important issue in the overall holonic system efficiency. Karageorgos *et al.* (2003) consider that, even simple applications could generate a substantial amount of communication, which "imposes an unacceptable overhead and slows down the application." The potential problems that could

appear due to an increased volume of messages when many unexpected changes occur in a short period of time are to be investigated during the software development and testing stages.

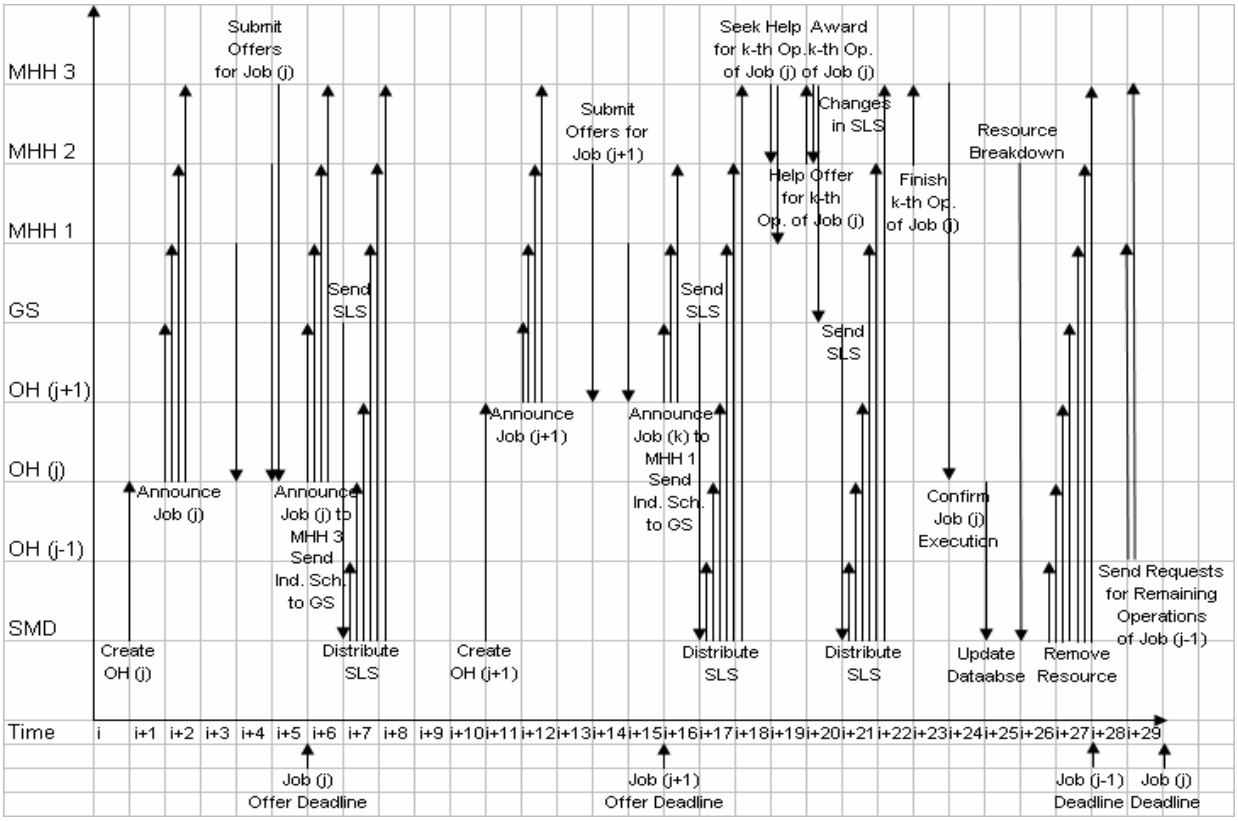

Figure 10.2. Timeline of the operation of the holonic system.

10.4. Job Evaluation, Allocation, and Execution Algorithms

All the algorithms developed for the MH holonic job evaluation, allocation and execution processes, and for the computation needs of the Global Scheduler take in consideration the precedence constraints of the jobs on each machine, the precedence constraints of the operations of each job, and the MH resource constraint, that states that, at each particular time, there should not be scheduled more MH operations than the number of available MH resources. Due to the distributed decision-making existing in the holonic system, the algorithms used for MH job evaluation and allocation are simple and not characterized by combinatorial explosiveness.

Each MH resource allocation process has as output an Individual Schedule (IS) which is sent to the Global Scheduler by the active Order Holons in the system. Using its system perspective, the Global Scheduler combines all the Individual Schedules received into a System Level Schedule, as presented before, which is then distributed to all the entities in the architecture. No complex algorithm is required for obtaining this emergent schedule. From the algorithmic point of view, a common personal computer (PC) is powerful enough to handle the needs for any entity in the proposed holonic system. For scheduling individual Material Handling Holons operations, these algorithms only require an arrangement of jobs along one dimension based on specific steps needed to be executed.

10.4.1. Order and Material Handling Holons Job Evaluation and Allocation Procedure

Each Material Handling Holon in the architecture has an associated account which is updated at the completion time of every job in which the associate MH resource participated. The Material Handling Holon objective of moving as many jobs as possible through the system in the shortest amount of time is translated in winning as many job awards as possible such that the amount in its associated account increases. Regardless of the overall system objective, since Order Holons are responsible only for their associated job, their objective is to finish the job as soon as possible.

The Order Holons evaluation procedure, calculates the *Order Holon - Expected Completion Time* (*OH-ECT*) and *Order Holon - Offered Price* (*OH-OP*), and sets a deadline for receiving transport offers. The *OH-ECT* is obtained by adding the expected MH operations time to the completion time for each job given by the processing jobs schedule:

$$OH - ECT = C_j + \sum_{k=1}^{K_j} \left(\left(LT_k \right)_j + \left(UT_k \right)_j \right) \tag{10.1}$$

where, C_j is the completion time of job j; k is the index for the operations of job j; K_j is the number of operations of job j; and $(LT_k)_j$, $(UT_k)_j$ are the times for the L/U operations of job j.

The *OH-OP* is a function of the weight of the job, the expected tightness of the job due date, and the number of necessary L/U operations. The expected tightness of the job due date is expressed by:

$$\tau = \frac{OH - ECT}{d_j} \tag{10.2}$$

where, *OH-ECT* is calculated using equation (10.1); d_j is the due date of the job *j*. For values of $\tau < 1$, the job is expected to complete before the due date, while for values of $\tau > 1$, the job will be tardy.

Then, *OH-OP* is calculated as follows:

$$OH - OP = w_j \tau \sum_{k=1}^{K_j} \left((L_k)_j + (U_k)_j \right) \tag{10.3}$$

where, w_j is the weight of job *j*; τ is the expected tightness of the job due date calculated using equation (10.2); $(L_k)_j$, $(U_k)_j$ are the L/U operations for operation *k* of job *j*; and, K_j is the number of operations of job *j*.

The initial deadline, d_i, for receiving transport offers set by an Order Holon depends on the number of operations to be performed for a particular job. If there are no offers received by the stated deadline, the Order Holon is relaxing the offer deadline constraint by using the following equation:

$$d_{c+1} = d_c + r \tag{10.4}$$

where, d_c is the offer deadline constraint for the c^{th} relaxation; *c* is the offer deadline index; and, *r* is the relaxation step value. Initially *c* is equal to zero.

The final offer deadline constraint is expressed by the following equation:

$$d_f = d_i + r * c \tag{10.5}$$

where, d_i is the initial offer deadline constraint; d_f is the final offer deadline constraint; and, *c*, defined previously, counts the number of relaxations.

After receiving a service request from an Order Holon, the Material Handling Holons use their internal evaluation procedure to assess the utility of executing the new jobs. Each Material Handling Holon that can execute the new L/U and transport operations by the stated due date, sends back its offer to the Order Holon containing a revised completion time (*Material Handling Holon - Expected Completion Time* (*MHH-ECT*)) and offer price (*Material Handling Holon - Offer Price* (*MHH-OP*)). These terms are calculated considering the *OH-OP* and *OH-ECT* presented above and the impact of this new potential awarded job on the already assigned jobs.

Two possible situations could appear depending on the location of the operations of the new job, *j*. In the first one, presented in Figure 10.3, the new job *j* is scheduled before an old job

i. The L/U operations of job *j* are taken into consideration when calculating the *OH-ECT* and *OH-OP*.

But, if the first case in Figure 10.3 occurs when job *j* is scheduled, the old job's L/U operations will be delayed. In the second case of Figure 10.3, depending on the value of the following expression, the old job's L/U operations will be delayed or not by the new job *j*:

$$(S_i - C_j) - (LT_j + UT_j) \tag{10.6}$$

where, S_i is the starting time of job *i*; C_j is the completion time of job *j*; and, LT_j, UT_j are the L/U times for job *j*.

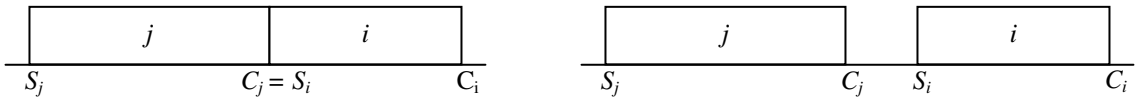

Figure 10.3. Influence of a new job, *j,* on the L/U times of an old job, *i*.

Without having a global view, a Material Handling Holon will interpret a negative value for expression (10.6) as a delay for the L/U operation times of its old job, while a zero or positive value will be seen as not introducing delays in its old job.

The second situation has the new job *j* scheduled after an old job *i* as presented in Figure 10.4. In this situation, the expected L/U operations are considered in the calculation of *OH-ECT* and *OH-OP*, but, if the first case in Figure 10.4 occurs, because of the L/U operations of the old job *i*, there will be a delay in the operations for the new job *j*.

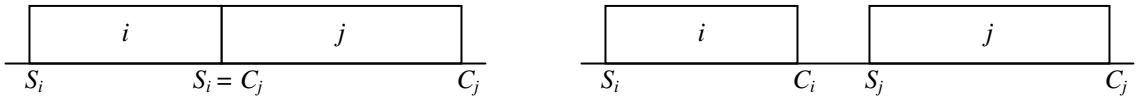

Figure 10.4. Influence of an old job, *i*, on the L/U times of a new job, *j*.

Similarly like above, depending on the following expression, there could be a more than expected delay for the L/U operations of the new job *j*:

$$(S_j - C_i) - (LT_i + UT_i) \tag{10.7}$$

where, S_j is the starting time of job *j*; C_i is the completion time of job *i*; and, LT_i, UT_i are the L/U

times for job *i*.

Examining only its jobs received, a Material Handling Holon will consider that a negative value for expression (10.7) is equivalent to introducing a delay in the L/U times of the new potential awarded job, while a zero or negative value of (10.7) is not coming with more delays. Using the above considerations, the *MHH-ECT* and *MHH-OP* are calculated as follows:

$$MHH-ECT = OH-ECT + \sum_{m=1}^{M}\sum_{i=1}^{K}\left[\left(\left(LT_i\right)_m + \left(UT_i\right)_m\right) - \left(\left(S_j\right)_m - \left(C_i\right)_m\right)\right]A_i + \sum_{m=1}^{M}\sum_{i=1}^{K}\left[\left(\left(LT_j\right)_m + \left(UT_j\right)_m\right) - \left(\left(S_i\right)_m - \left(C_j\right)_m\right)\right]A_j \quad (10.8)$$

where, *OH-ECT* is calculated using equation (10.1); $(LT_i)_m$, $(UT_i)_m$ are the L/U times of job *i* on machine *m*; $(LT_j)_m$, $(UT_j)_m$ are the times for the L/U operations of job *j* on machine *m*; *K* represents the number of jobs already assigned to that Material Handling Holon; *M* represents the number of machines; $(S_i)_m$ is the starting time of job *i* on machine *m*; $(S_j)_m$ is the starting time of job *j* on machine *m*; $(C_i)_m$ is the completion time of job *i* on machine *m*; $(C_j)_m$ is the completion time of job *j* on machine *m*;

$$A_i = \begin{cases} 1\ , & if\ \left(\left(S_j\right)_m - \left(C_i\right)_m\right) - \left(\left(LT_i\right)_m + \left(UT_i\right)_m\right) < 0 \\ 0\ , & otherwise \end{cases} \quad (10.9)$$

$$A_j = \begin{cases} 1\ , & if\ \left(\left(S_i\right)_m - \left(C_j\right)_m\right) - \left(\left(LT_j\right)_m + \left(UT_j\right)_m\right) < 0 \\ 0\ , & otherwise \end{cases} \quad (10.10)$$

and,

$$MHH-OP = OH-OP + w_i\sum_{m=1}^{M}\sum_{k=1}^{K}\left[\left(\left(LT_i\right)_m + \left(UT_i\right)_m\right) - \left(\left(S_j\right)_m - \left(C_i\right)_m\right)\right]A_i + w_j\sum_{m=1}^{M}\sum_{k=1}^{K}\left[\left(\left(LT_j\right)_m + \left(UT_j\right)_m\right) - \left(\left(S_i\right)_m - \left(C_j\right)_m\right)\right]A_j \quad (10.11)$$

where, *OH-OP* is calculated using equation (10.3); w_i is the weight of job *i*; and, w_j is the weight of job *j*.

10.4.2. Order Holon Evaluation and Allocation Algorithm

Using the above equations and the job processing procedure presented before, the Order Holon evaluation and allocation algorithm can be expressed in the following steps. The Order

Holon evaluation and allocation algorithm is also presented graphically in the form of a flowchart in Figure 10.5 below, and in the pseudocode form in Appendix A.1.

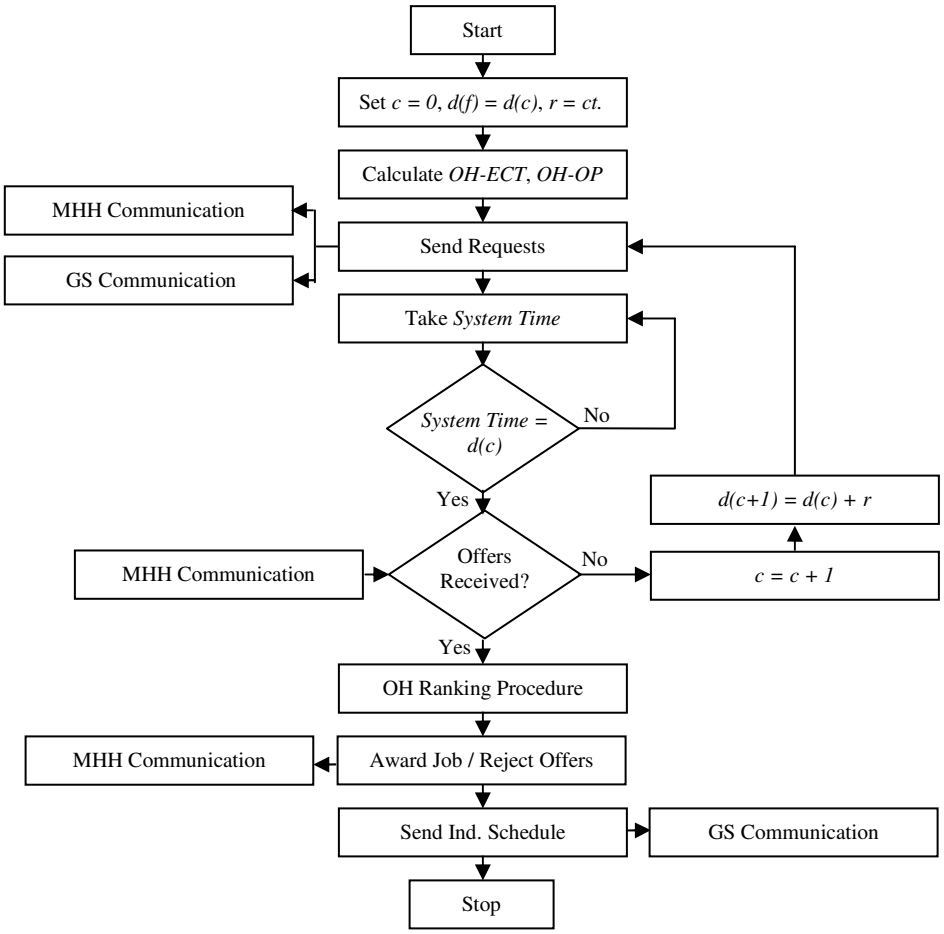

Figure 10.5. Order Holon evaluation and allocation algorithm.

1. Start
2. Set $c = 0$, $r = 0$, and the offer deadline constraint, $d_f = d_c$
3. Calculate *OH-ECT* and *OH-OP* values
4. Send transport requests to MHHs and the GS
5. Take *System Time*, the actual time of the OH controller internal clock
6. If *System Time* is equal to the offer deadline, go to 7, else go back to 5
7. If no offers received from MHHs, relax the offer deadline constraint and go

back to 4; else go to 8

8. Use the OH ranking procedure to select a winner
9. Award job to the first MHH on list, and reject all other MHHs offers
10. Send the IS of the winning MHH to the GS
11. Stop

10.4.2.1. Order Holon Ranking Procedure Algorithm

The Order Holon ranking procedure gives the priority method for ranking offers as well as a series of tie-breaking rules for the case in which two or more offers are ranked the same when using only the *MHH-ECT* value. This offer ranking algorithm is also presented in the form of a flowchart in Figure 10.6, below, and in pseudocode form in Appendix A.1.1.

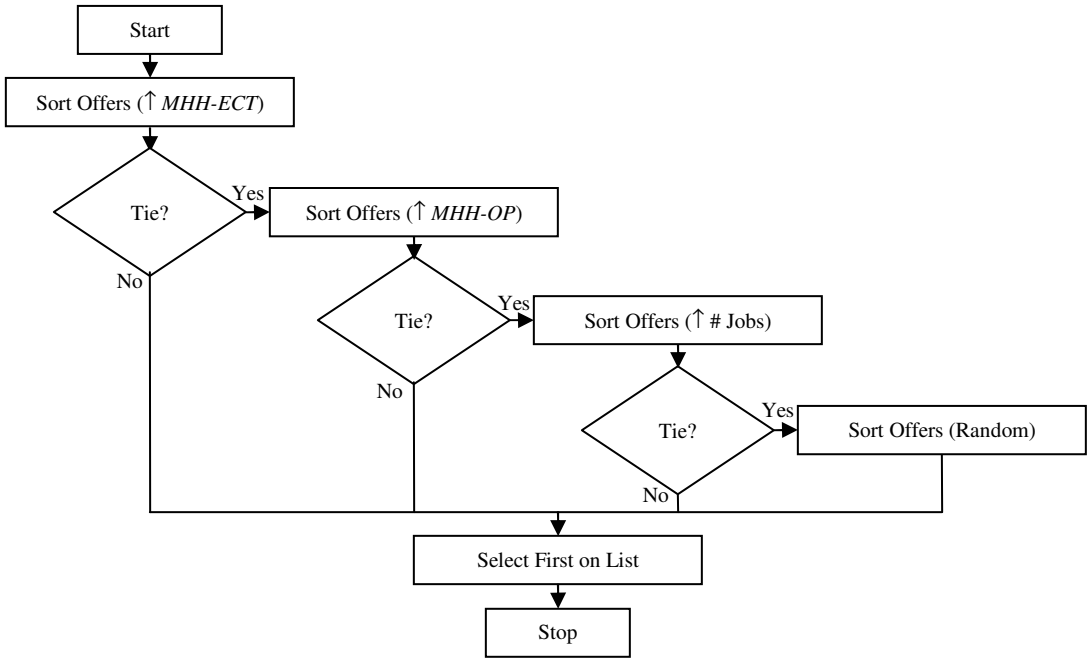

Figure 10.6. Order Holon's offers ranking procedure algorithm.

1. Start
2. Sort offers in increasing order of their *MHH-ECT* value
3. If tie, sort offers in increasing order of their *MHH-OP* value ; else go to 6

4. If tie, sort offers in increasing order of the number of already assigned jobs; else go to 6
5. If tie, sort offers randomly; else go to 6
6. Select first on list
7. Stop

10.4.3. Material Handling Holon Evaluation Algorithm

The Material Handling Holon evaluation algorithm is developed using the equations and the job processing procedure presented before. During the operation of the holonic system the Material Handling Holons are following the System Level Schedule existing at any particular time, and are answering to any requests received from the Order Holons. A flowchart form of this algorithm is presented in Figure 10.7, and the pseudocode can be found in Appendix A.2. The steps in the Material Handling Holon evaluation algorithm are as follows:

1. Start
2. Set T = constant
3. Take *System Time*
4. Set *Check Time* equal to *System Time*
5. Take *System Time*
6. If *System Time* is equal to *Check Time* plus constant T, go to 7, and go back to 4 to update the *Check Time* with the new *System Time*; else go back to 5
7. If request received from OH, go to 8; else follow the exiting SLS
8. Sort jobs using the MHH ranking procedure
9. If the list of sorted jobs is not empty, go to 10; else follow the existing SLS
10. Select and remove the first job on the list of sorted jobs
11. Calculate *MHH-ECT* and *MHH-OP* values for the selected job
12. Take *System Time*
13. If *System Time* is less than or equal to the offer deadline, go to 14; else go back to 9
14. Send offer to the corresponding OH

15. Wait until the OH awarding/rejecting process is performed, and go back to 9
16. Follow SLS

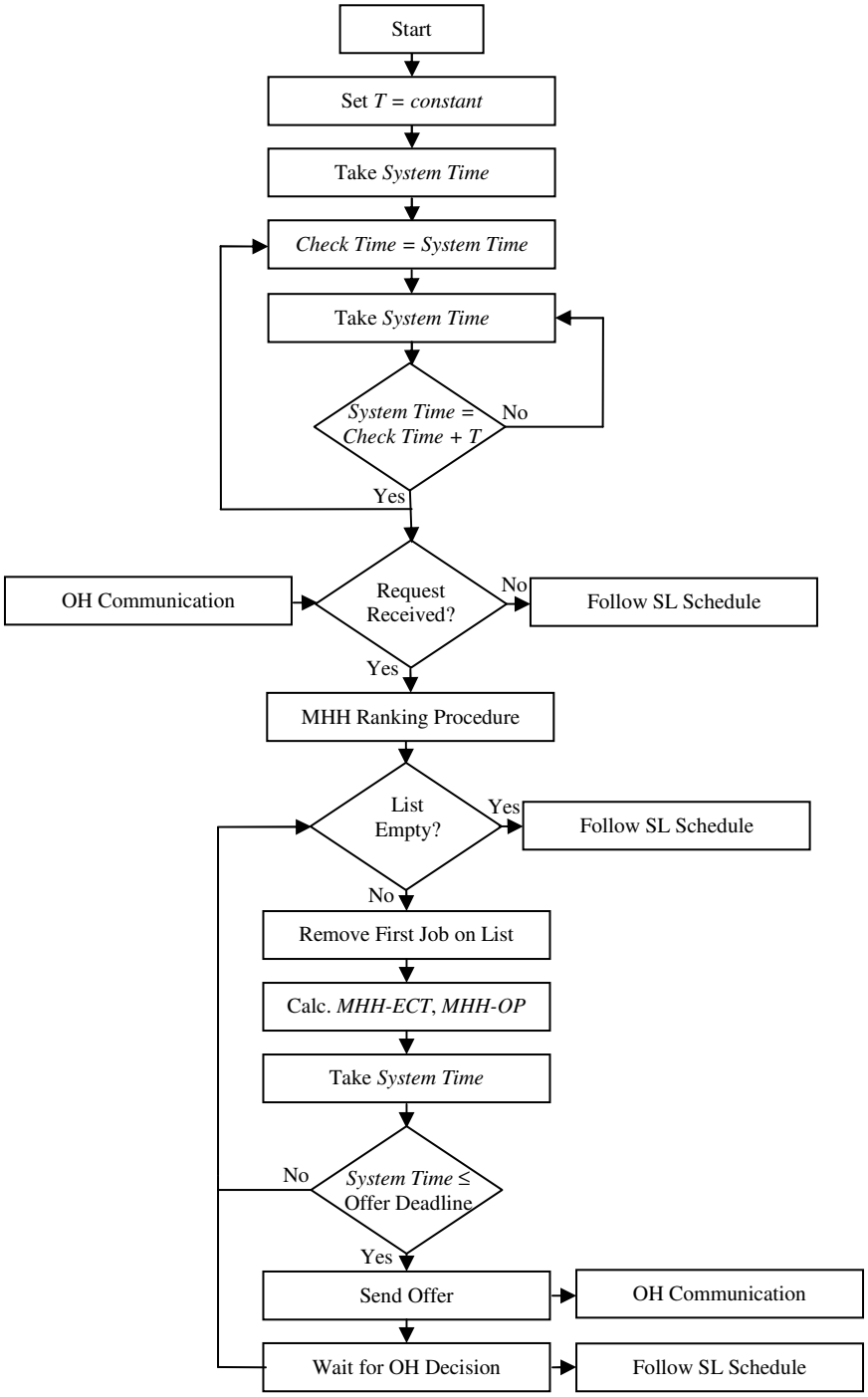

Figure 10.7. Material Handling Holon evaluation algorithm.

where, *T* is a constant which measures the time between two consecutive checks for a new L/U and transport request; *System Time* is the actual time taken from the Material Handling Holon controller internal clock; and, *Check Time* is a variable that helps in checking for new L/U and transport requests at equal *T* time intervals.

10.4.3.1. Material Handling Holon Ranking Procedure Algorithm

The Material Handling Holon ranking procedure sorts among the requests received at the same time and gives the order in which the Material Handling Holon responds to these requests. The algorithm to obtain this ordered list is presented below, as well as graphically in Figure 10.8. The pseudocode for the Material Handling Holon request ranking procedure can be found in Appendix A.2.1.

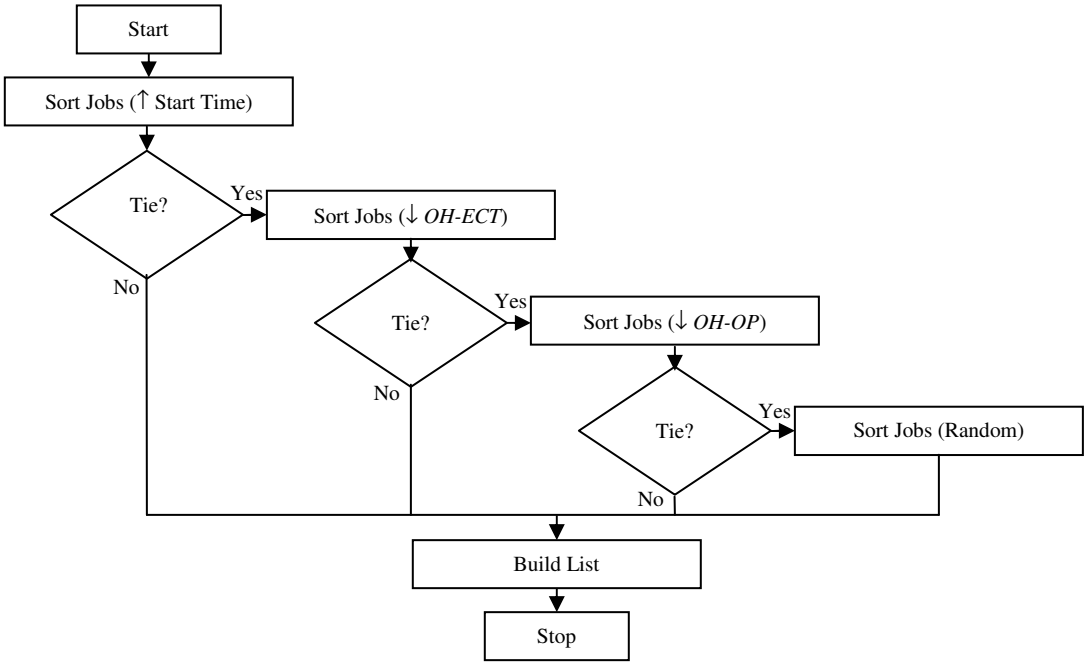

Figure 10.8. Material Handling Holon's requests ranking procedure algorithm.

1. Start
2. Sort jobs in increasing order of their starting times

3. If tie, sort jobs in decreasing order of their *OH-ECT* value; else go to 6
4. If tie, sort jobs in decreasing order of their *OH-OP* value; else go to 6
5. If tie, sort jobs randomly; else go to 6
6. Build list of sorted jobs
7. Stop

Because all Material Handling Holons in the system are fair players, there is no possibility of deceiving by offering lower *MHH-ECT* and *MHH-OP*. So, when selecting the Material Handling Holon to execute the transport operations for a particular job, the Order Holon will choose the most appropriate Material Handling Holon to complete the job.

10.4.4. Individual Schedules Generation Algorithm

The Individual Schedules for the Material Handling Holons are developed by the Order Holons using their limited view in the system. In these Individual Schedules there is no information regarding other jobs that might be present in the system. When inserting the loading operations in the individual Material Handling Holon schedule, one of the two situations, presented also in Figure 10.9, occurs:

$$STP_k - CTP_{k-1} < LT_k \tag{10.12}$$

or,

$$STP_k - CTP_{k-1} \geq LT_k \tag{10.13}$$

where, STP_k is the starting time of the k^{th} processing operation of job *j*; CTP_{k-1} is the completion time of the *(k-1)* processing operation of job *j*; and, LT_k is the loading time for the k^{th} operation of job *j*.

In the first case, the insertion of the loading time requires a movement of operation *k* to the right with the amount of time, T_1, given by the following equation:

$$T_1 = CTP_{k-1} + LT_k - STP_k \tag{10.14}$$

In the second case, the insertion of the loading time is done without modifying the processing operations schedule.

Similarly, for inserting the unloading operations, two cases appear:

$$STL_{k-1} - CTP_k < UT_k \tag{10.15}$$

or,

$$STL_{k+1} - CTP_k \geq UT_k \tag{10.16}$$

where, STL_{k+1} is the starting time of the loading of the *(k+1)* operation of job *j*; CTP_k is the completion time of the k^{th} processing operation of job *j*; and, UT_k is the unloading time for the k^{th} operation of job *j*.

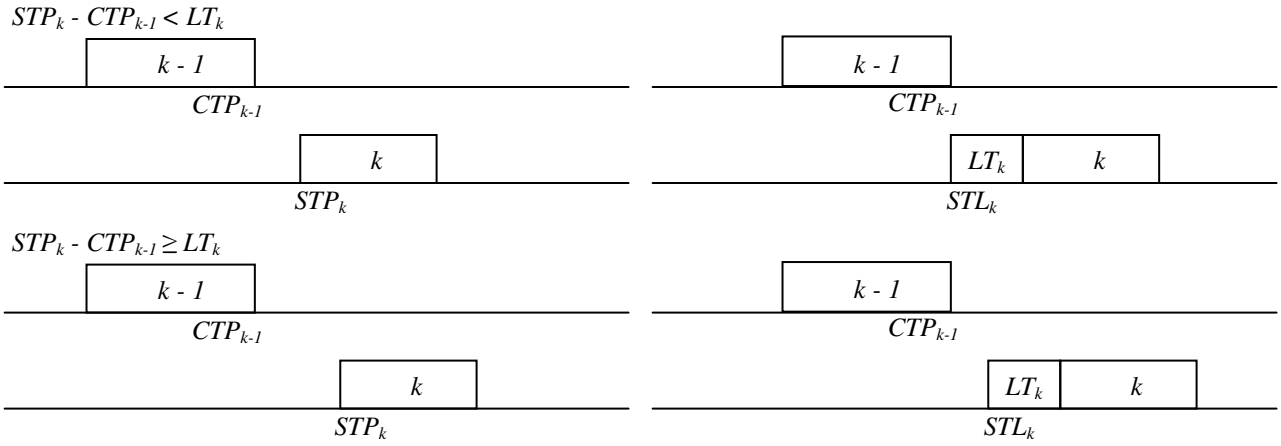

Figure 10.9. Insertion of the loading time in the Individual Schedules.

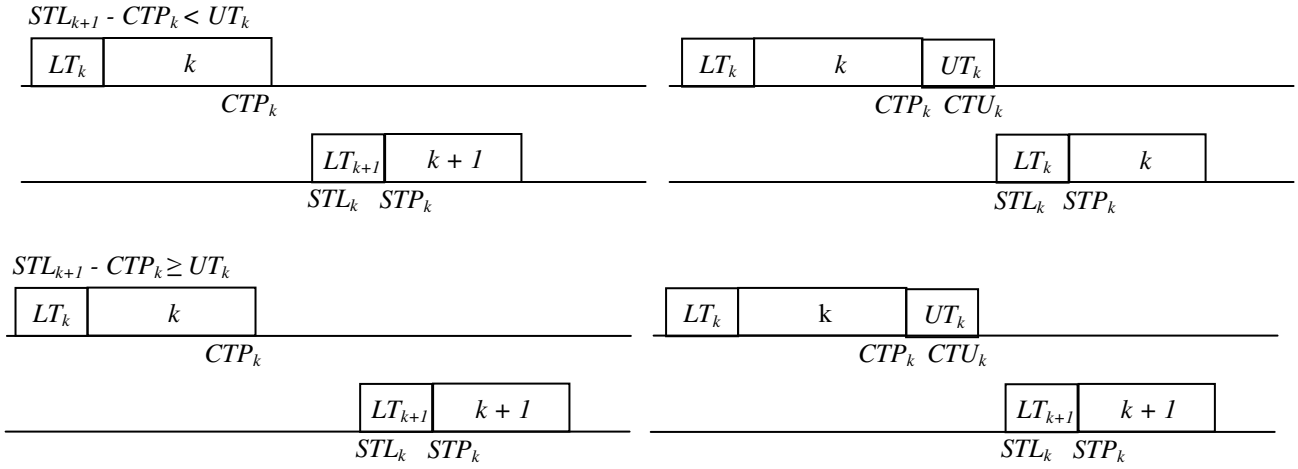

Figure 10.10. Insertion of the unloading time in the Individual Schedules.

As seen also in Figure 10.10, in the first case, the insertion of the unloading time requires a

movement of operation k to the right with the amount of time, T_2, given by the following equation:

$$T_2 = CTP_k + UT_k - STL_{k+1} \tag{10.17}$$

In the second case, the insertion of the loading time is done without modifying the processing operations schedule. The algorithm for generating the Individual Schedules is presented in Figure 10.11 and composed from the following steps:

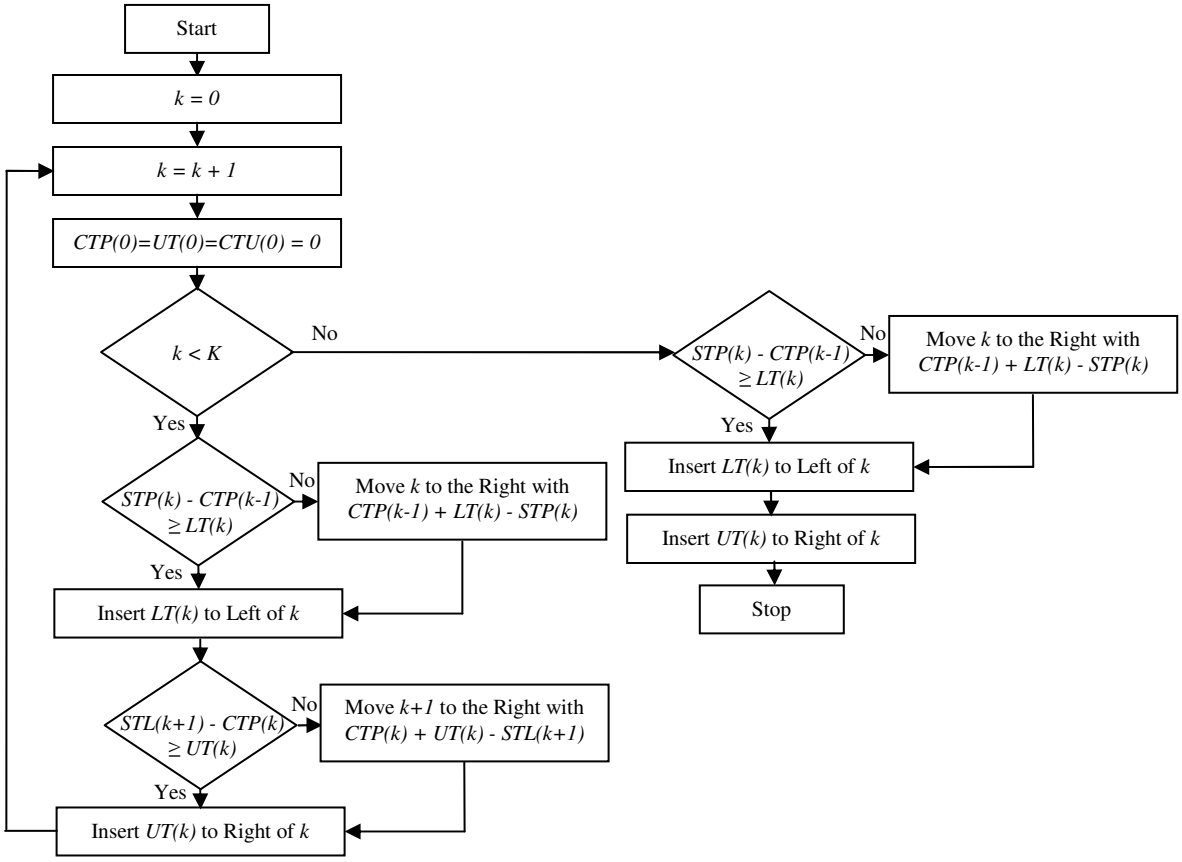

Figure 10.11. Individual Material Handling Holon schedule generation algorithm.

1. Start
2. Initialize k, the index of the operations of the job, with the value *0*
3. Increase k value by *1*
4. Set $CTP_0 = UT_0 = CTU_0 = 0$ (because operation *0* does not exist)
5. If k is less than the number of operations, K, of the job, go to 6; else go to 11

6. If inequality (10.12) is true, move operation k to the right with the time amount given by equation (10.14), and then go to 7; else go directly to 7
7. Insert LT_k to the left of k
8. If expression (10.15) is true, move operation $(k+1)$ to the right with the time amount given by equation (10.17), and then go to 9; else go directly to 9
9. Insert UT_k to the right of k
10. Go back to 3
11. If, for the last operation, inequality (10.12) is true, move operation k to the right with the time amount given by equation (10.14), and then go to 12; else go directly to 12
12. Insert LT_k to the left of k
13. Insert UT_k to the right of k
14. Stop

where, CTP_0, UT_0, CTU_0 are the completion time of the processing operation, unloading time and completion time of the unloading operation, respectively, applied for the imaginary operation zero; and, LT_k, UT_k are the L/U times of operation k. The pseudocode for this algorithm can be found in Appendix A.3.

10.4.5. The System Level Schedule Generation Algorithm

In the holonic system, under real-time constraints, a feasible System Level Schedule for the MH resources is generated by the Global Scheduler from the combination of the individual Material Handling Holons schedules. In fact, there is no final schedule at any time, the System Level Schedule is changing every time when some condition change (i.e., it is updated in real-time, and takes into account all the changes in the system and in the individual schedules). These System Level Schedules are distributed by the System Monitoring and Database module to all the entities in the system. Thus, the Material Handling Holons will have a global view of the place of their operations in the System Level Schedule, and thus, know when to perform a particular L/U or transport operation.

The Global Scheduler combines the Individual Schedules received from the Order

Holons following predefined rules and taking into account job and resource constraints. When two or more operations assigned to the same Material Handling Holon have the starting times in the same moment, the Global Scheduler should decide which operation comes first. As a general priority rule, the job in the process of unloading and then loading on another machine always comes first. The following algorithms and subsequent tie-breaking rules procedures are used by the Global Scheduler to develop the System Level Schedule and improve the job completion times. When only updating the System Level Schedule with the changes made using the Inter-Material Handling Holon cooperation mechanisms the algorithm will start from step 7. Figure 10.12 presents this algorithm in graphical form. The pseudocode form of the System Level Schedule generation algorithm is presented in Appendix A.4.

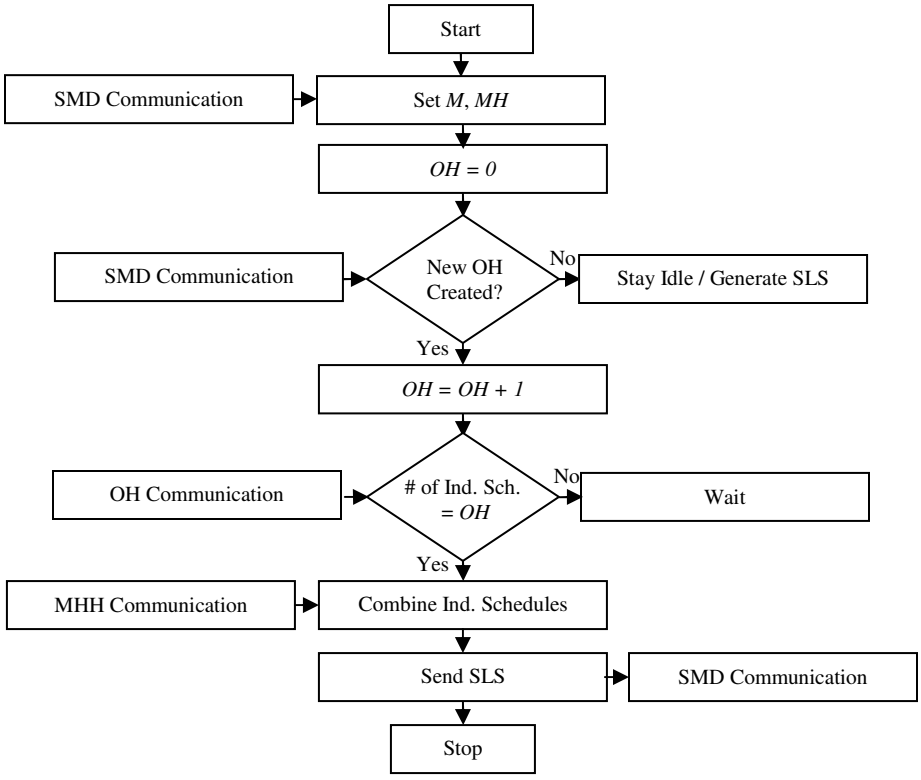

Figure 10.12. System Level Schedule generation algorithm.

1. Start
2. Set the values for the number of machines, *M*, and MH resource, *MH* from the

information received from the SMD

3. Initialize, *OH*, the number of OH in the system, with the value *0*
4. If a new OH message is received from the SMD, go to 5; otherwise continue previous activity or stay idle, if no activity
5. Increase *OH* value by *1*
6. If the number of IS received from the active OH is equal to *OH*, go to 7; else wait until the next IS is received and check again.
7. Combine the IS received previously from the OH or update the SLS with the changes received from MHHs
8. Send the resulting SLS to SMD
9. Stop

10.4.5.1. The System Level Schedule Constraint Satisfaction Algorithms

When combining the Individual Schedules received from the corresponding Order Holons (step 7 in the algorithm above) a series of three types of constraints must be satisfied by the resulting System Level Schedule. The first two types of constraints, job precedence constraints on processing machines and sequence of operations for the same job are coming from the job data. The third one takes into account the MH operations, and considers that at most one loading or unloading operation can be performed at any time by any of the MH resources in the system. The algorithm for the combination of Individual Schedules is presented below, and in a flowchart form in Figure 10.13. The psuedocode for this algorithm is given in Appendix A.5.

1. Start
2. Consider jobs precedence constraints on machines
3. Consider operations precedence constraint for the same job
4. Consider MH operations constraints
5. Stop

Three algorithms developed for satisfying each of the three types of constraints are presented next. The first one assures that the jobs precedence constraints on each machine in the

scheduling environment are satisfied. The only situation that requires changes in the schedule is when the following inequality between starting and completion times of consecutive loading and unloading operations appears:

$$STL_j - CTU_{j-1} < 0 \tag{10.18}$$

where, STL_j is the starting time of a loading operation of job *j*; and, CTU_{j-1} is the completion time of an unloading operation of job *(j-1)*.

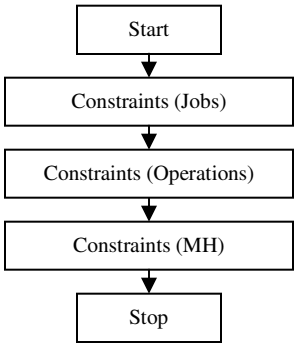

Figure 10.13. Steps in combining individual Material Handling Holons schedules.

To satisfy the jobs precedence constraints when inequality (10.18) is true, job *j* must be moved to the right with the time amount given by the following equation:

$$T_3 = CTU_{j-1} - STL_j \tag{10.19}$$

Figure 10.14 presents graphically how the jobs precedence constraints are addressed whenever inequality (10.18) is true.

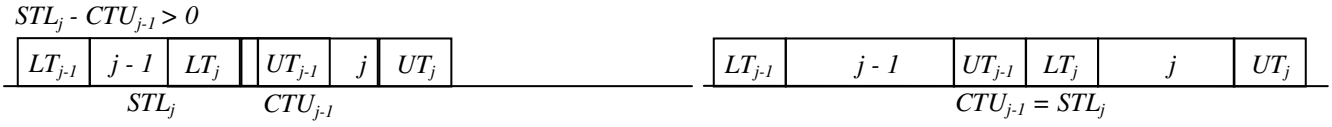

Figure 10.14. Handling job precedence constraints.

The following algorithm, presented also graphically in Figure 10.15, incorporates the above considerations and assures that all the job precedence constraints on every machine in the

system are satisfied. The pseudocode for this algorithm is presented in Appendix A.5.1.

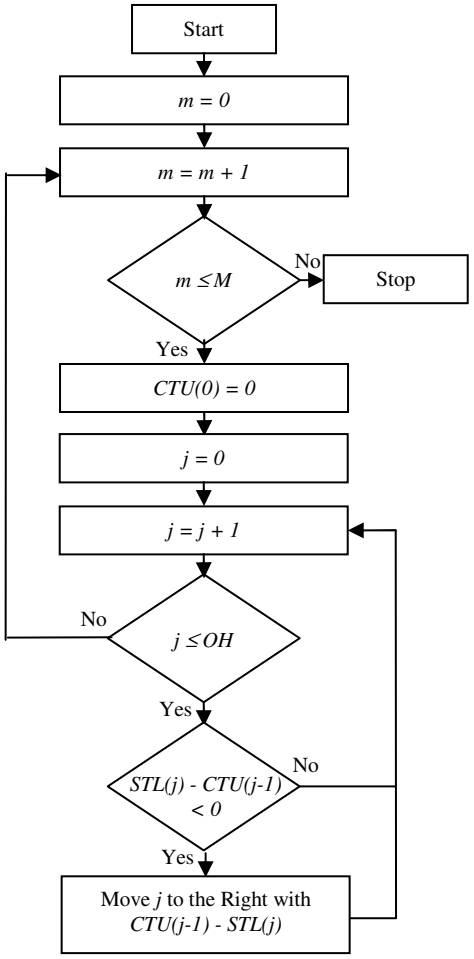

Figure 10.15. Algorithm used for satisfying the job precedence constraints.

1. Start
2. Initialize the machine index, *m*, with the value *0*
3. Increase *m* value by *1*
4. If *m* is less than or equal to the total number of machines, go to 5; else go to 10
5. Set the completion time of the unloading of imaginary operation *0* equal to *0*
6. Initialize the job index, *j*, with the value *0*
7. Increase *j* value by *1*

8. If the number of jobs is less than or equal to the total number of jobs in the system, go to 9; else go back to 3
9. If inequality (10.18) is true, move job *j* to the right with the amount given by equation (10.19), and go back to 7; else go directly back to 7
10. Stop

The second algorithm assures that the operation precedence constraints within all jobs in the system are satisfied. After combining the Individual Schedules, to satisfy the operation precedence constraints, changes in the schedule are needed when the following inequality becomes true (Figure 10.16):

$$STL_k - CTU_{k-1} < 0 \tag{10.20}$$

where, STL_k is the starting time for the loading of operation k of job j; and, CTU_{k-1} is the completion time for the unloading of operation $(k-1)$ of job j.

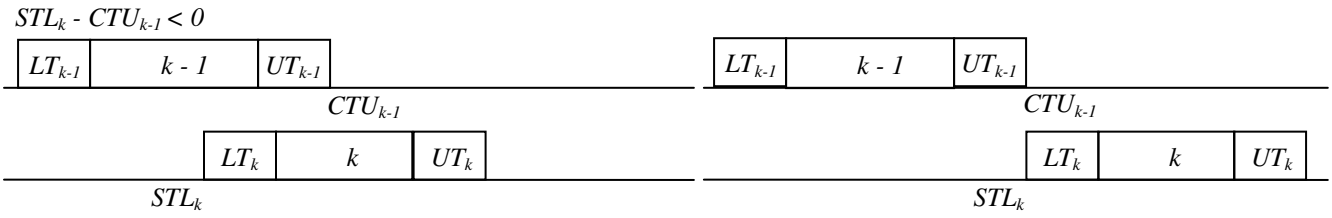

Figure 10.16. Handling operation precedence constraints.

To satisfy the operations precedence constraints when inequality (10.20) is true, operation k must be moved to the right with the time amount given by the following equation:

$$T_4 = CTU_{k-1} - STL_k \tag{10.21}$$

The steps needed to satisfy this type of constraints are presented in the algorithm below, as well as graphically in Figure 10.17. The pseudocode for the handling operation precedence constraints in the holonic system is presented in Appendix A.5.2.

1. Start
2. Initialize the job index, j, with the value 0
3. Increase j value by 1

4. If *j* is less than or equal to the total number of OH in the system, go to 5; else go to 10
5. Initialize the operation index, *k*, with the value *0*
6. Increase *k* value by *1*
7. If the number of operations is less than or equal to the total number of operations, *K*, for that particular job, go to 9; else go back to 3
8. Set the completion time of the unloading of imaginary operation 0 equal to 0
9. If inequality (10.20) is true, move operation *k* to the right with the amount given by equation (10.21), and go back to 6; else go directly back to 6
10. Stop

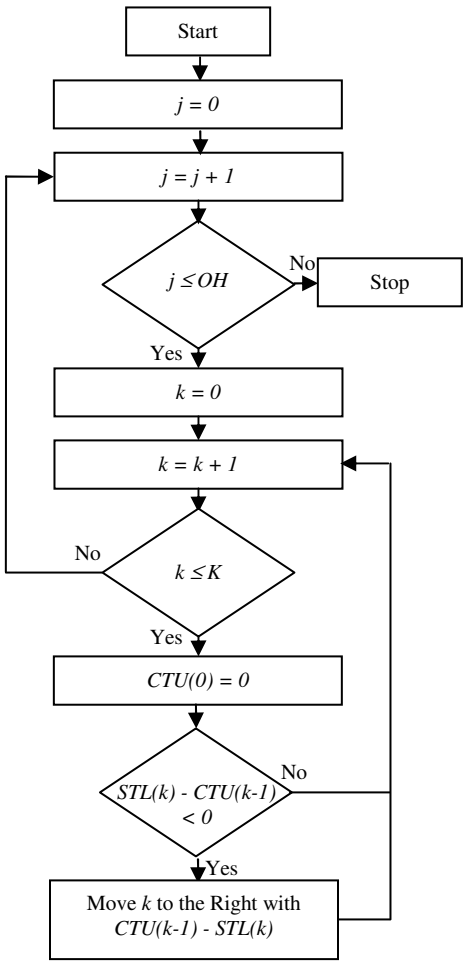

Figure 10.17. Algorithm used for satisfying the operations precedence constraints.

Figure 10.18 presents the third algorithm that takes care of the MH resources constraints within the system. After performing the steps in the algorithm below, presented also graphically in Figure 10.18, the operations assigned to a MH resource and initially scheduled at the same time are sorted using the Global Scheduler's operation ranking procedure, and each MH resource will be scheduled and execute only one operation at any time. The pseudocode for the MH resource constraint algorithm is presented in Appendix A.5.3.

1. Start
2. Initialize *T* with the value of *-1*, and *MT* with the value of the actual makespan at the starting of this process
3. Increase *T* value by *1*
4. If *T* is less than or equal to *MT*, go to 5; else go to 17
5. Initialize the MH resources index, *mh*, with the value *0*
6. Increase *mh* value by *1*
7. If *mh* is less than the total number of MH resources in the system, *MH*, go to 8; else go back to 3
8. Set *OP* equal to the total number of L/U operations to be started at time *T* by MH resource *mh*
9. If *OP* is equal to *0*, go back to 6; else go to 10
10. If *mh* is idle, go to 12; else go to 11
11. Increase all loading and unloading operations starting times with one time unit, and go to 14
12. If *OP* is equal to *1*, go back to 6; else go to 13
13. Use the GS operations ranking procedure
14. Increase *MT* value by *1*
15. Apply the jobs precedence constraints algorithm
16. Apply the operations precedence constraints algorithm, and go back to 6
17. Stop

where, *T* is the actual time in the schedule at which processing and MH operations are performed; *MT* is the completion time of the last unloading operation; and, *OP* measures the MH

operations to be performed by a MH resource at a particular time *T*.

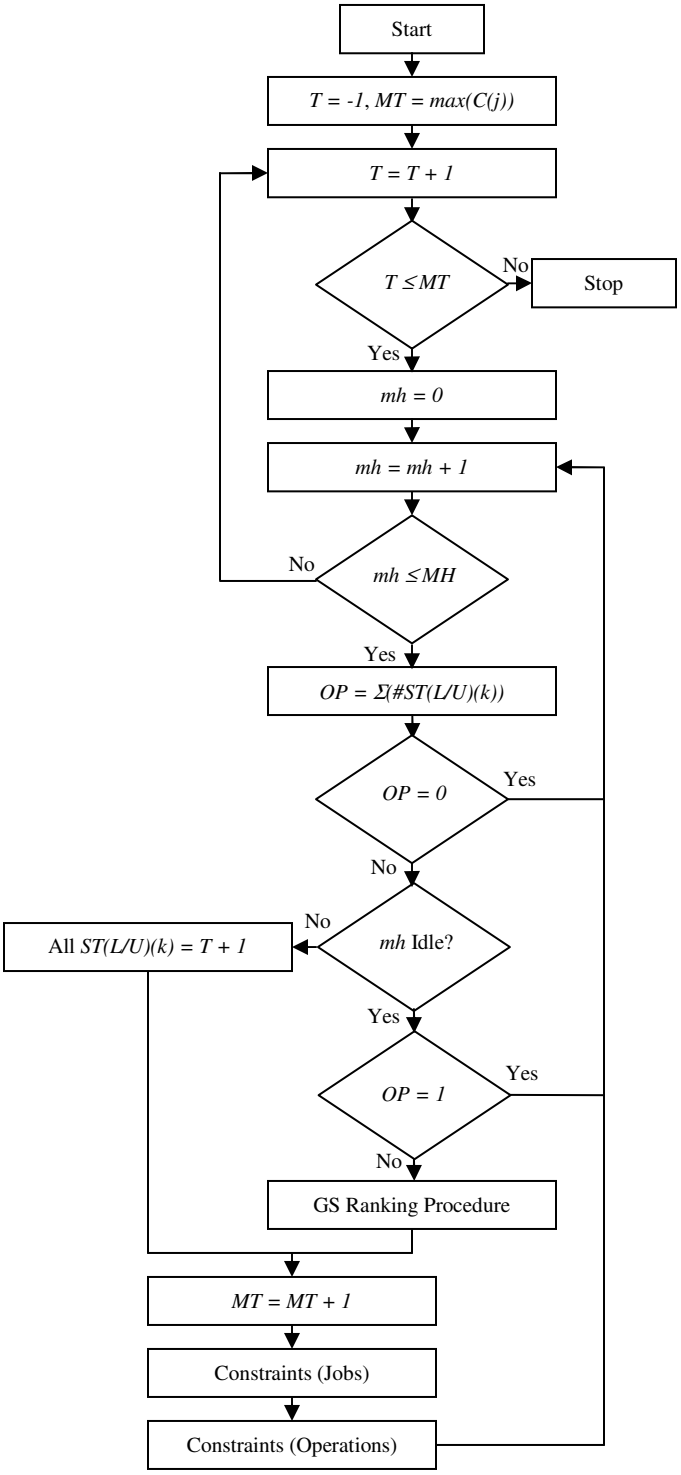

Figure 10.18. Algorithm used for satisfying the MH operations constraints.

10.4.5.2. Global Scheduler Ranking Procedure Algorithm

The Global Scheduler's algorithm for ordering MH operations to be performed by a MH resource when there is more than one MH operation needed at a time *T* in the work-in process schedule (step 13 in the algorithm for satisfying the MH operations above) is presented below, and in a flowchart form, in Figure 10.19. The pseudocode is given in Appendix A.5.4.

1. Start
2. If there is a job, *q*, for which the schedule shows that it finishes unloading and starts another loading at *T*, go to 3; else go to 4
3. Schedule the loading of job *q* at *T*, and go to 10
4. Calculate *RPJ*, the total remaining processing time, including MH operations, for the jobs that require a MH operation at time *T*, and *RPM*, the total remaining processing time, including MH operations, on the machines which have a job that require a MH operation at time *T*
5. If one or more of the (*T* + *RPJ*) or (*T* + *RPM*) values are equal to *MT*, go to 6; else go to 7
6. If tie, go to 7; else go to 9
7. Sort the L/U operations starting times using the job and operation precedence constraints between two consecutive operations
8. If tie, sort jobs randomly; else go to 9
9. Select the first job on list and schedule it at *T*
10. Increase the values of all starting times for the remaining operations by *1*
11. Stop

where, (*T* + *RPJ*) and (*T* + *RPM*) are two types of potential critical paths: $CP_1 = (T + RPJ)$ can appear along the operations of a job; $CP_2 = (T + RPM)$ can appear along the operations of a machine. *RPJ* and *RPM* are calculated only from schedule time *T* to *MT*. The potential critical paths duration are calculated using the following equations:

$$CP_1 = T + \sum_{k_1=i}^{K_1} \left(LT + PT + UT \right) \text{ for all } j = 1, \ldots, J \tag{10.22}$$

$$CP_2 = T + \sum_{k_2=i}^{K_2} \left(LT + PT + UT \right) \text{ for all } m = 1, ..., M \tag{10.23}$$

where, k_1 counts the remaining operations of a job; k_2 counts the remaining operations on a machine; i is the operation for which the L/U operations are considered at this step; K_1 is the total number of operations of a job; K_2 is the total number of operations on a machine; LT is the loading time; PT represents the processing time; UT is the unloading time; m counts the machines in the system; J is the total number of jobs in the system; and, M is the total number of machines. More details about the potential critical paths for the schedule will be presented in one of the next chapters when addressing the development of schedule lower bounds.

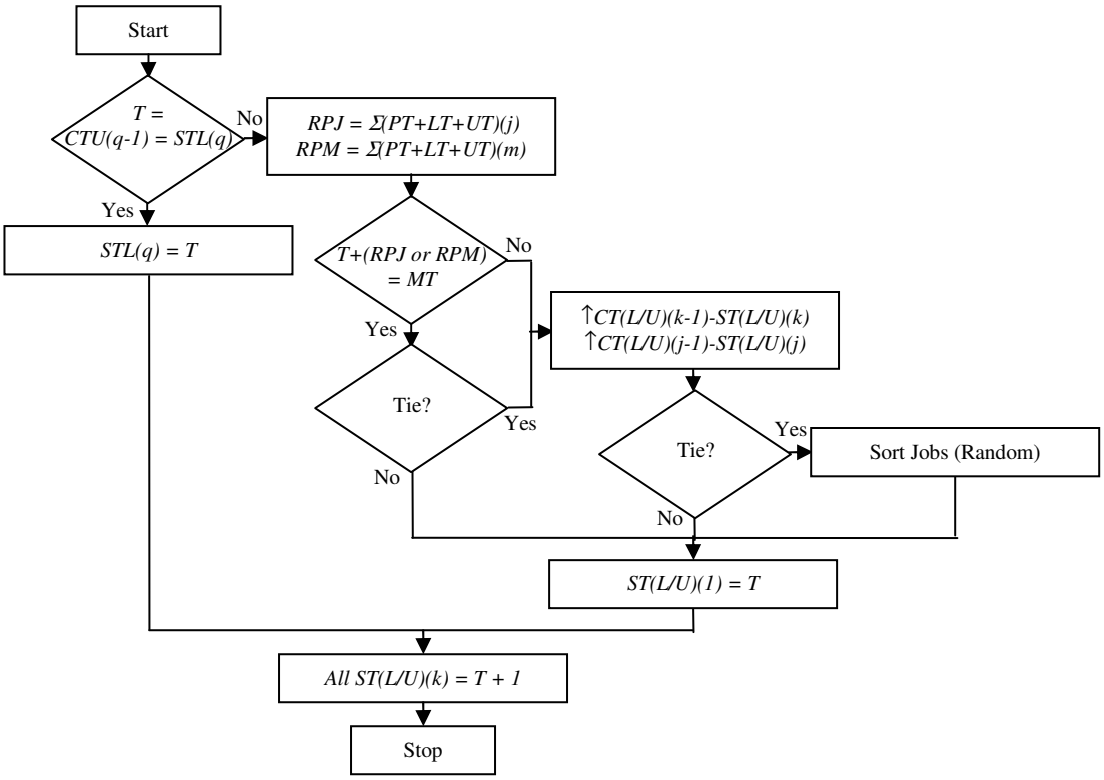

Figure 10.19. Global Scheduler's operations ranking procedure algorithm.

10.4.6. Choosing between Global Schedule and the System Level Schedule

In the case that there is a global schedule calculated by the imposed Order Holon's

deadlines, the following algorithm for choosing between the two schedules apply:

1. Start
2. Generate SLS, as above
3. If the UF value of the global schedule developed is greater than the UF value of the SLS, go to 3; else go to 4
4. Set the global schedule as the new SLS
5. Send the SLS to SMD
6. Stop

Graphically, the above algorithm is expressed as in Figure 10.20, below. The pseudocode for this algorithm is presented in Appendix A.6.

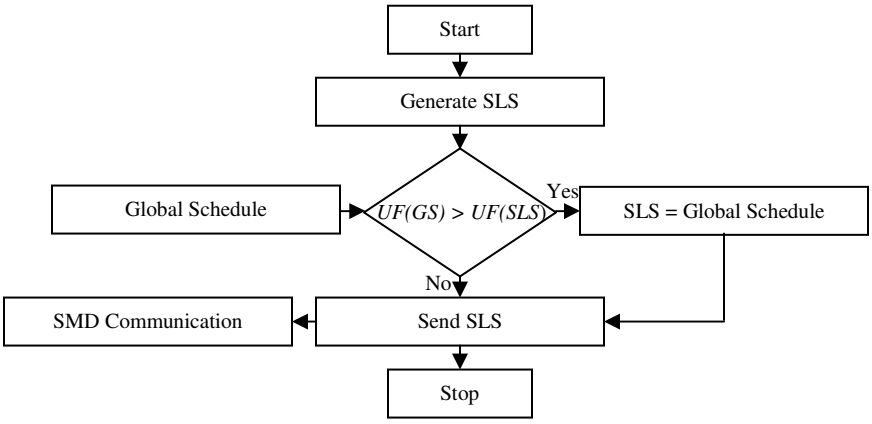

Figure 10.20. Choosing between a global schedule and the System Level Schedule.

10.4.7. Inter-Material Handling Holons Cooperation Algorithm

Because of the distributed architecture and decision-making process of the holonic system, the Material Handling Holons can change the already developed System Level Schedule by asking and offering help to perform one or more particular operations. This Inter-Material Handling Holon cooperation algorithm helps in improving the performance of the System Level Schedule by reducing the completion time of the jobs in the system.

The Material Handling Holon asking for help will offer a Material Handling Holon - Help Offer Price (*MHH-HOP*) and require that the particular operation be executed only at the specified time. The others Material Handling Holons will respond to this help request only if performing the operation(s) is increasing their internal account value.

$$MHH\text{-}HOP = w_j \sum_{k=1}^{K_s} \left((L/U) T_k \right)_j \tag{10.24}$$

where, w_j is the weight of the job *j*, $((L/U)T_k)_j$ is the loading or unloading time of operation *k* of job *j*; and, K_s is the number of operations of job j to be sub-contracted.

The following algorithm, presented both in step-by-step and graphical forms, is used by the Material Handling Holons to seek help from other MH resources in the architecture:

1. Start
2. Initialize *T* with the value of *-1*, and *MT* with the value of the actual makespan at the starting of this process
3. Increase *T* value by *1*
4. If *T* is less than or equal to *MT*, go to 5; else Stop
5. Set the completion time and unloading completion time of the imaginary operation *0* equal to *0* before any new operation
6. If there is a *Delay Type 2* in MH operations, go to 8; else go to 7
7. If there is a *Delay Type 3* in MH operations, go to 8; else go back to 3
8. Calculate *MHH-HOP*
9. Send help requests to other MHH in the system
10. Take *System Time*
11. If offers received from other MHHs, award operation(s) to that MHH, and go to 12; else go to 13
12. Send changes made to the GS
13. If *System Time* is equal to the offer deadline, go to 14; else go back to 10
14. Follow exiting SLS

where, *T* is the system time unit; *System Time* is the actual time of the Material Handling Holon controller; and, *Delay Type 2* and *Delay Type 3*, explained in one of the next chapters, are the

two possible delays introduced in the System Level Schedule when there is more than one MH operation to be performed in the same time by one MH resource. The pseudocode for this algorithm is given in Appendix A.7.

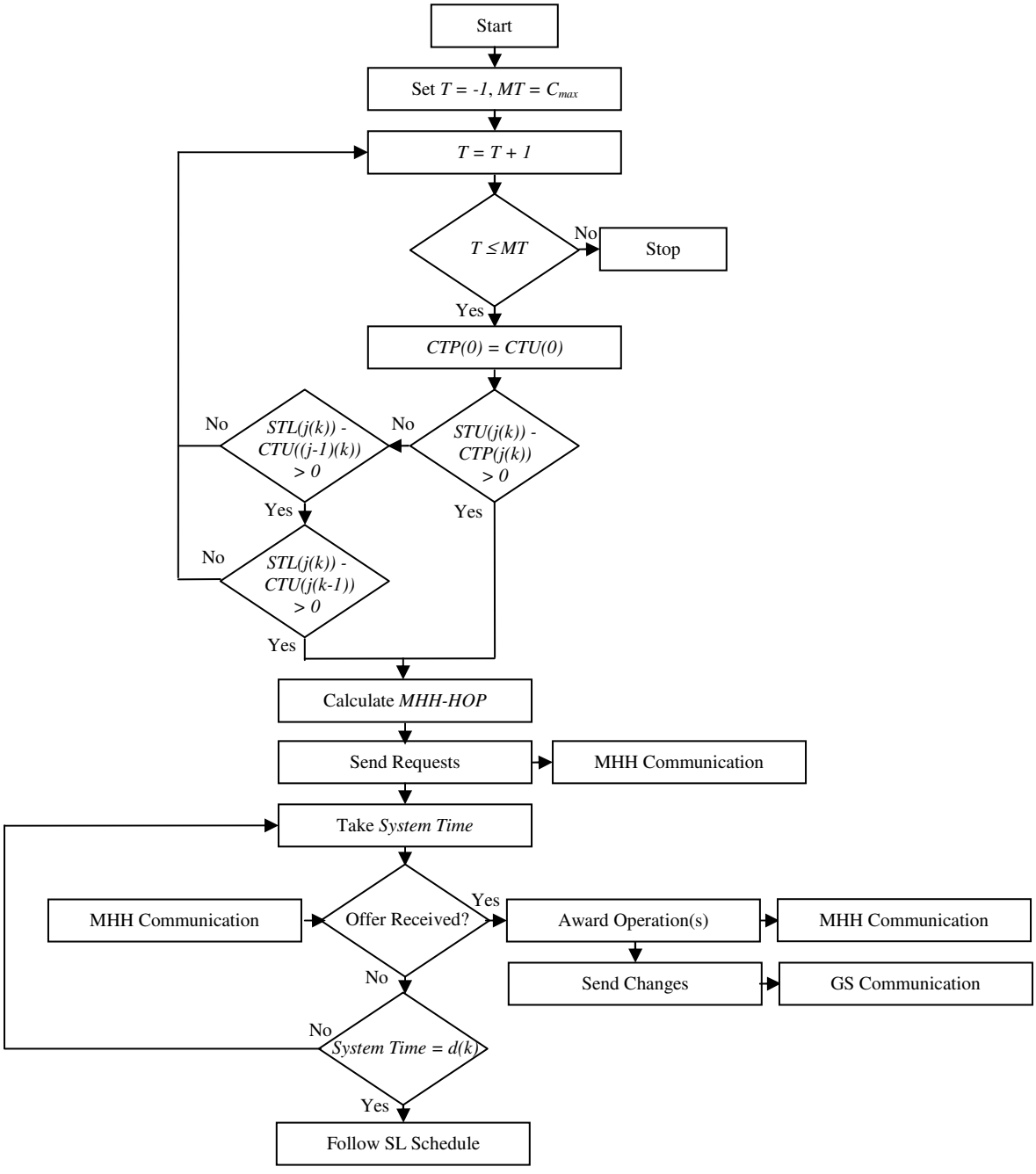

Figure 10.21. Inter-Material Handling Holons cooperation algorithm.

10.4.8. Order Holon Job Completion Algorithm

Using the job processing procedure described before, including the communication processes with the other entities in the holonic architecture, the Order Holon algorithm for job completion is presented below. The algorithm considers also MH resource hardware breakdowns. The Material Handling Holons accounts are updated at the completion of each job in which they participated, and take in consideration their relationships with the Order Holon responsible for that job and with the other Material Handling Holons in the system. A flowchart of the Order Holon job completion algorithm is presented in Figure 10.22. The pseudocode is given in Appendix A.8.

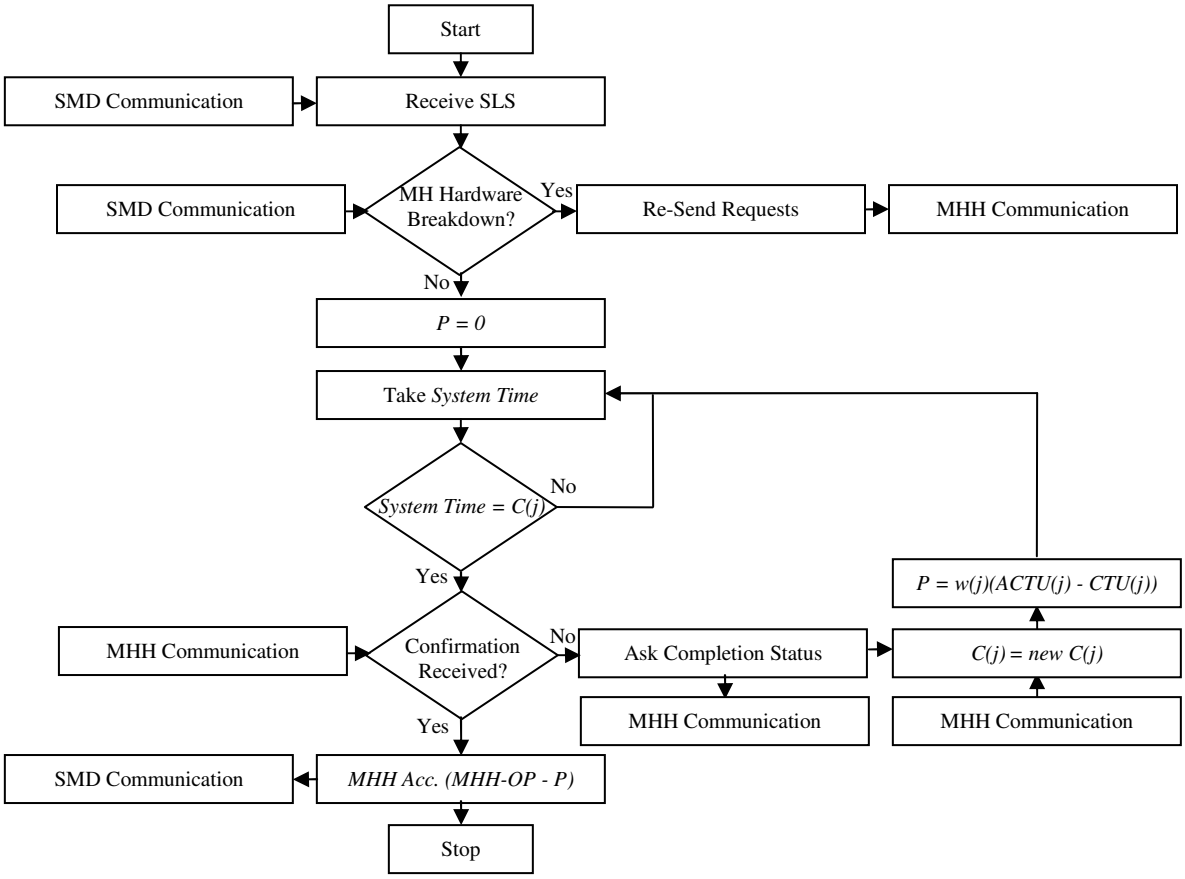

Figure 10.22. Order Holon job completion algorithm.

1. Start
2. Receive SLS from SMD
3. If there is a resource breakdown announcement distributed by SMD, re-send the requests to active MHHs; else go to 4
4. Set *P*, the penalty applied to contractor MHH if delay the completion of job more than in the initial IS
5. Take *System Time*
6. If the *System Time* is equal to the initially agreed completion time, go to 7; else go back to 5
7. If there is a confirmation of job execution coming from the contractor MHH, go to 11; else go to 8
8. Ask completion status from the contractor MHH
9. Update the job completion time with the new agreed time
10. Impose a penalty to contractor MHH, and go back to 6
11. Update the MHH account, and send this information to SMD
12. Stop

where, *P* is the penalty imposed by the Order Holon to the contractor Material Handling Holon if the job is not completed in the initially agreed time.

The penalty imposed by the Order Holon to the contractor Material Handling Holon if the job is not completed in the initially agreed time is given by the following equation:

$$P = w_j * \left(ACTU_j - CTU_j\right) \tag{10.25}$$

where, w_j is the weight of the job; $ACTU_j$ is the actual completion time for unloading the last operation of job j; and CTU_j is the initially agreed completion time for the unloading of the last operation of job j.

The new completion time is taken from the last version of the System Level Schedule sent by the System Monitoring and Database. At the completion of the job, the Material Handling Holon account is updated using the following equation:

$$Amount = MHH\text{-}OP - P \tag{10.26}$$

where, *MHH-OP* is the Material Handling Holon - Offer Price, defined before; and, *P* is the penalty given by equation (10.25). Initially, $P = 0$.

10.4.9. Inter-Material Handling Holons Sub-contracted Operations Completion Algorithm

The algorithm for checking the completion of sub-contracted operations and updating the sub-contractor account is presented below. Its flowchart is depicted in Figure 10.23, and the pseudocode is given in Appendix A.9.

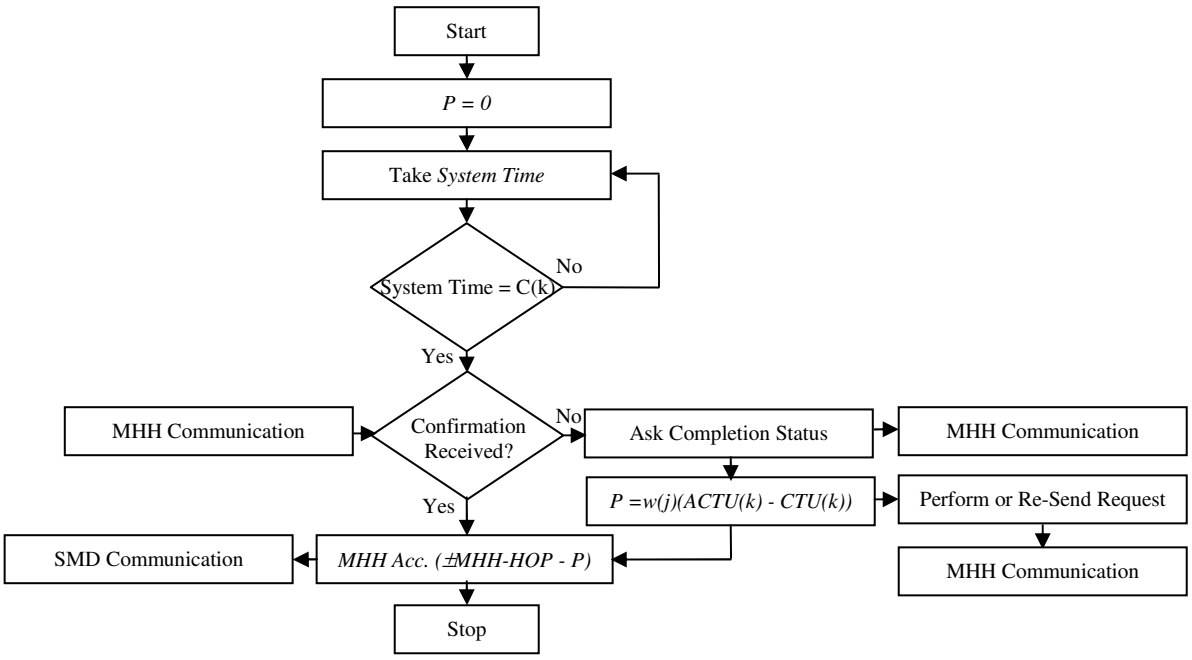

Figure 10.23. Material Handling Holon sub-contracted operation(s) completion algorithm.

1. Start
2. Set the initial penalty, *P*, equal to *0*
3. Take *System Time*
4. If *System Time* is equal to agreed completion time for the operation(s), go to 5; else go back to 3
5. If there is a confirmation of job execution coming from the sub-contractor MHH, go to 8; else go to 6
6. Ask completion status from the sub-contractor MHH

7. Impose a penalty to sub-contractor MHH, and go to 8
8. Update the MHH account, and send this information to the SMD
9. Stop

where, P is the penalty imposed by the manager Material Handling Holon to the sub-contractor Material Handling Holon if the operation(s) awarded is not completed in the initially agreed time.

Penalty, P, is given by the following equation:

$$P = w_j * (ACTU_k - CTU_k) \tag{10.27}$$

where, w_j is the weight of the job; $ACTU_k$ is the actual completion time for the unloading of the last operation sub-contracted; and, CTU_k is the initially agreed completion time for the unloading of the last operation sub-contracted.

The completion time is equal with the actual schedule time plus the needed MH operation time. At the completion of the job, the sub-contractor Material Handling Holon account is updated using the following equation:

$$Amount = MHH\text{-}HOP - P \tag{10.28}$$

where, *MHH-HOP* is the Material Handling Holon – Help Offer Price, defined before; and, P is the penalty given by equation (10.27).

10.5. Dealing with Uncertainties and Real-Time Response

In recent years, the real-time response to any changes that occur within manufacturing environments became a requirement for next generation manufacturing systems. Besides accommodating any changes in the manufacturing orders, achieving fault-tolerance to potential breakdowns and modifying the hardware architecture of the system without any significant delays is at least as important to the process of achieving a true real-time response. Three different types of MH hardware breakdowns that can occur during operations are studied along with the capability to add or remove MH resources at any time. When increasing or decreasing the number of active MH resources the holonic system adapts itself to make use of only those available.

10.5.1. New Jobs

Whenever a new job comes into the system, the job allocation process described above is started and the new job is accommodated in the System Level Schedule. The holonic system performs the job allocation process in real-time because of the reduced complexity of the computational problems to be solved by the entities in the holonic architecture. The only computational complex problem that, usually, appears is the Global Scheduler's attempt to deliver a global schedule by the stated deadline. Generally, this problem is not solved in the required time, its solving is abandoned, and the system works using the System Level Schedule obtained by combining the individual Material Handling Holons schedules.

System Level Schedule changes are only requiring a re-arrangement of the operations in the schedule of each machine taking into account operation precedence constraints on both jobs and machines and the MH resource constraint, process which can be done in real-time, as well. Other unexpected events during operations could be modifying work-in-process characteristics, such as changing due dates. An exchange of messages between the Order Holon and the contractor Material Handling Holons, followed by a re-sending of the individual Material Handling Holon schedule to the Global Scheduler for updating the System Level Schedule, and re-distribution of the System Level Schedule to all other entities are all the operations needed to accommodate this change.

10.5.2. Adding and Removing Material Handling Resources

Considering that the hardware part of a new MH resource is already installed, adding it to the system is simply done by sending out a new resource message to all other entities in the decentralized architecture. After being acknowledged by the other entities, the new MH resource is then available for any new MH operation requested. The process of removing MH resources from the architecture is similar to the process of adding resources. A simple message informs all the entities that a particular Material Handling Holon is down and/or not accepting any more requests, so the other Order and Material Handling Holons in the system will update their databases with this information.

10.5.3. System Response to Breakdown Occurrences

Since in real-world manufacturing environments breakdowns of the hardware part of manufacturing resources occur, the holonic system needs to have mechanisms to respond in short time to any MH resource hardware breakdown. Three different types of hardware breakdowns could occur: a MH hardware resource is idle and does not respond to a new command issued by its control unit; a MH hardware resource is in motion without carrying a part and gets blocked; and, a MH hardware resource which carries a part is in motion and gets blocked.

A mechanism to check for MH resources breakdowns is included in the Order Holon's job completion algorithm. In the first two cases, the control unit of the affected MH resource is sending a breakdown message to the System Monitoring and Database module, such that this information is made available in real-time to all the entities in the architecture. The job affected by the breakdown is re-announced as available for processing and the job allocation process will accommodate it to another Material Handling Holon. After the maintenance service is finished, the recovered Material Handling Holon is added back to the architecture by using the procedure described in the previous sections. The third type of breakdown is solved similarly to the first two, with the added activity of transferring the part from the broken MH resource to the one selected for that particular job after the new System Level Schedule is generated and announced to all the entities in the architecture. The part transfer is obtained using methods and tools specific to the MH resources that form the MH system, and requires the announcement of the exact position of the out of order MH resource in the system. Using the above procedures, both MH hardware reconfigurability and fault-tolerance characteristics are supported in real-time by the proposed holonic architecture.

11. Holonic Material Handling System Software Development

For the purpose of this dissertation, because there is no strict requirement that every entity in the system should have its own CPU, and there is no possibility to build a network formed from a large number of computers, the holonic system is simulated on one computer by associating a software object to every holon in the system. Using a single CPU for the computation necessities of several entities will delay the overall system response compared to a real network of computers. However, for research purposes, the imposed deadlines can be relaxed such that the performance of the proposed system could be evaluated.

All coding is done in C++ to benefit from its object encapsulation and inheritance characteristics. Encapsulation provides autonomy, while inheritance guarantees common features and same access rights to a group of objects. By incorporating internal decision-making and communication capabilities, these software objects become software agents capable of reasoning and exchanging information. When these software agents are part of the same decision-making process the architecture formed by them becomes a society of agents. Finally, by embedding the holonic characteristics to the individual entities in the already formed society, the software agents in this society will form the holonic architecture described in this research.

All the entities in the holonic architecture need to prepare, send, check for new messages and understand them, including receiving the information of how large the message is to be sure that it is received in its entirety. Standards for exchanging information among entities part of large computer networks are already developed, and software packages are commercially available. In this dissertation, communication is simulated using a file system on which the individual software programs can read and write data such that it is accessible to all programs that run in the background when simulating the holonic system.

PART IV

PERFORMANCE EVALUATION OF THE PROPOSED HOLONIC SYSTEM

12. Tools Used in the Performance Evaluation Process

To justify the use of the holonic approach for manufacturing control and assess the performance of the MH holonic system in comparison to conventional systems, a series of validation tests and a simulation study are carried out. Four types of theoretical lower bounds on the schedule makespan and the potential delays that may be introduced in the schedule by not selecting the best order of performing the MH operations are considered in the evaluation process. Moreover, to compare the holonic solution with the one given using the classical scheduling approach several algorithms, both optimal and heuristic, are developed for the Evaluation Module of the Global Scheduler. Artificial intelligence techniques are employed to develop these algorithms with the objective of minimizing the total completion time of both processing and L/U operations.

12.1. Lower Bounds on the Schedule Makespan when Considering Material Handling Operations

To estimate the influence of the MH operations on a processing job schedule, a lower bound (LB) needs to be determined. This LB will give the lowest makespan value that could be attainable after having also scheduled the MH operations. Four types of LBs on the schedule makespan, when considering MH operations, are described in this section.

12.1.1. Intuitive Lower Bound (LB_1)

An intuitive LB for the makespan of a schedule which includes MH operations, LB_1, is given by equation (12.1). This LB can be calculated easily and gives a first image of the makespan value when MH operations are considered.

$$LB_1 = \left(C_{max}\right)_{w/oMH} + \sum_{j=1}^{K_m} \left(\left(LT_j\right)_m + \left(UT_j\right)_m \right) \tag{12.1}$$

where, $\left(C_{max}\right)_{w/oMH}$ is the makespan when MH operations are disregarded; $\left(LT_j\right)_m$, $\left(UT_j\right)_m$ are the times for the loading and unloading operations of job *j* on machine *m*; *m* is the machine that

gives the schedule makespan; and, K_m is the number of jobs performed on machine m.

12.1.2. Algorithmic Lower Bound (LB_2)

A more precise LB on makespan (LB_2) when considering MH operations is obtained starting from the schedule of processing jobs and using the two-step algorithm described below, and presented in flowchart form in Figures 12.1 and 12.2. The output of the LB_2 algorithm is a schedule having the MH operations inserted between processing operations that takes into account the operation precedence constraints on both machines and jobs. The LB_2 algorithm is presented in pseudocode format in Appendix B.1.

The first part of the algorithm, presented below and in flowchart form in Figure 12.1., is performing the insertion of the loading and unloading times in the machine's schedules while assuring that the operation precedence constraints on all machines are satisfied.

1. Start
2. Initialize the machine index, *m*, with *0*
3. Increase *m* value by *1*
4. If *m* is less than or equal to the total number of machines in the system, go to 5; else go to step 2
5. Set the completion and unloading times associated with the imaginary operation *0* equal to *0*
6. Initialize the job index, *j*, with *0*
7. Increase *j* value by *1*
8. If *j* is less than *OH*, the total number of jobs, go to 9; else go to 15
9. If the idle time between two consecutive operations is greater than or equal to the loading time needed for the second operation, go to 11; else go to 10
10. Move all the remaining operations to the right with the amount necessary to insert the loading operation
11. Insert loading time in the schedule to the left of the second operation
12. If the idle time after a processing operation is greater than or equal to the unloading time needed for that operation, go to 14; else go to 13

13. Move all the remaining operations to the right with the amount necessary to insert the unloading operation
14. Insert unloading time in the schedule to the right of the processing operation, and go back to 7
15. If the idle time before the last operation is greater than or equal to the loading time needed for the last operation, go to 17; else go to 16
16. Move the last operation to the right with the amount necessary to insert the loading operation
17. Insert loading time in the schedule to the left of the last operation
18. Insert unloading time in the schedule to the right of the last operation, and go back to 3

When inserting a loading operation in the schedule, two possible situations, presented in Figure 12.3, could appear. In the first one, represented by inequality (12.2), the insertion of the loading time does not require any other change in the schedule, while the second case, represented by inequality (12.3), is further divided in two other sub-cases corresponding to the situations given in inequalities (12.4) and (12.5), requires that job *j* and all subsequent operations (processing and MH operations) be moved to the right in the schedule with the time amount T_5, given by inequality (12.6).

$$STP_j - CTP_{j-1} \geq LT_j \tag{12.2}$$

$$STP_j - CTP_{j-1} < LT_j \tag{12.3}$$

where, STP_j is the start time of a processing operation of job *j*; CTP_{j-1} is the completion time of an operation of job *(j-1)*, and, LT_j is the loading time for an operation of job *j*.

$$STP_{j-1} - CTP_{j-1} \geq 0 \tag{12.4}$$

$$STP_{j-1} - CTP_{j-1} < 0 \tag{12.5}$$

$$T_5 = CTP_{j-1} + LT_j - STP_j \tag{12.6}$$

Similarly, when inserting an unloading operation in the schedule (Figure 12.4) two possible situations, could appear. In the first one, represented by inequality (12.7), the insertion of the unloading time does not require any other change in the schedule, while the second case, represented by inequality (12.8), requires that, starting with the loading operation of job *(j+1)*,

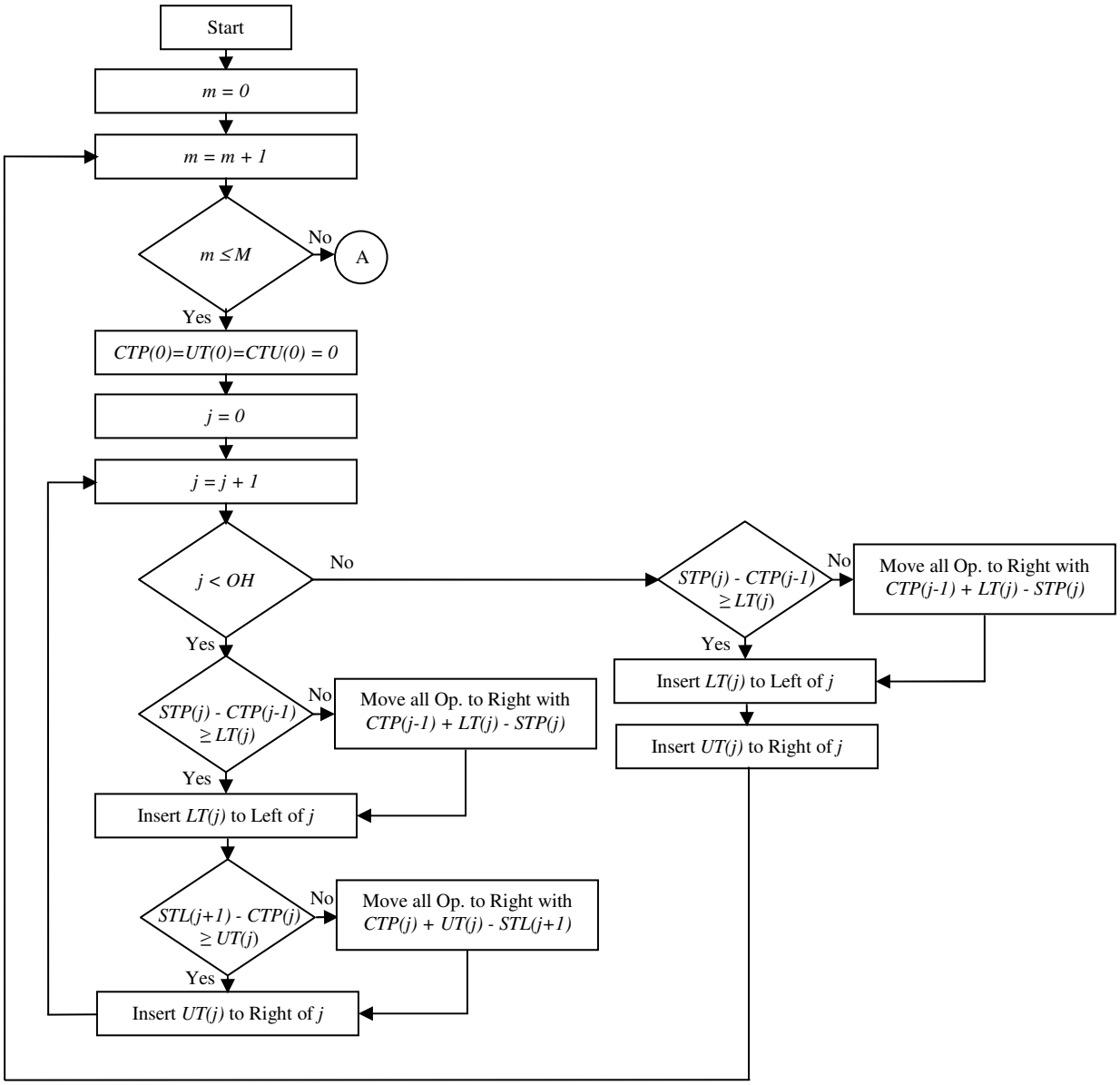

Figure 12.1. Algorithm to determine LB_2 on the schedule makespan.

all subsequent operations need to be moved to the right in the schedule with the time amount T_6 given by inequality (12.9).

$$STL_{j+1} - CTP_j \geq UT_j \tag{12.7}$$

$$STL_{j+1} - CTP_j < UT_j \tag{12.8}$$

$$T_6 = CTP_j + UT_j - STL_{j+1} \tag{12.9}$$

where, STL_{j+1} is the loading start time of an operation of job *(j+1)*; CTP_j is the completion time of an operation of job *j*; and, UT_j is the unloading time for an operation of job *j*.

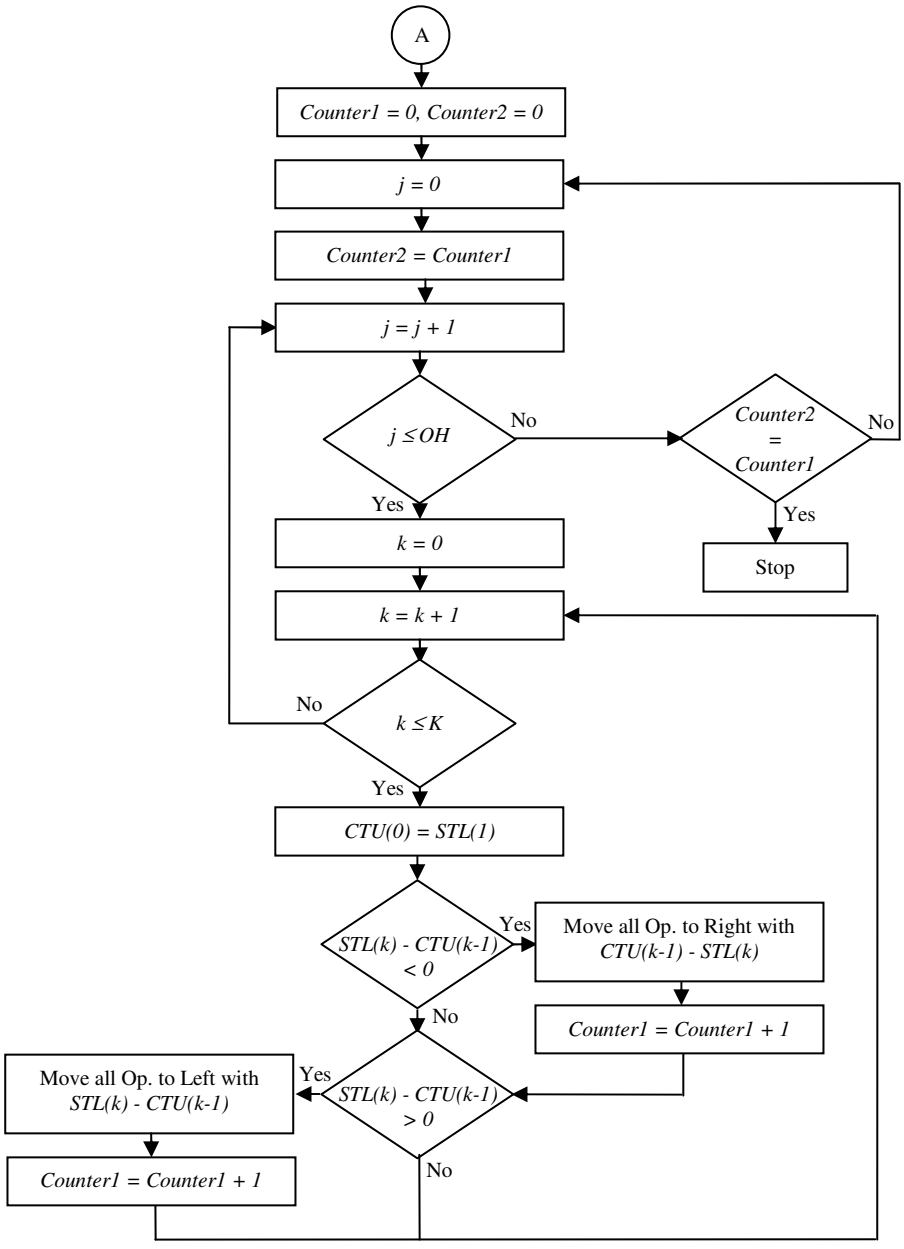

Figure 12.2. Algorithm to determine LB_2 on the schedule makespan (continued).

For the last operation on each machine, only the conditions for the insertion of the

loading operations are considered, the unloading operation is just added to the right of the last processing operation. Satisfying same job's operation precedence constraints is assured by running the second part of the LB_2 algorithm, presented below and in flowchart form in Figure 12.2.

1. Initialize *Counter1* and *Counter2* with *0*
2. Initialize the job index, *j*, with *0*
3. Set *Counter2* value equal to *Counter1* value
4. Increase *j* value by *1*
5. If *j* is less than or equal to the total number of jobs, *OH*, go to 6; else go to 16
6. Initialize the operation index, *k*, with *0*
7. Increase *k* value by *1*
8. If *k* is less than or equal to the total number of operations, go to 9; else go back to 4
9. Set the value of the unloading completion time of imaginary operation *0* equal to the loading starting time of the first operation
10. If the difference between the unloading completion time and loading starting time of two consecutive operations is positive, go to 11; else go to 13
11. Move all the remaining operations to the right such that the difference above becomes *0*
12. Increase *Counter1* value by *1*
13. If the difference between the unloading completion time and loading starting time of two consecutive operations is negative, go to 14; else go back to 7
14. Move all the remaining operations to the left such that the difference above becomes *0*
15. Increase *Counter1* value by *1*, and go back to 7
16. If *Counter2* is equal to *Counter1*, go to 17; else go back to 2
17. Stop

The algorithm keeps running until there is a run in which no change is made to the schedule. Any change made to the schedule takes into consideration the job precedence

constraints on machines described in the first part of the algorithm. Graphically, the operation precedence constraint satisfaction is explained in Figure 12.5.

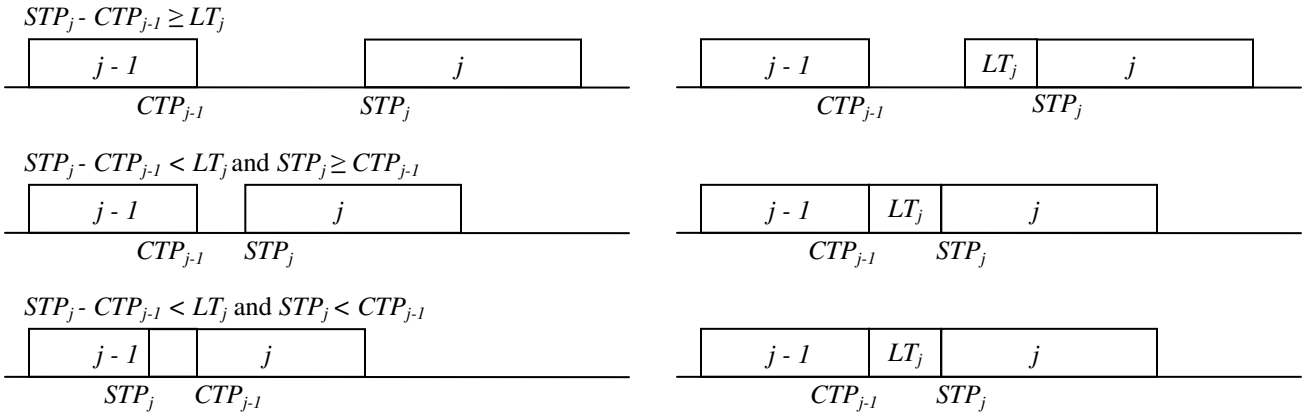

Figure 12.3. Insertion of a loading operation in the schedule.

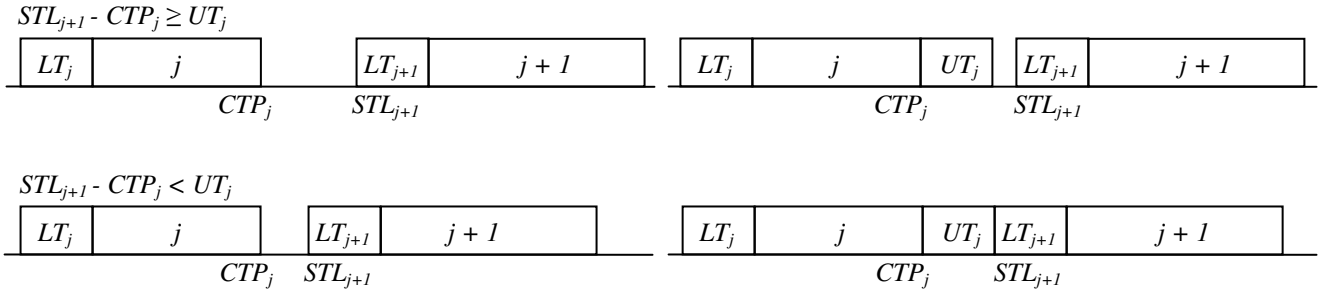

Figure 12.4. Insertion of an unloading operation in the schedule.

Depending on the value of T_7, given by equation (12.10), two types of changes in the schedule need to be performed. If T_7 has a negative value, all the remaining operations starting with the loading operation of job k need to be moved to the right in the schedule with the time amount T_8, given by equation (12.11). On the other hand, if T_7 has a positive value, all the remaining operations starting with the loading operation of job k need to be moved to the left in the schedule with the time amount, T_7. No change is needed in the schedule when equation (12.10) is equal to zero.

$$T_7 = STL_k - CTU_{k-1} \tag{12.10}$$

$$T_8 = CTU_{k-1} - STL_k \tag{12.11}$$

where, STL_k is the loading starting time of operation k of job j; and, CTU_{k-1} is the unloading completion time of the operation $(k-1)$ of job j.

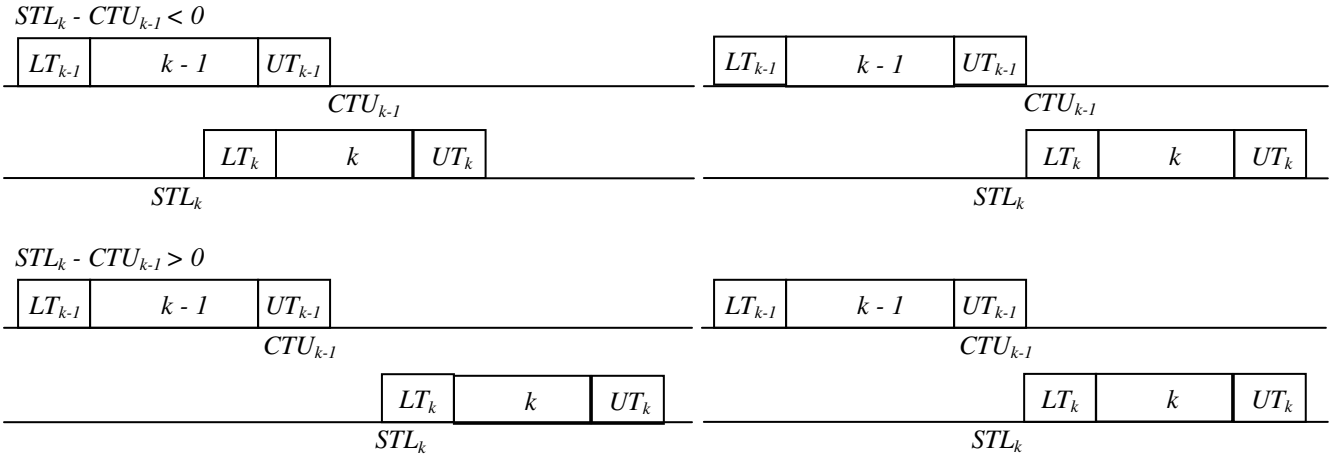

Figure 12.5. Satisfying operations precedence constraints.

The algorithm gives the theoretical best possible makespan value when there are enough MH resources to perform all the L/U operations whenever needed. This value is not always attainable since at a particular moment there could be scheduled more L/U operations than MH resources available in the system to perform them.

12.1.3. Improved Lower Bound (LB_3)

An even improved lower bound (LB_3) can be obtained by considering the critical paths in the schedule developed for LB_2. A critical path (CP) appears when any delay in a MH operation triggers an increase in the schedule makespan. The last operation of a CP is located on the machine that gives the schedule makespan, and the duration of a CP is equal to the schedule makespan. Of course, there could be more than one machine that has the same maximum completion time for their jobs, and consequently, there could be potentially more than one CP for a schedule.

12.1.3.1. Types of Critical Paths

Three types of CP could appear when scheduling the necessary L/U operations. The first one, CP_1, appears along the operations of a job, when there is no idle time between any processing and MH operations from the release time to the completion time of the job. The duration of the operations on a CP_1 is:

$$CP_1 = \sum_{k=1}^{K_j} \left(LT_k + PT_k + UT_k \right) = C_{max} \tag{12.12}$$

where, K_j is the number of operations of job j; LT_k is the loading time for operation k; PT_k is the processing time for operation k; UT_k is the unloading time for operation k; and, C_{max} is the schedule makespan.

The second type, CP_2, could appears along the schedule of operations on a machine when there is no idle time between any processing and MH operations, and its duration is given by the following equation:

$$CP_2 = \sum_{k=1}^{K_m} \left(LT_k + PT_k + UT_k \right) = C_{max} \tag{12.13}$$

where, K_m is the number of operations performed on machine m; LT_k is the loading time for operation k; PT_k is the processing time for operation k; UT_k is the unloading time for operation k; and, C_{max} is the schedule makespan value.

The third type of CP is a combination of CP_1 and CP_2, where sequences of critical paths types CP_1 and CP_2 can alternate several times from the loading time of the first job on the machine to the completion time of the last job. CP_3 types are much harder to detect even for small systems. Trying to find a feasible CP_3 leads to combinatorial explosion because the number of possible combinations increases with the increase in the number of machines, jobs, and operations. For the calculation of LB_3, CP_3 is taken in consideration only if it does not lead to combinatorial complex problems.

12.1.3.2. Calculation of LB_3

When the number of CPs described above is greater than the number of resources

available to perform the MH operations, the operations located on one or more CPs need to be delayed. This results in an increase in the value of the algorithmic LB_2. This improved LB_3 is calculated as follows:

$$LB_3 = LB_2 + f(CP) \tag{12.14}$$

where, LB_2 is given by the algorithm above; and, f(CP) measures the influence of the CPs on the LB, and is given by the next equation.

$$f(CP) = \left\lceil \frac{\left(\sum_p (CP_1)_p + \sum_q (CP_2)_q + \sum_r (CP_3)_r \right)}{\sum_{i=1}^{MH} (mh)_i} \right\rceil \tag{12.15}$$

where, p counts the number of CP_1 found; q counts the number of CP_2 found; r counts the number of CP_3 found; i counts the number of MH resources in the system; MH is the max number of MH resources in the system; and, $(mh)_i$ represents a MH resource.

LB_3 takes in consideration the number of available MH resources that can execute the MH operations, so it is more precise than LB_2. Still, for large problems the number of possible combinations is increasing exponentially, so the use of LB_3 might not be practical. LB_3, given by equation (12.14) will be considered as the makespan LB based on which the results given by the holonic approach are compared. Generally, even this improved LB can not be achieved because in many cases there could be scheduled more L/U operations than MH resources available in the system to perform them.

12.1.4. Exact Lower Bound (LB_4)

Lastly, by scheduling at each particular moment, a number of L/U operations lower or equal to the number of MH resource in the system, an even better LB may be determined. This further improved, LB_4, is likely to be greater than LB_3 calculated using the two-step algorithm and adjusted with the above CPs considerations, and thus give a better basis for solution evaluation, but in most cases it leads to combinatorial explosion. Finding which operation to perform first, and which to leave for the next available time slot requires checking many combinations of possible situations. Tree search techniques can be used for finding this improved

LB, but considering the real-time requirements imposed on the holonic system, this solution is, in many cases, not practical.

12.2. Potential Delays that May Appear when Scheduling Material Handling Operations

Depending on the number of MH resources available in the system and the operations' assignments to these resources, there are three types of delays that may appear when scheduling MH operations. First type of delay (*Loading Shorter Operation First Delay*), described by Smith *et al.* (1999), appears when two processing operations, on two different machines, have the same completion time, and there is only one MH resource available to perform the MH L/U operations. The order in which the next job is loaded on both machines is influencing the final makespan value, as presented in Figure 12.6. Loading the job with shorter processing time first will delay the completion time of the other job.

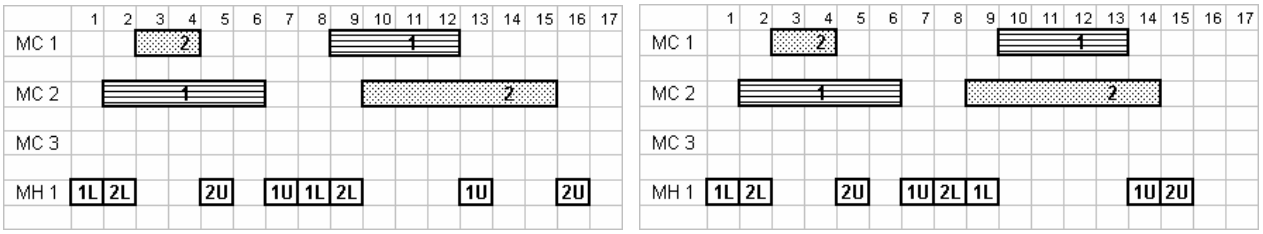

Figure 12.6. Example showing a type 1 delay in the System Level Schedule.

Delay types 2 and 3, explained below, are two other possible delays introduced in the final schedule when there is more than one MH operation to be performed in the same time by one MH resource. A delay type 2 (*Unloading Delay*) appears when there is another MH operation performed when there is a need for an unloading operation, or:

$$\left(STU_k\right)_j - \left(CTP_k\right)_j > 0 \tag{12.16}$$

where, $(STU_k)_j$ is the unloading starting time for operation k of job j; and $(CTP_k)_j$ is the completion time of processing operation k of job j.

A delay type 3 (*Loading Delay*) may appear when there is another MH operation

performed between the unloading and loading of two consecutive operations of different jobs, on the same machine, or:

$$(STL_k)_j - (CTU_k)_{j-1} > 0 \tag{12.17}$$

where, $(STL_k)_j$ is the loading start time for operation k of job j; and $(CTU_k)_{j-1}$ is the unloading completion time for operation k of job $(j-1)$.

The above condition is necessary, but not sufficient for a delay type 3. A delay appears when the starting of loading the second operation, $(STL_k)_j$, is not performed consecutively to the unloading of the previous operation of the same job $(CTU_k)_{j-1}$:

$$(STL_k)_j - (CTU_{k-1})_j > 0 \tag{12.18}$$

where, $(STL_k)_j$ is the loading start time for operation k of job j; and $(CTU_{k-1})_j$ is the unloading completion time for operation $(k-1)$ of job j.

The following figures show simple examples of these two types of delays. All the MH operations are initially assigned to only one MH resource. The use of Inter-Material Handling Holon cooperation may assign some of the operations to another MH resource, and thus reduce the total time for completion of the considered operations, such that there will be no delays.

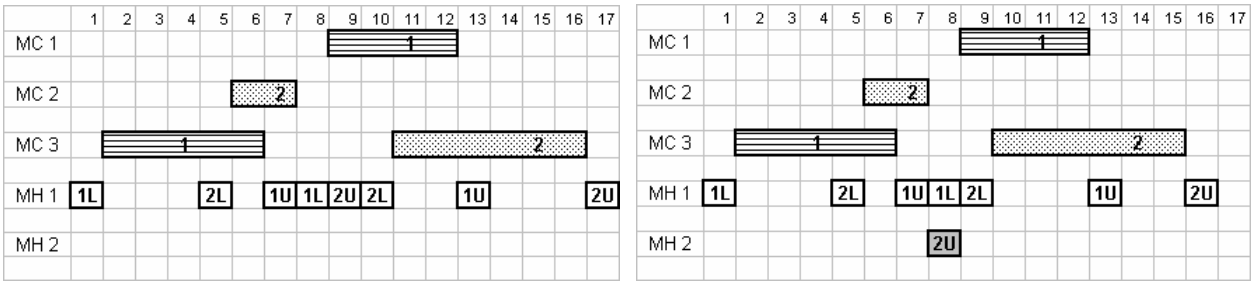

Figure 12.7. Example showing a type 2 delay in the System Level Schedule.

Figure 12.7 presents the *Unloading Delay* (type 2) in which the MH resource delays the unloading of job *2*, because is executing another MH operation (loading of job *1*). If another MH resource were available there would be no delays for both jobs *1* and *2*. Figures 12.8 and 12.9 present two cases in which constraint 12.17 is satisfied. The *Loading Delay* (type 3) is presented in Figure 12.8, were constraint 12.18 is also satisfied. The loading of operation *2* could be done earlier by another MH resource, and thus the completion time of job *2* can be reduced. In Figure

12.9, constraint 12.18 is not satisfied, so there is no delay in the schedule.

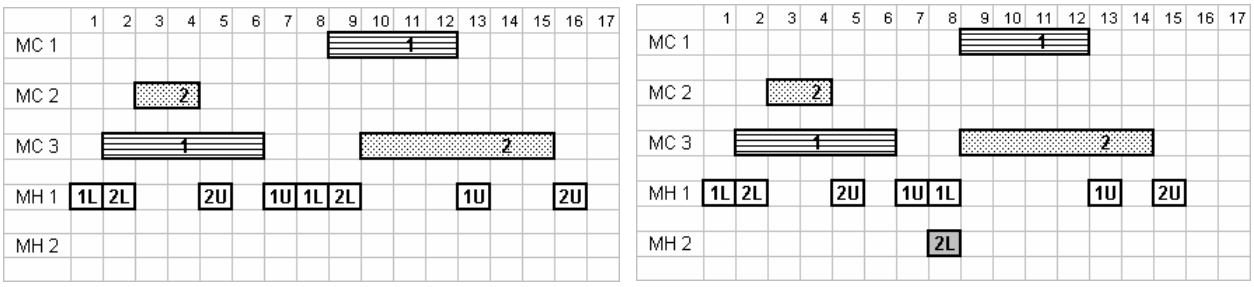

Figure 12.8. Example showing a type 3 delay in the System Level Schedule.

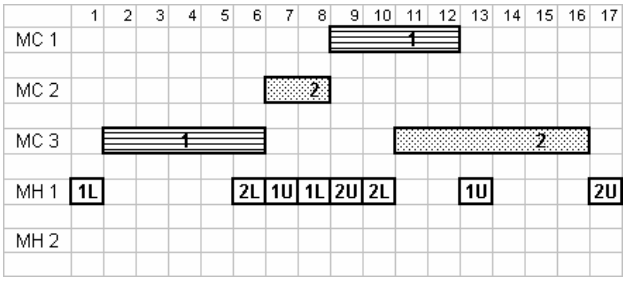

Figure 12.9. Example showing a non-delay situation.

12.3. Global Scheduler's Centralized Decision-Making

Besides using the algorithm for finding the theoretical LB_3 on the schedule makespan presented in one of the previous sections, to evaluate the solutions given by the decentralized holonic system, a series of algorithms are developed for the Global Scheduler, the global view entity part of the holonic system.

In the conventional control approach, or global scheduling, all possible combinations of MH resource assignments lead to a tree-search type of problem, for which optimal solving algorithms are usually accompanied by combinatorial explosion. The system is working using the centralized control approach in which the Global Scheduler acts as a central computer which is making all the decisions regarding the assignment of MH operations to the existing

resources in the system. One optimal and three heuristic algorithms are considered for the scheduling needs of the global view entity.

12.3.1. Material Handling Conventional Scheduling Approach

As stated in a previous chapter, calculating the makespan of a schedule by using the common way of disregarding MH operation times or just adding these amounts of time to the processing times of jobs usually gives unrealistic results. To find an exact value for the total processing time of the orders released in production, it is necessary to insert the MH operations between processing operations and recalculate the schedule makespan based on this new information. The mathematical formulation for scheduling MH operations is presented below:

$$Min \quad \sum_{k_m=1}^{K_m}\left(\left(LT_{k_m}\right)_{mc}+\left(PT_{k_m}\right)_{mc}+\left(UT_{k_m}\right)_{mc}\right) \qquad mc = 1, \ldots, M \qquad (12.19)$$

$$s.t. \quad \left(STL_{k_j}\right)_{jb}-\left(CTU_{k_j-1}\right)_{jb}\geq 0 \qquad k_j = 1, \ldots, K_j, \text{ and for all } jb = 1, \ldots, J \qquad (12.20)$$

$$\left(STL_{k_m}\right)_{mc}-\left(CTU_{k_m-1}\right)_{mc}\geq 0 \qquad k_m = 1, \ldots, K_m, \text{ and for all } mc = 1, \ldots, M \qquad (12.21)$$

$$\left(CTU_0\right)_{jb}=0 \qquad \text{for all } jb = 1, \ldots, J \qquad (12.22)$$

$$\left(CTU_0\right)_{mc}=0 \qquad \text{for all } mc = 1, \ldots, M \qquad (12.23)$$

$$\sum_{mh=1}^{MH}\left(L_{mh}+U_{mh}\right)\leq MH \qquad \text{at all times } T = 0, \ldots \qquad (12.24)$$

where, k_m is the index for the operations scheduled on machine mc; K_m is the total number of the operations scheduled on machine mc; M is the total number of machines; $\left(LT_{k_m}\right)_{mc}$ represents the loading time for operation k_m on machine mc, $\left(PT_{k_m}\right)_{mc}$ represents the processing time of operation k_m on machine mc; $\left(UT_{k_m}\right)_{mc}$ represents the unloading time of operation k_m on machine mc; k_j is the index for the operations of job jb; $\left(STL_{k_j}\right)_{jb}$ represents the loading starting time for operation k_j of job jb; $\left(CTU_{k_j-1}\right)_{jb}$ represents the unloading completion time of operation $(k_j\text{-}1)$ of job jb; K_j is the total number of operations of job jb; J is the total number of jobs in the system; $\left(STL_{k_m}\right)_{mc}$ represents the loading starting time for operation k_m on machine mc;

$\left(CTU_{k_m-1}\right)_{mc}$ represents the unloading completion time for operation *(k_m-1)* on machine *mc*; $\left(CTU_0\right)_{jb}$ represents the unloading completion time of imaginary operation *0* for job *jb*; $\left(CTU_0\right)_{mc}$ represents the unloading completion time of imaginary operation *0* for machine *mc*; *mh* is the index for the MH resources in the system; *MH* is the total number of MH resources in the system; L_{mh} is the loading operation performed by MH resource *mh*; U_{mh} is the unloading operation performed by MH resource *mh*; and, *T* is the actual time taken from the schedule.

The objective function, (expression 12.19) in the mathematical programming formulation minimizes the total time, processing and MH operations, on each machine considered in the problem. The first constraint (inequality 12.20) assures that the operation precedence constraint for every job is satisfied, by imposing the condition that no loading operation can be started before the unloading operation of the previous processing operation of every job is completed. The second constraint (inequality 12.21) assures that the operation precedence constraint on each machine is satisfied, by imposing the condition that no loading operation can be started before the unloading of the previous operation for each of the machines considered in the problem.

Constraints 12.22 and 12.23 are imposed specifically for the coding part of the algorithms presented in Appendix B, and assure that the imaginary completion time of the imaginary operation zero of every job and on each machine is equal to zero. The last constraint 12.24 takes in consideration the availability of MH resources and assures that at any time the number of MH operations scheduled is less than, or at most equal to, the number of MH resources available in the system.

12.3.2. Global Scheduler's Optimal Algorithm

To always find an optimal schedule for the assignment of MH resources, an exhaustive search needs to be performed every time changes occur in the system. Tree-search algorithms are used to perform this search of the best possible combination for scheduling MH operations. However, these tree-search algorithms are usually combinatorial complex. Even in practice these algorithms are not used due to the time needed to obtain a solution, to evaluate and validate the performance of the decentralized holonic system, an optimal algorithm is developed. The algorithm uses the best-first search technique coming from the AI field presented in Russell and

Norvig (2003) and adapts its tree-search by using back-loops, such that it gives optimal results.

The algorithm, called Enhanced Best-First Search (EBFS) and presented in Figure 12.10, starts with the LB_3 schedule developed by using the processing operations' schedule and the insertion of the MH operations between processing operations. This LB_3 gives the theoretical best possible makespan value that takes into account the operation precedence constraints on both jobs and machines. The EBFS algorithm checks at each particular time step for the violation of the MH resource constraint using inequality (12.24), and, if this constraint is violated, it considers all the possible combinations of MH operations.

Besides the MH resource constraint violation, two other types of constraint violations could appear when performing the rearrangement of MH operations. The first one, represented by the inequality (12.20) above, appears when attempting to load a job which is not unloaded from another machine, and the second one represented by inequality (12.21) above, appears when attempting to load a job on a machine which is not yet unloaded. To account for these violations the jobs needed to be loaded are delayed with the amounts T_9 and T_{10} given be equations (12.25) and (12.26) below, until there is no violation of the two constraints.

$$T_9 = \left(STL_{k_j}\right)_{jb} + T \tag{12.25}$$

$$T_{10} = \left(STL_{k_m}\right)_{mc} + T \tag{12.26}$$

where, $\left(STL_{k_j}\right)_{jb}$ represents the loading starting time for operation k_j of job jb; $\left(STL_{k_m}\right)_{mc}$ represents the loading starting time for operation k_m on machine mc; and, T is the time step used.

The number of possible combinations of scheduling the MH operations, N, is given by the binomial coefficient presented in equation (12.27) below, and it depends on the number of the available MH resources.

$$N = \begin{pmatrix} MH \\ op \end{pmatrix} \tag{12.27}$$

where, MH is the total number of MH resources in the system; and, op is the number of L/U operations needed to be scheduled at a particular time.

To select one of these combinations of MH operations to be performed at a particular time step, the intermediate makespan value, IC_{max}, is calculated using equation (12.28).

$$IC_{max} = EC_{max} + PC_{max} \tag{12.28}$$

where, EC_{max} is the existing makespan value calculated at that particular time for the already

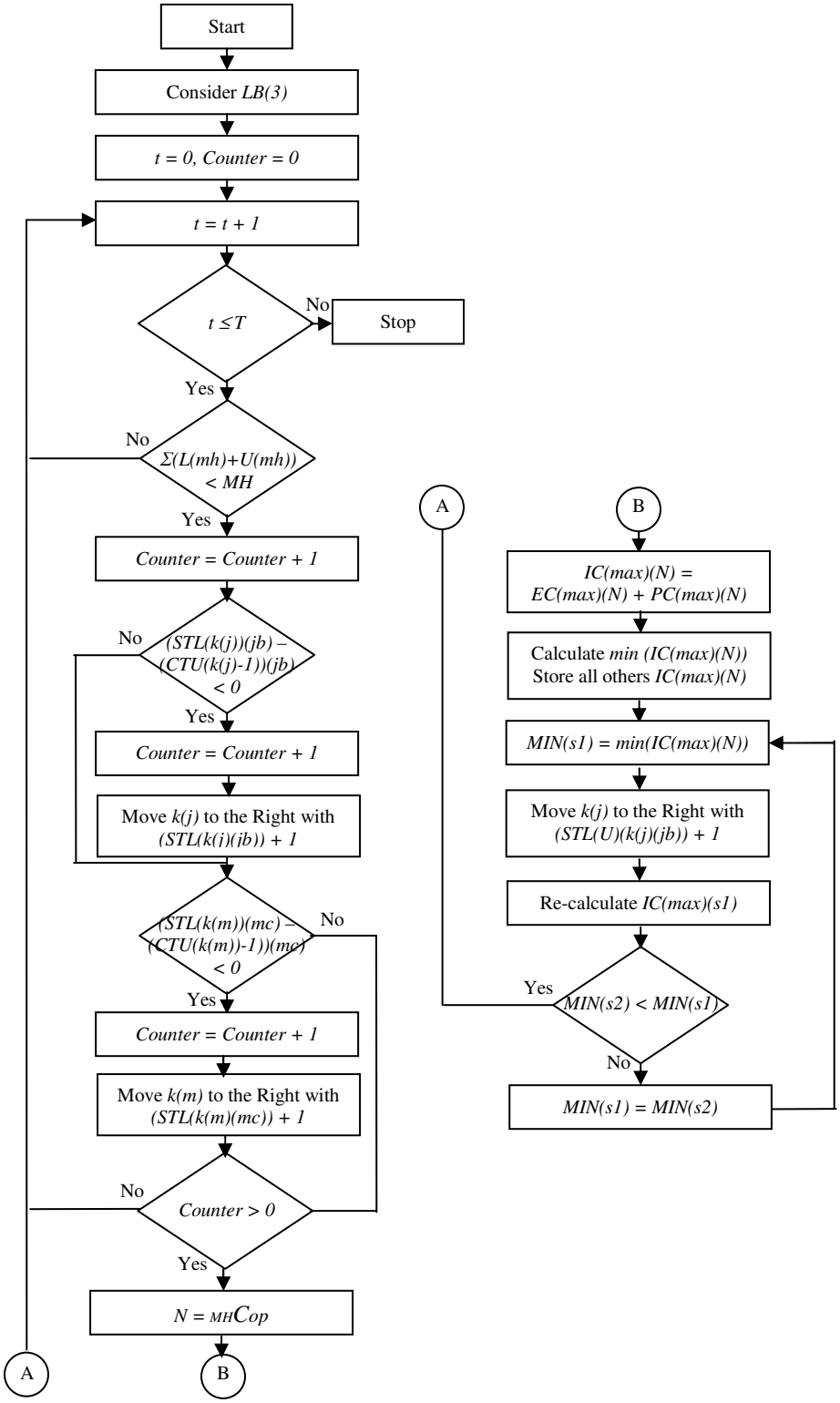

Figure 12.10. Global Scheduler's Enhanced Best-First Search optimal algorithm.

scheduled operations; and, PC_{max} is the potential makespan value of the remaining unscheduled operations. Initially, EC_{max} is equal to zero.

Then, the algorithm selects the combination that minimizes the intermediate makespan value, and in the same time stores the other intermediate makespan values obtained at that particular time step for future reference. The MH operations not selected in a particular combination are moved to the right with the amount T_{11} given by equation (12.29) below.

$$T_{11} = \left(ST(L/U)_{k_j}\right)_{jb} + T \tag{12.29}$$

where, $\left(ST(L/U)_{k_j}\right)_{jb}$ represents the loading or unloading starting time for the operation k_j of job *jb,* whichever comes first; and, T is the time step used, as before.

If the value of the intermediate makespan on the selected branch becomes greater than a stored value, the algorithm loops back to the stored intermediate makespan values and expands the node having the lowest makespan value. When all MH operations are scheduled, the EBFS algorithm stops. Compared to regular tree search techniques, the EBFS algorithm usually requires fewer steps to find the optimal solution. The pseudocode for the algorithm is given in Appendix B.2.

Figure 12.11 gives an example of the visited nodes (presented in black) on the branching tree for a small problem when applying the EBFS algorithm. For this problem the optimal schedule makespan is found to have a value equal to 59.

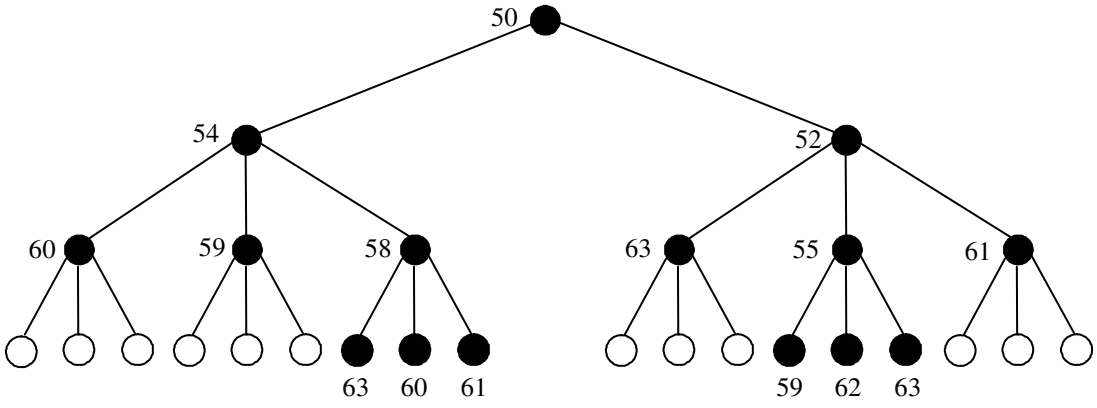

Figure 12.11. Example for the branching tree of the Enhanced Best-First Search algorithm.

In Figure 12.12 the same problem is solved using a regular Branch and Bound (B&B)

algorithm. The number of nodes visited to find the optimal makespan solution of 59 in the case of the EBFS algorithm is about a half compared to the number of nodes visited to find the same solution when applying the regular B&B algorithm.

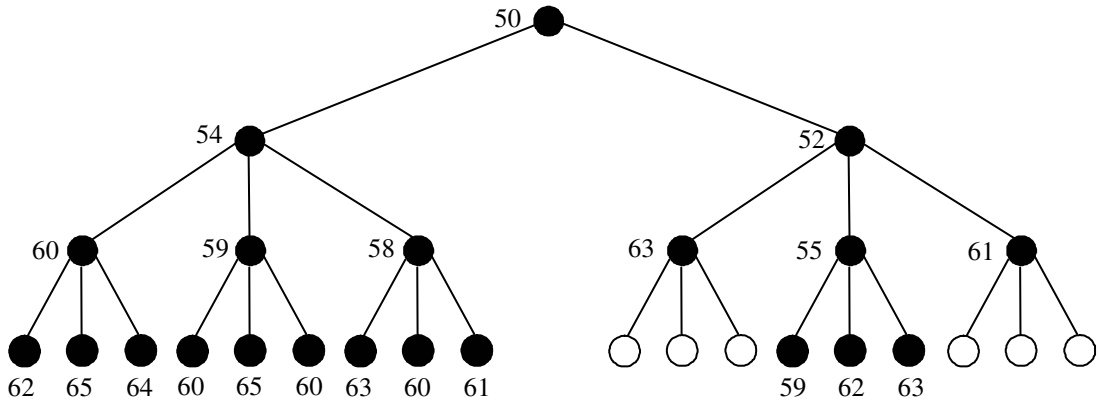

Figure 12.12. Example for the branching tree of a Branch and Bound algorithm.

12.3.3. Global Scheduler's Heuristic Algorithms

Three heuristic algorithms are developed to give the Global Scheduler the possibility of finding a feasible solution in a short amount of time and compare it with the solution given by the holonic approach. The first algorithm is designed to perform the MH operation selection process based on the completion times for the jobs in process, the second one considers the maximum time on every machine, and the third one takes into account the number of MH resource violations at each particular time step.

All three heuristics start with the same schedule as the optimal algorithm presented above. The two violations referring to the loading operations, described by inequalities 12.20 and 12.21 are avoided by scheduling first the unloading operations at each particular time unit. This mechanism reduces the complexity of the algorithms and the time necessary to obtain a feasible solution. The amounts of time T_{12} and T_{13} with which the loading operations need to be delayed to satisfy the two constraints are given by equations (12.30) and (12.31), respectively.

$$T_{12} = \left(CTU_{k_j-1}\right)_{jb} - \left(STL_{k_j}\right)_{jb} \tag{12.30}$$

$$T_{13} = \left(CTU_{k_m-1}\right)_{mc} - \left(STL_{k_m}\right)_{mc} \tag{12.31}$$

where, $\left(CTU_{k_j-1}\right)_{jb}$ represents the unloading completion time of operation *(k_j-1)* of job *jb*; $\left(STL_{k_j}\right)_{jb}$ represents the loading starting time for operation k_j of job *jb*; $\left(CTU_{k_m-1}\right)_{mc}$ represents the unloading completion time for operation *(k_m-1)* on machine *mc*; and, $\left(STL_{k_m}\right)_{mc}$ represents the loading starting time for operation k_m on machine *mc*.

12.3.3.1. Job Completion Time Heuristic Algorithm

After checking for the violation of the first two constraints in the mathematical programming model using the mechanism described above, the first algorithm (Figure 12.13; the pseudocode is given in Appendix B.3), called *Job Completion Time Heuristic* (JCTH), is then, checking for the MH resource constraint violation (inequality 12.24). If this constraint is violated, by having scheduled more MH operations than MH resources available to perform them, the JCTH algorithm is sorting the jobs in the reverse order of their completion time. If a tie appears the larger operation is taken first, and, if the tie persists, the operation to be scheduled is selected at random. The MH operations not selected in a particular combination are moved to the right with the amount T_{14} given by equation (12.32) below. The algorithm stops when, at all times, there are no more MH operations scheduled than the available MH resources.

$$T_{14} = \left(ST(L/U)_{k_j}\right)_{jb} + T \tag{12.32}$$

where, $\left(ST(L/U)_{k_j}\right)_{jb}$ represents the loading or unloading starting time for the operation k_j of job *jb,* whichever comes first; and, *T* is the time step used, as before.

12.3.3.2. Machine Completion Time Heuristic Algorithm

The second algorithm (Figure 12.14; the pseudocode is given in Appendix B.4), called *Machine Completion Time Heuristic* (MCTH), starts with the LB_3 schedule, and, first, runs the mechanism for avoiding the violation of the first two constraints in the mathematical programming model just as the JCTH algorithm. In the next step, the algorithm is checking for the MH resource constraint violation, and depending on the result of this step, is doing a sorting of the operations, or goes on the next time step. If the MH resource constraint is violated, the

MCTH algorithm ranks the machines in the reverse order of their completion times.

Considering the machines on which the jobs need to be loaded or unloaded, whenever there is a violation of the MH resource constraint, the necessary MH operations are sorted based on the already developed machine ranking. If a tie appears the larger operation is taken first, and, if the tie persists, the operation to be scheduled is selected at random. The MH operations not selected in a particular combination are moved to the right with the amount T_{15} given by equation (12.33) below. The algorithm stops when, at all times, there are no more MH operations scheduled than the available MH resources.

$$T_{15} = \left(ST(L/U)_{k_m}\right)_{mc} + T \tag{12.33}$$

where, $\left(ST(L/U)_{k_m}\right)_{mc}$ represents the loading or unloading starting time for the operation k_m on machine *mc*, whichever comes first; and, *T* is the time step used, as before.

12.3.3.3. Material Handling Resource Constraint Violation Heuristic Algorithm

Starting with the same schedule like the above two heuristics and using the same mechanism to easily avoid the violations of the first two constraints in the MH model, the third heuristic algorithm (Figure 12.15; its pseudocode is given in Appendix B.5), called *Material Handling Resource Constraint Violation Heuristic* (MH-RCVH), is considering at each particular time step the number of MH resource constraint violations in the remaining schedule.

The heuristic checks the MH constraint, and if there is a violation in the number of MH resources scheduled at any time, it selects the combination of MH operations that minimizes this number of violations. If a tie appears in the process, the larger operation is considered first, and, if the tie persists, the operation to be scheduled is selected at random. The MH operations not selected in a particular combination are moved to the right with the amount T_{16} given by equation (12.34) below. The algorithm stops when, at all times, there are no more MH operations scheduled than the available MH resources.

$$T_{16} = \left(ST(L/U)_{k_m}\right)_{mc} + T \tag{12.34}$$

where, $\left(ST(L/U)_{k_m}\right)_{mc}$ represents the loading or unloading starting time for the operation k_m on machine *mc*, whichever comes first; and, *T* is the time step used, as before.

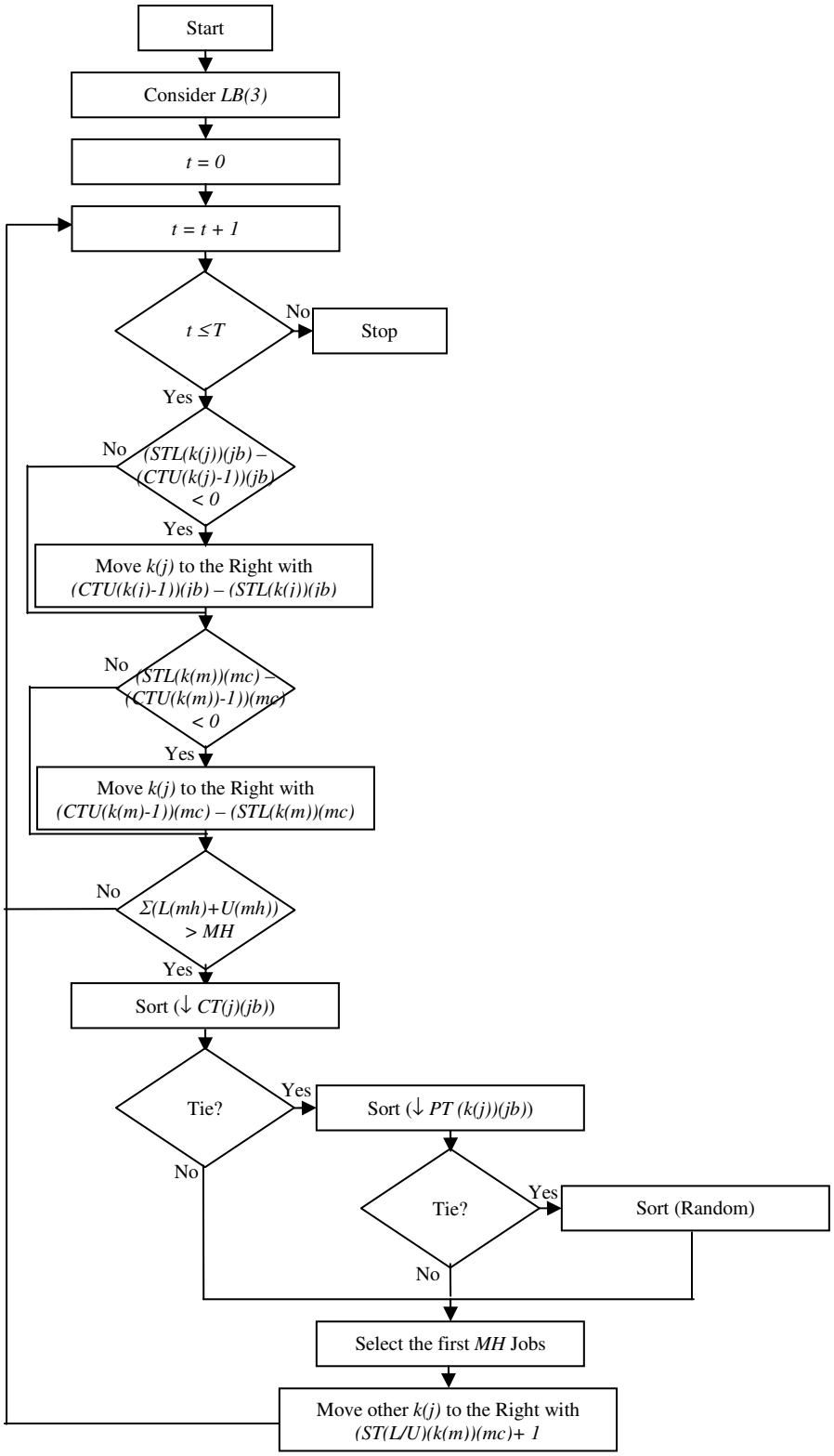

Figure 12.13. Job Completion Time Heuristic algorithm.

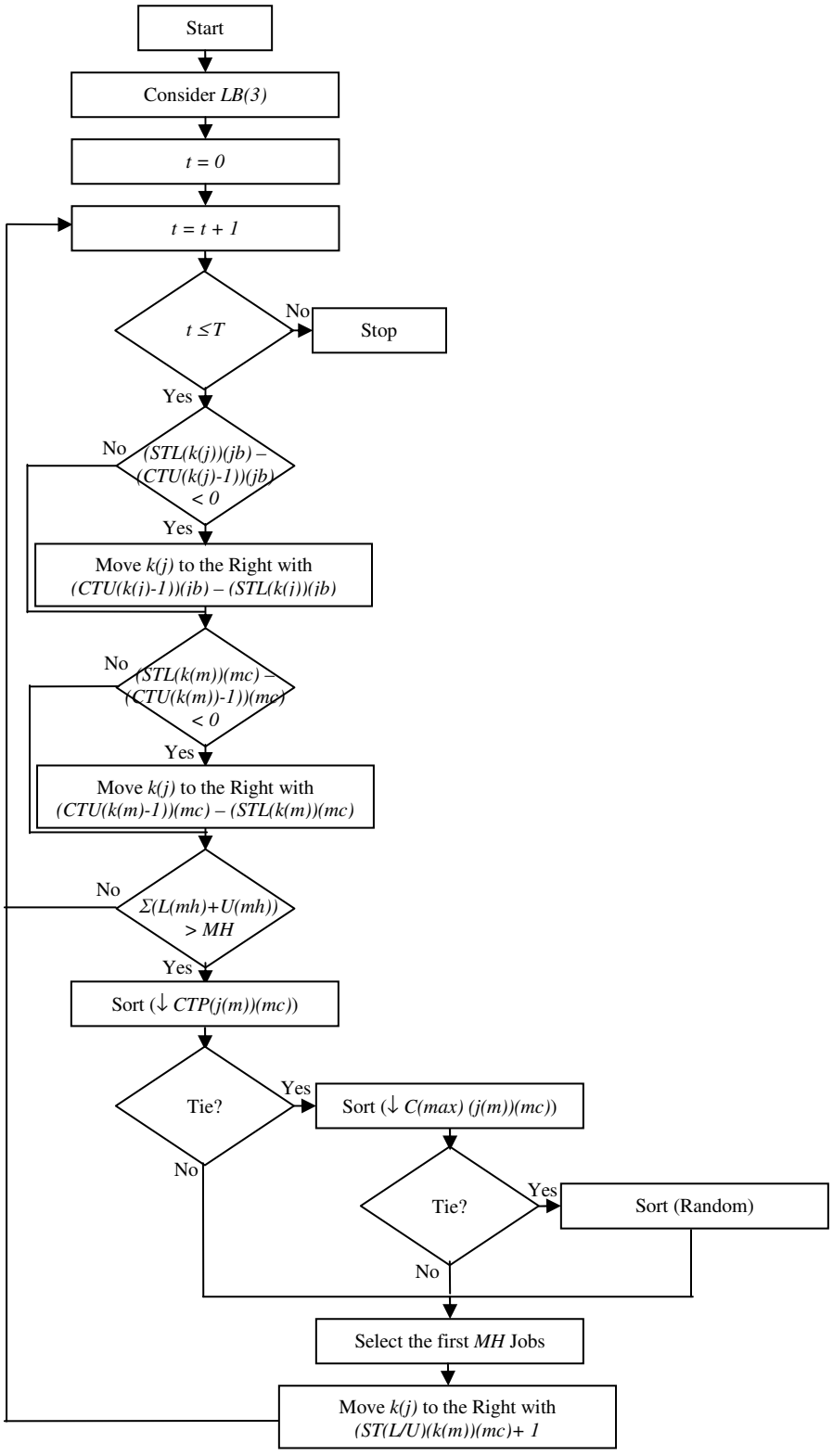

Figure 12.14. Machine Completion Time Heuristic algorithm.

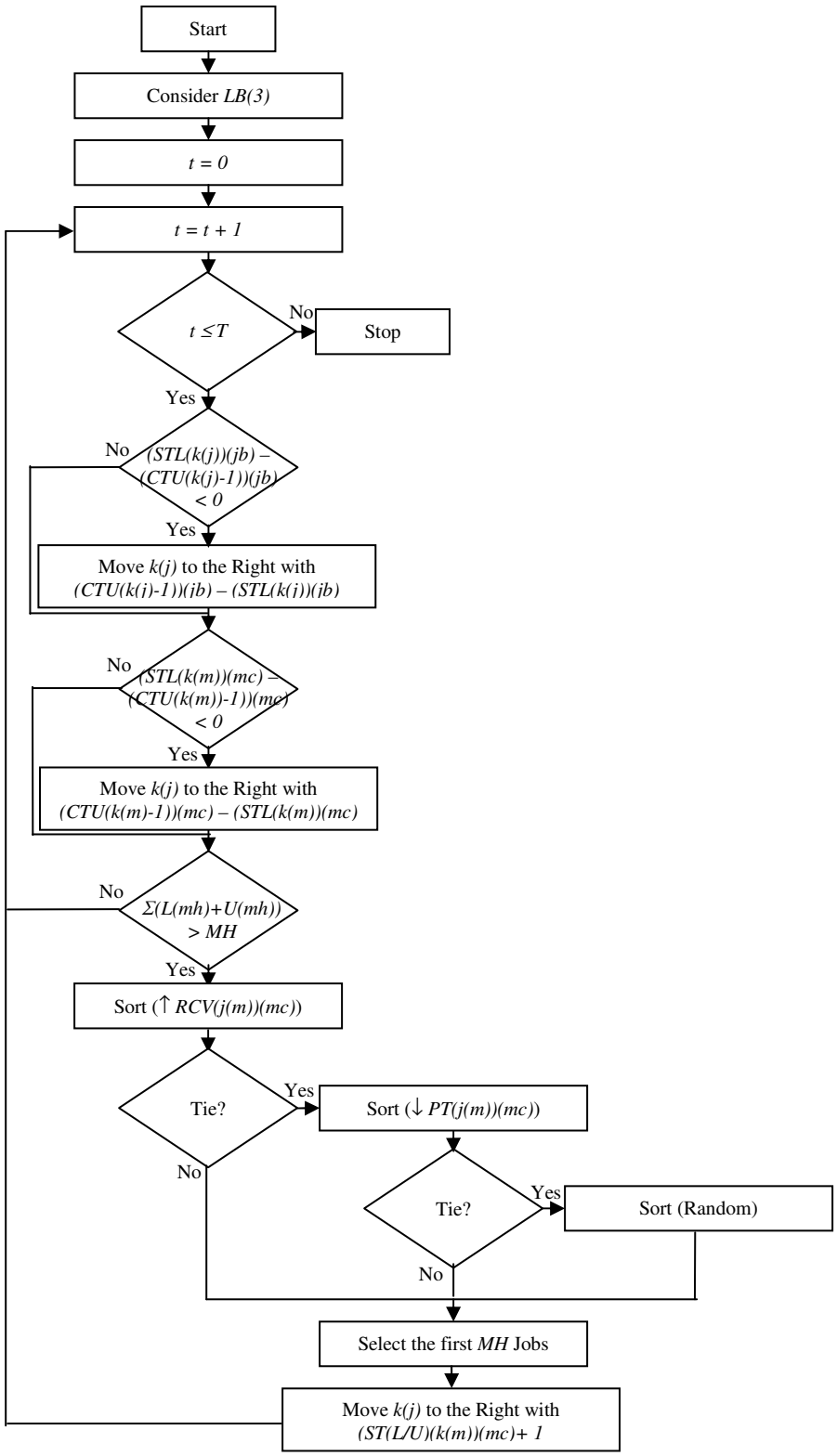

Figure 12.15. Material Handling Resource Constraint Violation Heuristic algorithm.

12.4. Adding Internal Learning Modules Analysis

In one of its general definitions, learning in intelligent systems is regarded as the "process based on experience… that provides higher efficiency…," and is considered to be "a subset of the given assignment for the intelligent system" (Meystel and Albus, 2002). As HMS research is strongly related to MAS research, advances in MAS theory can offer valuable tools for future development and enhancement of holonic systems. One of the research areas in MAS that has the potential to deliver significant improvements to the overall holonic system performance is the study of the agent learning mechanisms.

There are two types of machine learning (ML) techniques as reported in the DAI literature (Pendharkar, 1999), Single-Agent Learning (SAL), where "individual agents learn action strategies based on their perceptions of the local and global environment," and Multi-Agent Learning (MAL), where "agents not-only learn action strategies, but also learn to cooperate with other agents and share information for global problem solving." The Global Scheduler seems suitable to have incorporated Single-Agent Learning mechanisms for improving its response to the received MH operations requests. A Single-Agent Learning mechanism incorporated in each Material Handling Holon combined with suitable information exchange among them may lead to a form of emergent learning like that in Multi-Agent Learning.

12.4.1. Reinforcement Learning in Multi-Agent and Holonic Systems

Generally, ML is the study of methods for constructing and improving software systems by analyzing examples of their behavior rather than by through direct programming (Barto and Dietterich, 2004). In MAS, learning mechanisms are designed to give the agents in the architecture the capacity to improve their performance while in operation (Wang and Usher, 2004). Three different types of agent learning techniques are categorized in the DAI literature based on the type of feedback received from the environment: supervised, reinforcement, and unsupervised learning (Weiss, 1996).

Reinforcement learning (RL) is the mechanism in which the agent learns an optimal behavior for a particular task using trial-and-error techniques, and is based on the idea that the agent receives a reward depending on the degree of satisfaction resulted from taking a particular

course of action. The general framework for the reinforcement learning problem, presented in Figure 12.16, was developed by Sutton and Barto (1998) and considers the interaction between the agent and its environment over a potentially infinite sequence of discrete time steps. At one time step, the agent identifies the state, s_t, of the system and selects an action, a_t, that moves the system to another state. Based on the desirability of the new state, s_{t+1}, the agent receives a positive or a negative reward, r_t, (also called reinforcement). Reinforcement learning techniques have been used in recent years in a large number of applications, including manufacturing scheduling (Zhang and Dietterich, 1995, Zhang and Dietterich, 1996, Aydin and Oztemel, 2000, Kretchmar, 2002, Claus and Boutilier, 1998, Sallans and Hinton, 2001).

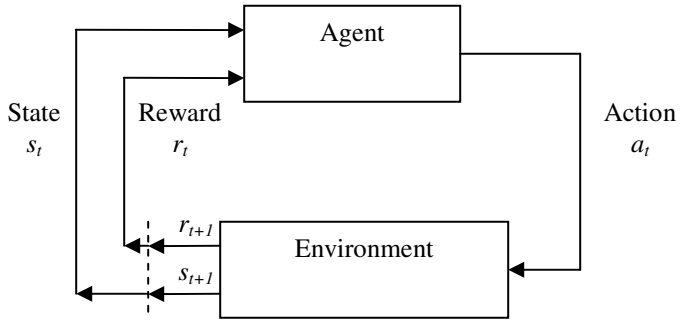

Figure 12.16. General reinforcement learning framework. Adapted from Sutton and Barto (1998) with the kind permission of MIT Press[8].

One of the most widely used reinforcement learning algorithm, called Q-learning was developed in 1989 by Watkins (Sutton and Barto, 1998). The Q-learning algorithm works in such a way that the agent increases its knowledge during execution of the processes in which it is involved. The agent learns the state-action pair value, $Q(s_t, a_t)$, by receiving feedback from the environment. In other words, by executing trials, the agent learns the value of each action in each particular state (Russell and Norvig, 2003).

---

[8]Reprinted from *Reinforcement learning: An Introduction*, chapter 3, figure 3.1 at page 52. The reference for the original material is as follows: Sutton, R. S. and Barto, A. G., *Reinforcement learning: An Introduction*, Bradford Books, MIT Press, Cambridge, MA, 1998.

12.4.2. Global Scheduler's Q-Learning Algorithm

In the holonic system, the Q-learning algorithm is designed to improve the solutions in both quality and timely response, given by the Global Scheduler. As stated before a Q-learning agent learns an state-action pair function, $Q(s_t, a_t)$, which tells how valuable it is to do action a_t in state s_t, at time t. The policy for state s_t can be computed as the action a_t, that maximizes $Q(s_t, a_t)$ (Barto and Dietterich, 2004). A model of an exploratory Q-learning agent is presented in Russell and Norvig (2003). Using this model the general Q-learning algorithm for scheduling the MH resources in the holonic system is presented below. The following notations are used:

- $t \in [0, T]$ is the time step, where T is the final schedule makespan value which is to be found
- $s_t \in S$, is the state of the system at a particular time t, where S is the vector of all states of the system
- $a_t \in A$, is an action selected at a particular time t, from the set of actions, A, available for that particular time
- $Q(s_t, a_t)$, called value function, is the value of doing action a_t in state s_t
- $f(s_t, a_t)$ is the table of frequencies for the state-action pairs (s_t, a_t)
- r_{t+1} is the reward received by the agent after executing action a_t in state s_t
- $\alpha \in [0, 1]$, called learning rate, is the parameter which influences the rate of learning.
- $\beta \in [0, 1]$, called discount factor, is the parameter that considers the influence of the chosen state-action pair, (s_t, a_t), on the emerging policy.

The objective of the agent is to find a policy that relates the set of states to the set of actions such that the reward is maximized. If the new state is desirable, the reward is positive; in the case that the new state is not desirable a penalty (negative reward) is given to the last state-action pair. The formula to calculate the reward is given by the following equation.

$$r_{t+1} = V(s_t) - V(s_{t+1}) \tag{12.35}$$

where, $V(s_t)$ and $V(s_{t+1})$ are the number of violations of the MH resource constraint at times t and $t + 1$, respectively.

The Q-value is update using a recursive formula, presented below, that takes into account the actual $Q(s_t, a_t)$ value, the frequency of a particular state-action pair, the reward received after taking a particular action, and the learning (α) and discount (β) factors.

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(f(s_t, a_t))(r_{t+1} + \beta Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)) \quad (12.36)$$

Using the above considerations the Q-learning algorithm used by the Global Scheduler for scheduling the MH operations is presented in pseudocode form below:

```
Initialize t ← 0
While (t ≤ T)
   Initialize s_t, a_t, f(s_t, a_t), Q(s_t, a_t), r_t ← 0
   If in state s_t the MH resource constraint (12.24) is violated
      Select a_t having the best Q(s_t, a_t) from the actions in s_t
      Update f(s_t, a_t) ← f(s_t, a_t) + 1
      Apply reward r_t+1 = V(s_t) – V(s_t+1)
      Update the entry in the Q(s_t, a_t) table
         Q(s_t, a_t) ← f(s_t, a_t)(Q(s_t, a_t) +
                     α(f(s_t, a_t))(r_t+1 + β Q(s_t+1, a_t+1) – Q(s_t, a_t))
t ← t + 1
s_t ← s_t+1
```

When using large discount factors, the chosen state-action pair has an increased importance compared to the reward, whereas when using small discount factors the reward is more important. Reinforcement learning mechanism is based on both exploitation and exploration (Sutton and Barto, 1998). The agent has to exploit what it has already learned, and in the same time it has to explore more to increase its knowledge. Exploitation increases the reward received, while exploration is used for finding better actions for the future.

Exploitation and exploration are controlled by the learning rate. Learning rates close to one will give more importance to the exploration part. A value of one will erase any old $Q(s_t, a_t)$. As the learning rate is decreased, the $Q(s_t, a_t)$ is starting to have more importance in the decisions made by the agent. Values of learning rate close to zero will give more importance to the exploitation part. Starting with a value of one, during simulation the learning rate is decreased in a systematic way, such that at the end of the simulation, its value is close to zero. Using a

recursive formula like the one in equation (12.37) the learning rate can be decreased gradually.

$$\alpha_t = y\alpha_{t-1} \tag{12.37}$$

where, α_t is the learning rate at time t, and $y \in (0, 1)$ is the depleting factor.

12.4.3. Multi-Agent Learning in the Holonic System

In ML theory, Multi-Agent Learning is based on Single-Agent Learning mechanisms and the emergence properties of MAS. The knowledge gained by the learning entity is influenced by the decisions made and the actions performed by all the other entities in the architecture (Alonso *et al.*, 2001). Just like in the case of emergent schedules, the knowledge acquired by the learning entity is interactive in the sense that it comes from several distributed sources that cover the entire architecture, so the process can be considered as learning at the system level.

In the case of the holonic system, to benefit from the potential improvements that may be delivered by an emergent Multi-Agent Learning process, several conditions need to be satisfied:

- A Single-Agent Learning mechanism needs to be incorporated in the internal structure of each Material Handling Holon in the system.
- A reliable way of communication among the Material Handling Holons capable of sustaining large amounts of messages in real-time needs to be designed, tested and validated.
- The holonic system needs to be formed of enough Material Handling Holons to generate several different solution paths for the same problem.

Adding reinforcement learning modules to the internal architecture of the Material Handling Holons in the holonic system may improve the solutions obtained when dealing with large real-world problems. Each Material Handling Holon attempts to learn the value function of a particular task on its own. Generally, in a learning environment, if the number of trials is large then the agent is presumably selecting a better action than in the case that the number of trials is small. A holon in the decentralized architecture can use the information exchanged with other peer holons and, consequently, improve its knowledge about a particular state-action pair. By combining the experience accumulated by all holons in the system using their individual learning

capabilities, at the system level it results an emergent type of learning that may further help in obtaining better and faster results.

12.5. Simulation Model

The simulation study is conducted for both the decentralized holonic control approach and the conventional centralized scheduling approach discussed in this dissertation. The operation and the results given by the MH system under these two control policies when working in a simulated job-shop environment are the subject of the simulation study. By comparing the results delivered by the two alternative system configurations, the performance of the proposed holonic MH system can be evaluated.

The conditions, simulation scenarios, and characteristics tested are similar for the two approaches such that a reliable comparison is obtained. The study is performed using different configurations for the job-shop model by varying the number of MH resources and jobs among the different tests, as well as during the same test. Throughout the simulation study, diverse types of changes are considered in the system. Processing times, new arriving jobs, their weights and due dates, the breakdown and recovery times for the MH resources are all considered random, and coming from specific distributions. The characteristics tested in the simulation study include: the quality of the solution delivered, the real-time scheduling ability, including the real-time response to changes in production orders, and the fault-tolerance and MH hardware reconfigurability capabilities.

12.5.1. Design of Experiments

The simulation study is carried out using a ten-machine job-shop problem served by three MH resources where two types of real-world potential events, such as new arriving jobs and MH resource breakdowns, are simulated. The holonic-based job evaluation, allocation and execution algorithms and the conventional scheduling approach are run every time the number of jobs in the system is increasing with one or more jobs, or there is a change in the number of available MH resources.

Bunches of replications for each of the two approaches are performed such that a

confidence interval on the difference between the expected responses of the two different systems can be built. The independence of replications is obtained by using different random numbers for each replication. The performance measure considered in the simulation study is the total completion time of all the jobs in the system. For each simulation replication, when a specified number of jobs (e.g., 100 jobs) are released, the system is not accepting any new jobs, and the simulation replication stops after all the jobs already released to the system are fully processed. The simulation logic for the holonic-based scheduling approach is presented in flowchart form in Figure 12.17, and that of the centralized scheduling approach is depicted in Figure 12.18 below.

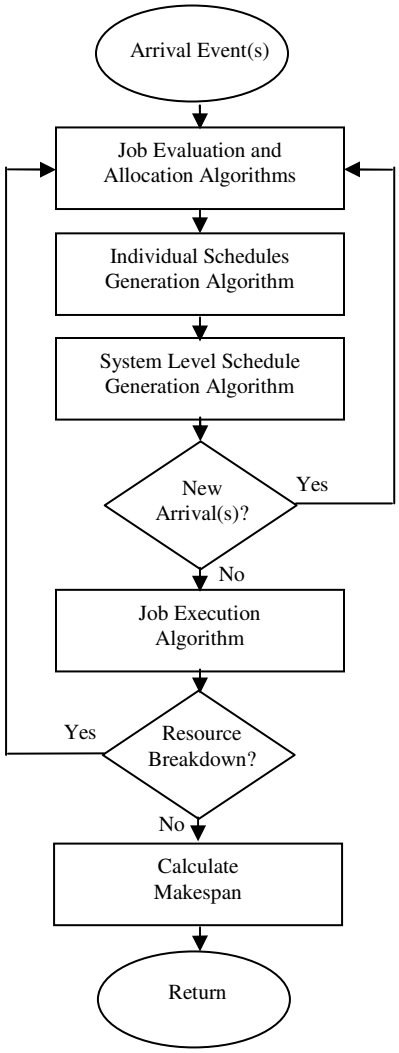

Figure 12.17. Simulation logic for holonic scheduling approach.

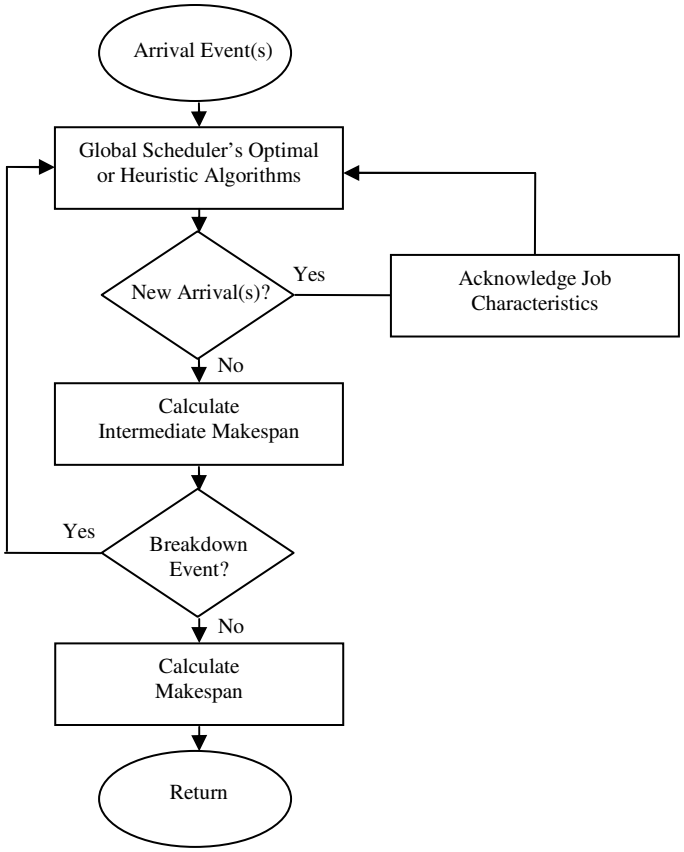

Figure 12.18. Simulation logic for centralized scheduling approach.

12.5.1.1. New Jobs

In the simulation study, the new arriving jobs are used to evaluate the capability of the proposed system to respond in real-time to changes in manufacturing orders. Each new coming job, denoted as an arrival event, has ten operations and the processing time for each operation is drawn from a uniform distribution *U(2, 9)*, while the time between job arrivals into the system follows an exponential distribution having a mean of ten, such that by using equation (12.38), jobs released at the same time can be obtained.

$$t = \lfloor 10(10EXP(-10x)) \rfloor \tag{12.38}$$

where, t represents the inter-arrival time taken as the nearest smaller integer resulted from the distribution, and x is a random number in the *[0, 1]* interval.

The actual release time of the jobs are obtained by accumulating the inter-arrival times

delivered by the exponential distribution as in equation (12.39), below:

$$r_j = r_{j-1} + t \tag{12.39}$$

where, r_j is the release time of job j, and t is the inter-arrival times define above. Initially $r_0 = 0$.

The weights of the jobs are coming from an exponential distribution which uses equation (12.40). In this way, virtually all the weights for the released jobs can be found on the *[1, 8]* interval with the regular job importance having the weight value, $w_j = 1$, being the most value obtained.

$$w_j = \lceil 8(EXP(-5x)) \rceil \tag{12.40}$$

where, w_j is the weight of job j, and, x is the random number.

The due dates for the coming jobs are calculated using equation (12.41) below:

$$d_j = r_j + f(TPT_j) \tag{12.41}$$

where, d_j is the due date for job j, r_j is the release time for job j, and $f(TPT_j)$ is the total processing time of the job adjusted using an allowance factor as in equation (12.42) below.

$$f(TPT) = (U(1.7, 2.3))TPT_j \tag{12.42}$$

where, $U(1.7, 2.3)$ is the uniform distribution between 1.7 and 2.3, and TPT_j is the total processing time for job j.

12.5.1.2. Changes in the Number of Available Material Handling Resources

Resource breakdowns, simulated using time periods coming from an exponential distribution, are used to assess the fault-tolerance and MH hardware reconfigurability capabilities of the system. Resource breakdowns are considered in the simulation study as unexpected disturbances in the manufacturing environment, where the amount of time between two failures for each resource, called *Mean Time between Failures* (MTBF), is drawn from exponential distributions given by equation (12.43) below. For the overall behavior of the system these breakdowns are similar to removing MH resources from the system. After the breakdowns, the simulated mean times to recover for each resource, called *Mean Time to Repair* (MTTR), is drawn also from exponential distributions using equation (12.44). For the overall behavior of the system this process is equivalent to adding new MH resources and making them available for the job evaluation and allocation processes. Another way to simulate breakdowns is to consider only one MTBF distribution and select randomly which MH resource is to be removed from the

system. In this case only one MTTR distribution is sufficient.

$$MTBF(t) = \lfloor 8(EXP(U(2,9))) \rfloor \quad (12.43)$$

where, *MTBF(t)* represents the time at which a resource breakdown occurs, and *U(2, 9)* is the uniform distribution of the processing times for all jobs in the system.

$$MTTR(t) = \lfloor 5(EXP(U(2,9))) \rfloor \quad (12.44)$$

where, *MTTR(t)* represents the time at which a resource is added back to the system, and *U(2, 9)* is the uniform distribution of the processing times for all jobs in the system.

12.5.2. Simulation Scenarios and Output Analysis

The paired-t approach (Law and Kelton, 2000) is used to build a confidence interval (CI) on the difference between the expected responses of the holonic scheduling approach and the global scheduling approach, results obtained by using the optimal EBFS algorithm and the three heuristics. For both approaches, after each replication which is performed using the same distribution, the completion time of the last job to be processed in the system is monitored. Since the paired-t method requires that observations drawn from each simulated approach be normally distributed and independent within that approach, bunches of replications are simulated and their mean is taken in consideration when constructing the confidence intervals.

The paired-t confidence interval method does require also that the number of observations obtained from one simulated system be equal to the number of observations obtained from the second simulated system, so an equal number of bunches composed of an equal number of replications are considered for both systems. To describe mathematically the paired-t method used in the output analysis the following notations are used:

- holonic scheduling approach = system S_1
- global scheduling approach = system S_2
- Job completion time heuristic = system S_3
- Machine completion time heuristic = system S_4
- Material handling resource constraint violation heuristic = system S_5
- $j_1 = 1, \ldots, n_1 = n$, is the number of bunches of replications for system S_1

- $j_2 = 1, \ldots, n_2 = n$, is the number of bunches of replications for system S_2
- $j_3 = 1, \ldots, n_3 = n$, is the number of bunches of replications for system S_3
- $j_4 = 1, \ldots, n_4 = n$, is the number of bunches of replications for system S_4
- $j_5 = 1, \ldots, n_5 = n$, is the number of bunches of replications for system S_5
- $k_1 = k$, is the number of replications in a bunch for system S_1
- $k_2 = k$, is the number of replications in a bunch for system S_2
- $k_3 = k$, is the number of replications in a bunch for system S_3
- $k_4 = k$, is the number of replications in a bunch for system S_4
- $k_5 = k$, is the number of replications in a bunch for system S_5
- $Z_{1ji}, j = 1, \ldots, n, i = 1, \ldots, k$, are the individual points obtained after the replication i of bunch j for system S_1
- $Z_{2ji}, j = 1, \ldots, n, i = 1, \ldots, k$, are the individual points obtained after the replication i of bunch j for system S_2
- $Z_{3ji}, j = 1, \ldots, n, i = 1, \ldots, k$, are the individual points obtained after the replication i of bunch j for system S_3
- $Z_{4ji}, j = 1, \ldots, n, i = 1, \ldots, k$, are the individual points obtained after the replication i of bunch j for system S_4
- $Z_{5ji}, j = 1, \ldots, n, i = 1, \ldots, k$, are the individual points obtained after the replication i of bunch j for system S_5

Since to apply the paired-t comparison approach, the number of bunches of replications for the two systems needs to be equal (i.e., $n_1 = n_2$; $n_1 = n_3$; $n_1 = n_4$; $n_1 = n_5$), the means for all the replications in a bunch for systems S_1 and for one of the systems S_2, S_3, S_4, and S_5 are given by the following two equations, respectively:

$$X_{(HS)j} = \frac{\sum_{i=1}^{k} Z_{(HS)ji}}{k} \qquad \text{for all } j = 1, \ldots, n \tag{12.45}$$

$$X_{(GS)j} = \frac{\sum_{i=1}^{k} Z_{(GS)ji}}{k} \qquad \text{for all } j = 1, \ldots, n \tag{12.46}$$

where, $X_{(HS)j}$ is the mean for all replications in a bunch for the holonic scheduling approach;

$X_{(GS)j}$ is the mean for all replications in a bunch for the global scheduling approach; $Z_{(HS)ji}$ are the individual points obtained after the replication *i* of bunch *j* for the holonic scheduling approach; and, $Z_{(GS)ji}$ are the individual points obtained after the replication *i* of bunch *j* for the global scheduling approach.

Then:

- $X_{(HS)j}$, $j = 1,\ldots, n$ are the IID observations coming from the holonic system
- $X_{(GS)j}$, $j = 1,\ldots, n$ are the IID observations coming from global scheduling system

By pairing $X_{(HS)j}$, and $X_{(GS)j}$ matched by using the same distributions for all *n* observations of the two approaches, and considering their difference as in equation (12.47) a new random variable that denotes the difference between the j^{th} observations in the two simulated approaches can be defined.

$$X_{(HS-GS)j} = X_{(HS)j} - X_{(GS)j} \tag{12.47}$$

The $X_{(HS\text{-}GS)j}$'s are IID random variables and the quantity for which the CI needs to be constructed is:

$$E\left(X_{(HS-2GS)j}\right) = \zeta = \mu_{HS} - \mu_{GS} \tag{12.48}$$

where, μ_{HS}, μ_{GS} are the expected means of the responses of the two systems, and ζ is the quantity for which the CI needs to be constructed.

The point estimators, sample mean and sample standard deviation for the $X_{(HS\text{-}GS)j}$'s are defined below in equations (12.49) and (12.50) below:

$$\overline{X}_{(HS-GS)} = \frac{\sum_{j=1}^{n} X_{(HS-GS)j}}{n} \tag{12.49}$$

$$S_{(HS-GS)} = \sqrt{\frac{\sum_{j=1}^{n}\left[X_{(HS-GS)j} - \overline{X}_{(HS-GS)j}\right]^2}{n-1}} \tag{12.50}$$

An unbiased estimator of $Var\left[\overline{X}_{(HS-GS)}\right]$ needed to form a $100(1 - \alpha)$ CI is given by equation (12.51) below:

$$Var\left[\overline{X}_{(HS-GS)}\right] = \frac{\sum_{j=1}^{n}\left[X_{(HS-GS)j} - \overline{X}_{(HS-GS)j}\right]^2}{n(n-1)} \tag{12.51}$$

The *100(1 - α)* CI on the expected difference between the responses of the two systems can then be defined as:

$$\overline{X}_{(HS-GS)} \pm t_{n-1,1-\alpha/2}\sqrt{Var\left[\overline{X}_{(HS-GS)}\right]} \tag{12.52}$$

where, $t_{n-1,\ 1-\alpha/2}$ is the upper $1-\alpha/2$ critical point for the *t* distribution with *n-1* degrees of freedom.

The same paired-t approach is used to compare the performance of the three heuristics designed for the scheduling needs of the Global Scheduler. The goal is to construct confidence intervals for the differences between the expected responses of system S_2, represented by the EBFS algorithm, and each of the systems S_3, S_4, and S_5, represented by the three heuristics. In the same time a pairwise comparison for the three heuristics is performed. The same equations (12.45 - 12.52) presented above, adapted for the analysis of the three heuristic,s are needed for these comparisons.

13. Experimental Part

To obtain a first image of the performance of the algorithms designed for the proposed MH system, several job-shop problems having variations in the number of jobs, machines, and MH resources are studied. To further evaluate the response of the holonic system, a series of job-shop problems, having a larger number of jobs and machines, are considered as pilot runs to verify the validity of the algorithms designed. Then, to obtain reliable information for the quality of the results given by the holonic scheduling system a simulation study of a ten-machine job-shop problem is carried out. For each of the three series of tests, the holonic scheduling algorithms, and the Global Scheduler's optimal and heuristic algorithms, as well as the LB_2 algorithm are implemented in C++ and run on a Pentium 3 PC having a 600 MHz CPU.

13.1. Characteristics Tested for the Holonic Scheduling System

The characteristics tested for the holonic system are the solution quality, the real-time scheduling ability, as well as the real-time response to changes in production orders or system architecture. The changes in production orders are related to releasing new jobs into the system, which replicate the behavior of a dynamic manufacturing environment, while the changes in system architecture include modifying the number of available MH resources in the system to assess the fault-tolerance and MH hardware reconfigurabilty capabilities.

The solution quality is assessed by comparing the results delivered by the holonic system with the theoretical LB_2 or LB_3 values and with the results given by the centralized scheduling approach. A comparison between the time needed to obtain a feasible solution using the holonic scheduling system and the time to deliver the optimal solution is provided for each of the tests. The global optimal algorithm embedded in the internal structure of the Global Scheduler is used for this comparison. Time counters embedded in the C++ code of the both approaches tested are used to record the time needed to obtain the feasible holonic approach and optimal solutions in each case in which the complexity of the problem permits.

The real-time response to changes in production orders is evaluated by monitoring the CPU time when running the decentralized holonic system approach. Changes in the manufacturing system architecture are considered to assess the capability of the holonic system

to respond in real-time to the potential breakdowns during system operation. True fault-tolerance is obtained when all the jobs affected by a breakdown can be re-allocated in acceptable time to the remaining resources, and the holonic re-scheduling process still gives an objective function value of good quality. MH hardware reconfigurability, or the capability of the holonic system to add or remove MH resources at any time during operation, is evaluated by the output of the re-scheduling process like above, and by monitoring the CPU time necessary for the system to perform the re-scheduling process.

13.2. Replicating Dynamic Manufacturing Environments

The first tests will evaluate the capability of the holonic system to respond to changes characteristic to dynamic manufacturing environments. The new jobs that come into the system have their release dates unknown at the start of operations, and thus, are mimicking a dynamic manufacturing environment. First example is conducted using a six-machine, eight jobs, job-shop model ($J6 \mid\mid C_{max}$) served by two MH resources. The objective function to be optimized is considered the overall completion time or the makespan. A temporary storage (TS) is used to speed up the flow time of jobs in the system. For simplicity, all MH L/U operations are considered to last only one unit time. The new jobs that come into the system are reproducing the actual operations in a dynamic manufacturing environment. The data related to the eight jobs is presented in Table 13.1.

Table 13.1. Characteristics, sequence of operations and processing times for the eight jobs.

Job j	*Release Date* r_j	*Weight* w_j	*Due Date* d_j	***Sequence of Operations / Processing Times***											
				1		*2*		*3*		*4*		*5*		***6***	
1	0	1	55	1	7	2	4	3	2	6	2	5	3	4	5
2	0	1	58	1	3	4	2	3	2	5	4	6	2	2	7
3	0	3	51	4	6	2	4	5	1	3	2	6	4	1	6
4	3	5	48	3	2	4	4	6	6	1	1	5	3	2	1
5	6	1	56	2	7	5	3	3	2	6	5	4	2	1	4
6	6	2	56	1	3	4	6	5	6	2	3	6	2	3	1
7	10	3	58	2	2	3	4	5	8	6	1	4	3	1	5
8	10	4	55	5	4	4	6	1	6	3	9	6	5	2	6

The job assignment to the processing machines that disregards MH operations is

performed using the Shifting Bottleneck Heuristic (SBH) and Local Search Heuristic (LSH) techniques, but it could be done using any other method that gives a feasible solution including agent or holonic-based approaches. These job assignments are repeated each time something changes in the system (i.e., a new job is entering in the system). All the operations not started when changes occur are considered for rescheduling. The release dates for the jobs not yet in processing are considered not known in advance. The jobs enter the system at their release dates, and a new schedule that includes them is generated. If these job release dates were considered known from the start, the schedule that includes these jobs would be predictive, not reactive as intended. The resulting reactive schedule of processing jobs for the six-machine job-shop problem that disregards MH operations is presented in Figure 13.1. The intermediate job schedules generated at the release dates of the jobs entering the system (*T = 0*, *T = 3*, and *T = 6)* are presented in Appendices C.1. - C.4.

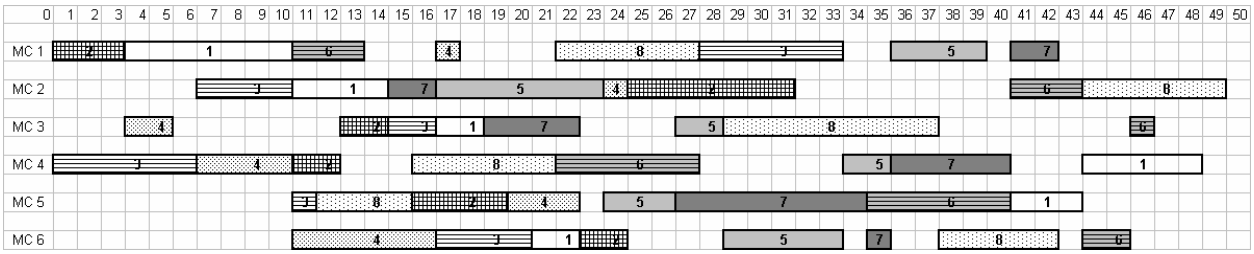

Figure 13.1. Job schedule when MH operations are not considered.

13.2.1. Lower Bounds on Schedule Makespan

Using the algorithm for calculating LB_3 presented in a previous chapter, a LB for the schedule makespan is found to be equal to *70*. The output of the LB_3 algorithm is a schedule having the MH operations inserted between processing operations that takes into account the operation precedence constraints on both machines and jobs. The resulting schedule, presented in Figure 13.2 below, is not practical since it uses more than the available two MH resources in several time periods. For example, the schedule uses five MH resources at time *T = 14*, four MH resources at time *T = 39*, and three MH resources in another eight situations (*T = 13, 22, 23, 31, 38, 54, 61,* and *62*). The other intermediate LBs are calculated whenever new jobs enter into the

system (*T = 0*, *T = 3*, and *T = 6)* and are presented graphically in Appendices C.1. - C.4.

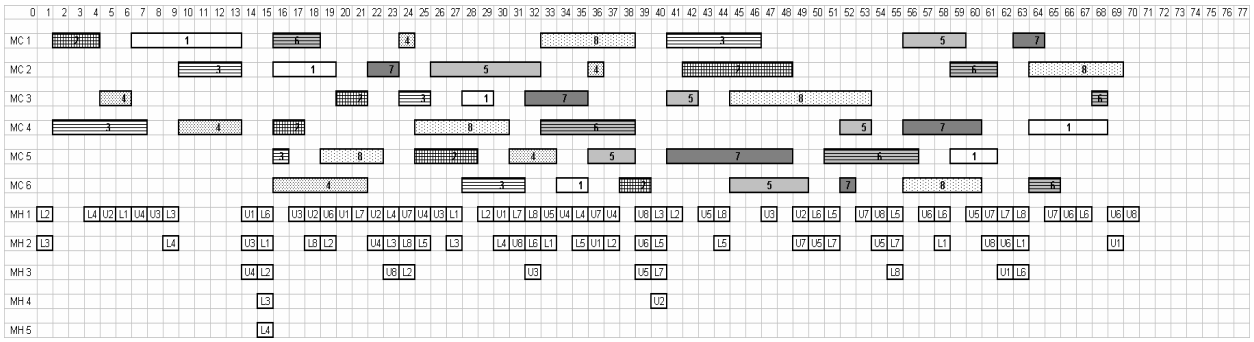

Figure 13.2. LB_3 on the schedule makespan when considering MH operations.

To find an improved LB, such as LB_4, the L/U operations in those particular ten time periods should be re-arranged and some of them move to the left. As stated before, this process leads to a large number of possible schedules. There is no information about the makespan of such an improved schedule. However, due to the fact that several operations are moved to the right the resulting schedule makespan is likely to increase.

13.2.2. Assignment of Jobs to the Material Handling Resources

Using the Order and Material Handling Holons job evaluation and allocation algorithms presented before, the assignment of jobs to the MH resources is presented in Table 13.2.

Table 13.2. Holonic job assignments for the six-machine problem.

Time	*Job*	*Order Holon*		*Material Handling Holon*			
				MH-ECT		*MH-OP*	
		OH-ECT	*OH-OP*	*MHH 1*	*MHH 2*	*MHH 1*	*MHH 2*
0	1	39	8.51	**43**	44	**12.51**	13.51
	2	40	8.28	**40**	40	**8.28**	8.28
	3	35	24.71	35	**35**	24.71	**24.71**
3	4	34	42.50	39	**38**	66.75	**69.75**
6	5	49	10.50	**53**	53	**14.50**	22.50
	6	49	21.00	56	**49**	33.00	**21.00**
10	7	55	34.13	67	**56**	66.13	**35.12**
	8	61	53.24	**65**	70	**63.24**	81.24

MHH1 will execute jobs *1*, *2*, *5*, and *8*, while *MHH2* will execute jobs *3*, *4*, *6*, and *7*. These assignments are presented in bold in Table 13.2. Inter-Material Handling Holon cooperation will change the Material Handling Holon that executes a few of the operations of these jobs with the advantage of achieving better performance measures for the system.

13.2.3. Generation of the System Level Schedule

Using the above MH job assignment and the System Level Schedule generation algorithm the resultant system schedule that takes in account also the MH operations for the six-machine job-shop problem is presented in Figure 13.3.

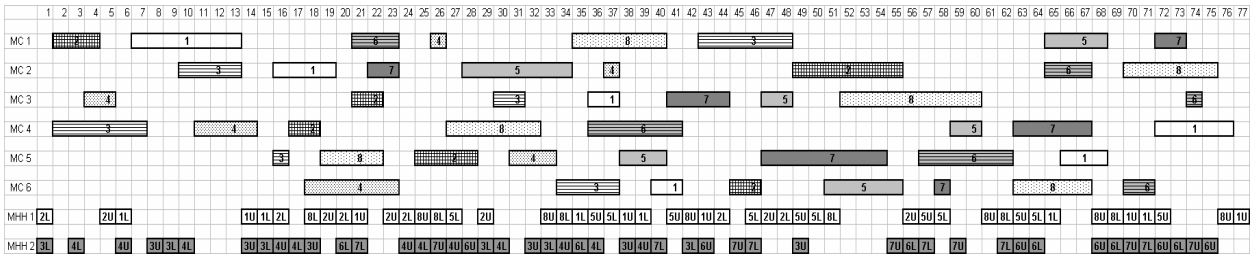

Figure 13.3. Assignment of MH operations obtained using the holonic approach.

In the figure the operations performed by *MHH1* are presented in white, while the operations performed by *MHH2* are presented in gray. There is an increase from *49* to *77* in the makespan from the initial job schedule to the System Level Schedule due to the MH operations.

13.2.4. Inter-Material Handling Holons Cooperation Improvements to the Initial System Level Schedule

The final schedule for processing the eight jobs that takes in account the necessary MH operations obtained after using the holonic job evaluation and allocation algorithms and Inter-Material Handling Holon cooperation is presented in Figure 13.4. It can be observed that *MHH1* executes five operations from the jobs initially assigned to *MHH2*, and *MHH2* executes four operations from the ones initially assigned to *MHH1*. The Inter-Material Handling Holons

cooperation mechanism reduces the schedule makespan from *77* to *75*. This result is close to the theoretical makespan of *70*, presented above. The intermediate System Level Schedules generated at times $T = 0$, $T = 3$, and $T = 6$, are presented in Appendices C.1. - C.4.

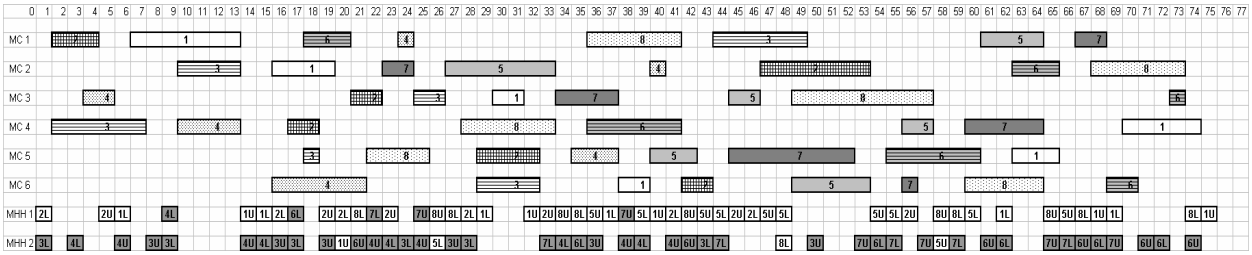

Figure 13.4. Final schedule when considering MH operations.

13.2.5. Comparison of the Initial System Level Schedule with All Material Handling Resources Possible Schedules

By assigning an equal workload of four jobs to each MH resource and considering all the possible assignments, a number of *70* possible combinations are obtained. This number of combinations is further reduced to *35* because the remaining *35* are the same when reversing the order of the two MH resources. Table 13.3 presents all these possible assignments, and their makespan values for all eight jobs. It can be seen that the base assignment (MHH_1 assigned to execute jobs *1*, *2*, *5*, and *8*; MHH_2 assigned to execute jobs *3*, *4*, *6*, and *7*) obtained using the holonic scheduling approach (number *12* in Table 13.3, shown in bold) is one of the best among the *35* possible schedules, and that the final schedule obtained by using also the Inter-Material Handling Holon cooperation process further improves this assignment. All the completion times and makespan values better than those of assignment *12* are presented in *italics.*

Table 13.4 considers some other performance measures, besides makespan, used in scheduling literature (maximum tardiness, sum of weighted tardiness, and number of tardy jobs). Once again, assignment *12*, resulted also using the holonic approach, is one of the best in the *35* possible assignments. All the performance measures better than those calculated using assignment *12* are presented in *italics.*

Table 13.3. All possible 4-job assignments and the completion times resulted for all eight jobs.

Assignment Number	*Assignment of Jobs to Material Handling Resources*		*Completion Time for the Jobs* C_j								*Makespan* C_{mac}
	MH Resource 1	*MH Resource 2*	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	
1	1 - 2 - 3 - 4	5 - 6 - 7 - 8	82	63	*48*	42	*71*	80	79	81	82
2	1 - 2 - 3 - 5	4 - 6 - 7 - 8	77	61	52	41	*70*	81	74	80	81
3	1 - 2 - 3 - 6	4 - 5 - 7 - 8	80	*55*	*48*	45	*71*	78	77	79	80
4	1 - 2 - 3 - 7	4 - 5 - 6 - 8	79	*55*	*48*	40	*69*	79	74	80	80
5	1 - 2 - 3 - 8	4 - 5 - 6 - 7	*76*	56	49	39	*71*	79	78	81	81
6	1 - 2 - 4 - 5	3 - 6 - 7 - 8	*75*	58	52	42	*67*	76	75	77	77
7	1 - 2 - 4 - 6	3 - 5 - 7 - 8	78	56	52	41	*66*	76	*72*	78	78
8	1 - 2 - 4 - 7	3 - 5 - 6 - 8	79	58	54	44	*69*	79	74	81	81
9	1 - 2 - 4 - 8	3 - 5 - 6 - 7	86	56	50	46	*69*	86	76	88	88
10	1 - 2 - 5 - 6	3 - 4 - 7 - 8	78	*55*	50	41	*68*	76	75	77	78
11	1 - 2 - 5 - 7	3 - 4 - 6 - 8	78	*55*	50	41	73	78	75	77	78
12	**1 - 2 - 5 - 8**	**3 - 4 - 6 - 7**	**77**	**56**	**49**	**39**	**72**	**75**	**74**	**76**	**77**
13	1 - 2 - 6 - 7	3 - 4 - 5 - 8	77	57	50	39	*66*	76	*72*	*74*	77
14	1 - 2 - 6 - 8	3 - 4 - 5 - 7	81	*55*	54	43	73	79	76	78	81
15	1 - 2 - 7 - 8	3 - 4 - 5 - 6	79	*54*	52	43	*68*	77	74	80	80
16	1 - 3 - 4 - 5	2 - 6 - 7 - 8	89	59	55	45	73	84	79	81	89
17	1 - 3 - 4 - 6	2 - 5 - 7 - 8	82	*55*	52	43	*70*	81	80	82	82
18	1 - 3 - 4 - 7	2 - 5 - 6 - 8	80	56	52	43	*69*	81	74	80	81
19	1 - 3 - 4 - 8	2 - 5 - 6 - 7	84	60	55	46	73	82	78	82	84
20	1 - 3 - 5 - 6	2 - 4 - 7 - 8	79	58	51	44	*67*	77	*72*	78	79
21	1 - 3 - 5 - 7	2 - 4 - 6 - 8	82	58	55	48	*70*	80	77	82	82
22	1 - 3 - 5 - 8	2 - 4 - 6 - 7	*76*	*53*	49	39	*64*	76	*71*	78	78
23	1 - 3 - 6 - 7	2 - 4 - 5 - 8	79	57	51	41	*66*	77	*71*	77	79
24	1 - 3 - 6 - 8	2 - 4 - 5 - 7	85	59	50	42	*69*	80	76	82	85
25	1 - 3 - 7 - 8	2 - 4 - 5 - 6	81	58	51	42	*68*	78	*72*	80	81
26	1 - 4 - 5 - 6	2 - 3 - 7 - 8	83	*54*	50	46	*70*	81	76	82	83
27	1 - 4 - 5 - 7	2 - 3 - 6 - 8	81	*54*	50	45	*70*	79	77	81	81
28	1 - 4 - 5 - 8	2 - 3 - 6 - 7	77	*54*	53	41	72	79	74	80	80
29	1 - 4 - 6 - 7	2 - 3 - 5 - 8	81	*54*	53	44	*70*	80	76	80	81
30	1 - 4 - 6 - 8	2 - 3 - 5 - 7	83	*54*	53	45	*70*	81	75	80	83
31	1 - 4 - 7 - 8	2 - 3 - 5 - 6	80	57	58	47	*65*	79	*70*	82	82
32	1 - 5 - 6 - 7	2 - 3 - 4 - 8	86	58	55	45	73	85	81	85	86
33	1 - 5 - 6 - 8	2 - 3 - 4 - 7	84	57	56	45	*71*	82	75	83	84
34	1 - 5 - 7 - 8	2 - 3 - 4 - 6	80	57	51	43	*67*	77	75	79	80
35	1 - 6 - 7 - 8	2 - 3 - 4 - 5	83	64	53	44	*68*	82	78	84	84
Holonic Approach	**Base Assignment**										
	1 - 2 - 5 - 8	**3 - 4 - 6 - 7**	**75**	**56**	**50**	**41**	**66**	**74**	**69**	**74**	**75**

13.2.6. Influence of Material Handling Operations on the Schedule Performance Measures

The results obtained clearly indicate that MH does add additional time to the processing cycle, in fact more than expected. Table 13.5 presents the influence of MH operations on the makespan of the schedule. The *Expected* C_{max} is the traditional way of taking into account the MH operations, and its value is obtained from the schedule makespan when MH operations are

not considered with the addition of the times for each L/U operation. This equation has been defined before as the intuitive LB_1 on the schedule makespan.

$$Expected\ C_{max} = C_{max} + \sum_{k=1}^{K_m} \left((LT_k)_m + (UT_k)_m \right) \tag{13.1}$$

where, C_{max} is the makespan value when MH operations are ignored; $(LT_k)_m$, $(UT_k)_m$ are the times for the L/U operations of job k on machine m; m represents the machine that gives the schedule makespan; and, K_m is the number of jobs performed on machine m.

Table 13.4. All possible 4-job assignments and values of other scheduling performance measures

Assignment Number	*Assignment of Jobs to Material Handling Resources*		*Makespan*	*Maximum Tardiness*	*Sum of Weighted Tardiness*	*Number of Tardy Jobs*
	MH Resource 1	*MH Resource 2*	C_{mac}	T_{max}	$\Sigma w_j T_j$	U_j
1	1 - 2 - 3 - 4	5 - 6 - 7 - 8	82	27	269	6
2	1 - 2 - 3 - 5	4 - 6 - 7 - 8	81	25	247	7
3	1 - 2 - 3 - 6	4 - 5 - 7 - 8	80	25	244	5
4	1 - 2 - 3 - 7	4 - 5 - 6 - 8	80	25	238	5
5	1 - 2 - 3 - 8	4 - 5 - 6 - 7	81	26	253	5
6	1 - 2 - 4 - 5	3 - 6 - 7 - 8	77	22	220	6
7	1 - 2 - 4 - 6	3 - 5 - 7 - 8	78	23	219	5
8	1 - 2 - 4 - 7	3 - 5 - 6 - 8	81	26	251	6
9	1 - 2 - 4 - 8	3 - 5 - 6 - 7	88	33	297	5
10	1 - 2 - 5 - 6	3 - 4 - 7 - 8	78	23	221	5
11	1 - 2 - 5 - 7	3 - 4 - 6 - 8	78	23	230	5
12	**1 - 2 - 5 - 8**	**3 - 4 - 6 - 7**	**77**	**22**	**215**	**5**
13	1 - 2 - 6 - 7	3 - 4 - 5 - 8	77	22	*197*	5
14	1 - 2 - 6 - 8	3 - 4 - 5 - 7	81	26	251	6
15	1 - 2 - 7 - 8	3 - 4 - 5 - 6	80	25	236	6
16	1 - 3 - 4 - 5	2 - 6 - 7 - 8	89	34	294	7
17	1 - 3 - 4 - 6	2 - 5 - 7 - 8	82	27	275	6
18	1 - 3 - 4 - 7	2 - 5 - 6 - 8	81	25	246	6
19	1 - 3 - 4 - 8	2 - 5 - 6 - 7	84	29	287	7
20	1 - 3 - 5 - 6	2 - 4 - 7 - 8	79	24	218	5
21	1 - 3 - 5 - 7	2 - 4 - 6 - 8	82	27	273	6
22	1 - 3 - 5 - 8	2 - 4 - 6 - 7	78	23	*207*	5
23	1 - 3 - 6 - 7	2 - 4 - 5 - 8	79	24	228	5
24	1 - 3 - 6 - 8	2 - 4 - 5 - 7	85	30	253	6
25	1 - 3 - 7 - 8	2 - 4 - 5 - 6	81	26	231	5
26	1 - 4 - 5 - 6	2 - 3 - 7 - 8	83	28	261	5
27	1 - 4 - 5 - 7	2 - 3 - 6 - 8	81	26	254	5
28	1 - 4 - 5 - 8	2 - 3 - 6 - 7	80	25	245	6
29	1 - 4 - 6 - 7	2 - 3 - 5 - 8	81	26	255	6
30	1 - 4 - 6 - 8	2 - 3 - 5 - 7	83	28	256	6
31	1 - 4 - 7 - 8	2 - 3 - 5 - 6	82	27	252	6
32	1 - 5 - 6 - 7	2 - 3 - 4 - 8	86	31	314	6
33	1 - 5 - 6 - 8	2 - 3 - 4 - 7	84	29	281	6
34	1 - 5 - 7 - 8	2 - 3 - 4 - 6	80	25	232	5
35	1 - 6 - 7 - 8	2 - 3 - 4 - 5	84	29	287	7
Holonic Approach	**Base Assignment**					
	1 - 2 - 5 - 8	**3 - 4 - 6 - 7**	**75**	**20**	**182**	**5**

The *Actual* C_{max} is taken from the schedule obtained using the holonic approach which considers MH as separate operations. The *Actual* C_{max} is in all cases greater than the *Expected* C_{max}, and has a tendency of increasing with the increase in the number of jobs in the system. The lower value in the difference between *Actual* C_{max} and *Expected* C_{max} in the eight-job case compared to the six-job case is explained by a better positioning of the last two jobs in the already existing six jobs schedule. It is likely that the difference between *Actual* C_{max} and *Expected* C_{max} will increase with an additional increase in the number of jobs.

Table 13.5. Influence of MH operations on the schedule makespan.

Number of jobs in system	C_{max} *(MH not considered)*	*Expected* C_{max} *(MH added at the completion of jobs)*	*Lower Bound on* C_{max}			*Actual* C_{max} *(MH considered as separate operation)*	*Actual* C_{max} *- Expected* C_{max}
			LB_1	LB_2	LB_3		
3 jobs	28	34	34	44	44	44	10
4 jobs	31	39	39	49	49	51	12
6 jobs	38	50	50	61	61	67	17
8 jobs	49	65	65	70	70	75	10

Moreover, the results show that the traditional way of taking in account MH operations, by adding the L/U times to the completion times of the operations on each machine, is not giving a correct image of the influence of MH operations on the schedule makespan. The calculated LB_3 is in all cases greater than the *Expected* C_{max}, which shows that the *Expected* C_{max} is inaccurate. The values of the *Actual* C_{max} are close to those of the calculated LB_3. Considering that the calculated LB_3 might not be always attainable the results given by the holonic approach are not only feasible and obtained in a short amount of time, but also of good quality.

A comparison of the performance of these three types of schedules when all eight jobs are completed, performed for due date related objectives is presented in the next three tables. Table 13.6 corresponds to the case in which MH operations are totally ignored, and shows that considering the due dates as presented in Table 13.1, there are no late jobs. The second one, Table 13.7, presents the expected lateness and tardiness corresponding to the second situation in Table 13.5. The results presented show that three new tardy jobs appeared only by adding MH operations time to the completion time of the jobs and without considering possible schedule conflicts or unavailability of MH resources.

Table 13.6. Job lateness and tardiness values obtained when MH operation times are not considered.

Job j	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*
Completion Time C_j	48	31	33	24	39	46	42	49
Due Date d_j	55	58	51	48	56	56	58	55
Lateness L_j	-7	-27	-18	-24	-17	-10	-16	-6
Tardiness T_j	0	0	0	0	0	0	0	0

Table 13.7. Expected job lateness and tardiness values obtained by adding MH operation times to the job's completion times.

Job j	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*
Completion Time C_j	64	47	49	40	55	62	58	65
Due Date d_j	55	58	51	48	56	56	58	55
Expected Lateness EL_j	9	-9	-2	-8	-1	6	0	10
Expected Tardiness ET_j	9	0	0	0	0	6	0	10

The last of the three tables, Table 13.8, presents the due date related objectives for the real-world situation when MH operations should be taken in consideration. When considering all its steps the holonic scheduling approach gives a better solution than any other assignment of four jobs per each MH resource. However, the performance of the due date related objectives is decreasing considerably compared to the first two cases in which MH operations were ignored or just had their time unit values added to the completion times for the processing jobs. There are five late jobs and the values for the actual tardiness had increased also for the first three late jobs found when calculating the expected tardiness.

Table 13.8. Actual job lateness and tardiness values obtained using holonic scheduling when considering MH operations.

Job j	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*
Completion Time C_j	75	56	49	41	66	74	69	74
Due Date d_j	55	58	51	48	56	56	58	55
Actual Lateness AL_j	20	-2	-2	-7	10	18	11	19
Actual Tardiness AT_j	20	0	0	0	10	18	11	19

13.2.7. Global Scheduler's Optimal and Heuristic Algorithms Solutions

The results obtained by running all four global scheduling algorithms and the solution given by the holonic approach, together with the CPU time to obtain those results, are presented in Table 13.9. In the solution given by the holonic approach, the MH resources are assigned a predefined number of jobs and need to perform all the L/U operations necessary for these jobs.

Table 13.9. Comparison of the makespan values and CPU times obtained using the holonic approach and the Global Scheduler's four algorithms.

Algorithm	*LB_3*	*Holonic Approach*	*EBFS Optimal Algorithm*	*JCTH*	*MCTH*	*MH-RCVH*
Makespan or Theoretical Value	70	75	73	75	80	77
CPU Time (seconds)	0.3	6.8	28.4	0.1	0.1	0.1

The difference between the solutions coming from the Global Scheduler's algorithms and the one given by the holonic approach comes from the assignment of jobs to MH resources. While in the Global Scheduler's algorithms there is no specific assignment of jobs to MH resources, in the holonic approach each job entering into the system is assigned to a particular MH resource. The holonic assignment approach (Figure 13.4) is giving a more clear image of the MH operations needed to be performed by a specific MH resource, since a particular MH resource needs only to know that it has to perform the L/U operations associated with, let's say jobs *1*, *2*, *5*, and *8*, from the all pool of jobs *1* to *8* that are to be processed in the system.

It can be seen from Table 13.9 that the first heuristic, JCTH, that considers the total completion time of the jobs that need to be sorted, is giving the same solution as the holonic approach having a makespan value of *75*, very close to the optimal one obtained using the optimal EBFS algorithm, which has a makespan value of *73* (Figure 13.5). The results given by the other two heuristics are also good taking in consideration the real-time response obtained.

Using the time counters embedded in the C++ code, the CPU time can be computed for all the algorithms designed. Except for the optimal EBFS algorithm, all others algorithms designed provide their solutions in less than a few seconds. In the case of the holonic approach

since there are several algorithms that need to be run to obtain the final schedule, the total CPU time is obtained by adding the individual CPU times for all the job evaluation, allocation, and execution algorithms, and the time needed to read/write the information associated with each new evaluation and allocation process.

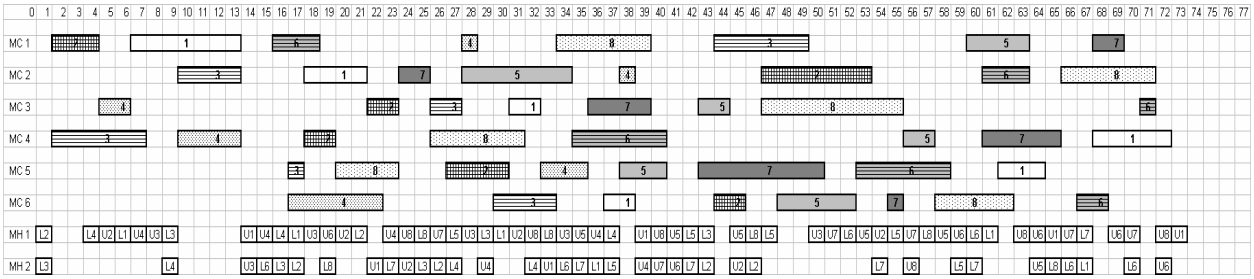

Figure 13.5. Optimal schedule determined using the EBFS algorithm.

The tree search to obtain the optimal solution by using the EBFS algorithm is sketched in Figure 13.6. Due to the combinatorial complexity of the problem only the branches of the selected pairs of MH operations that are on the path to the optimal solution are presented. The other branches that are fathomed during the execution of the algorithm steps are not considered in the figure.

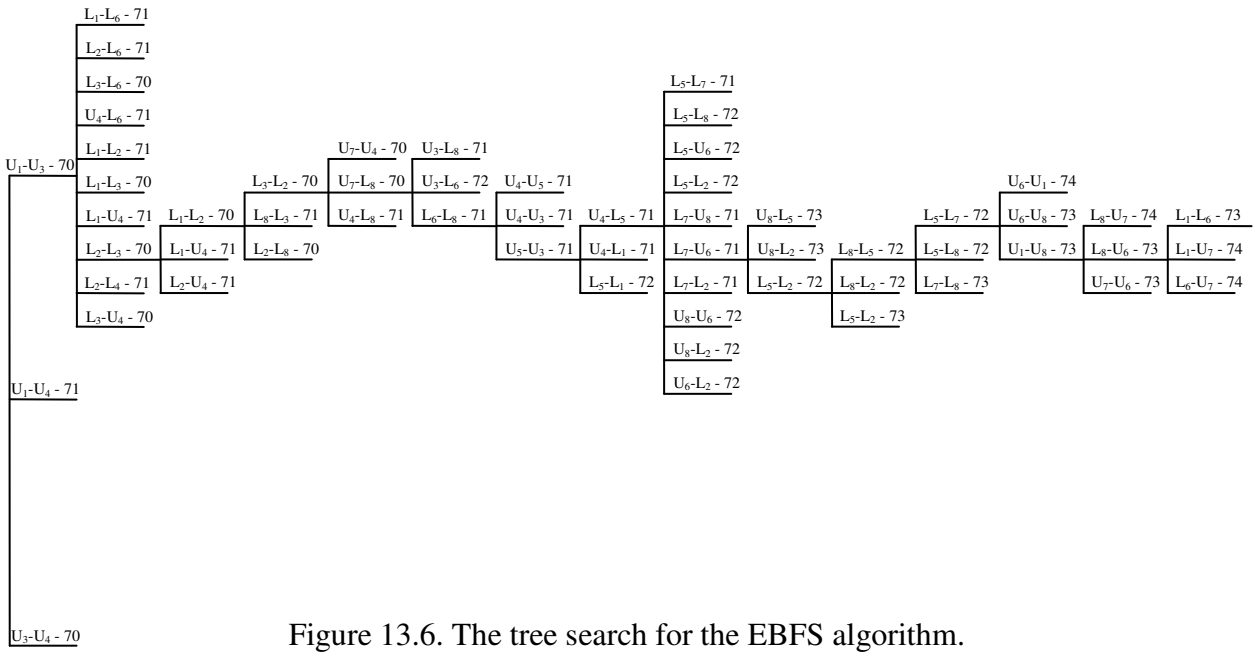

Figure 13.6. The tree search for the EBFS algorithm.

13.3. Varying the Number of Available Material Handling Resources in the Holonic System

Resource breakdowns and modifying the number of MH resources are considered in the second test with the objective of assessing the fault-tolerance and MH hardware reconfigurability capabilities of the holonic system. The first example designed to evaluate the capability of the proposed holonic system to recover in real-time from a resource breakdown is conducted using a four-machine ten-job job-shop model ($J4 \mid\mid C_{max}$) served by three MH resources. The objective function to be optimized is considered the overall completion time or the makespan. Just like in the previous example a TS is used to speed up the flow time of jobs in the system and all MH operations are considered to have one unit processing time. The data related to the ten jobs is presented in Table 13.10.

Table 13.10. Characteristics, sequence of operations and processing times for the ten jobs.

Job j	*Release Date* r_j	*Weight* w_j	*Due Date* d_j	***Sequence of Operations / Processing Times***							
				1		*2*		*3*		***4***	
1	0	2	40	1	3	2	8	3	5	4	2
2	0	3	60	3	6	2	5	4	2	1	5
3	0	4	60	3	5	1	2	4	3	2	3
4	0	1	60	2	8	1	6	3	6	4	4
5	0	1	40	2	2	4	7	1	7	3	6
6	0	2	50	1	4	3	3	2	7	4	6
7	0	5	50	2	3	1	3	3	2	4	6
8	0	2	60	4	5	3	2	1	8	2	2
9	0	3	60	1	6	3	4	2	3	4	1
10	0	1	50	4	3	2	4	1	1	3	2

Just as in the first example, the job assignment to the processing machines that disregards MH operations is performed using the SBH and LSH techniques, but it could be done using any other method that gives a feasible solution including agent or holonic-based approaches. The resulting schedule of processing jobs for the four-machine job-shop problem that disregards MH operations has a makespan value of *45* and is presented in Figure 13.7.

A System Level Schedule for the MH resources is generated each time something changes in the system (i.e., a resource is removed from. or added to the architecture). All the MH

operations not started at the moment when an add/remove process takes place are considered for rescheduling.

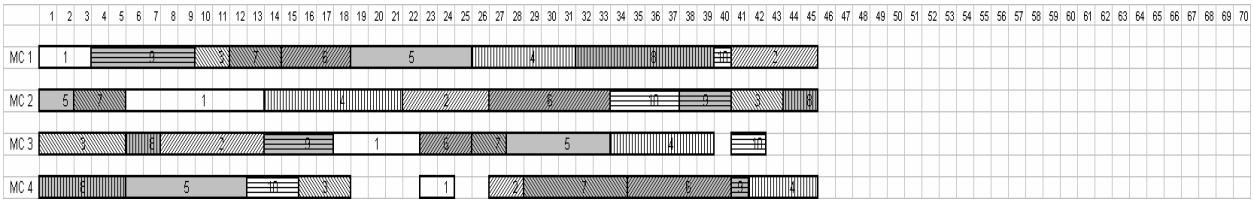

Figure 13.7. Job schedule when MH operations are not considered.

13.3.1. Lower Bounds on Schedule Makespan

The calculated LB_2 (Figure 13.8) on the schedule makespan is found to be equal to *66*, one unit more than the intuitive LB_1 which has a value of *65*. However, there is a need for four MH resources at time $T = 0$ to get this value, so LB_2 may not be achievable. The processing jobs schedule shows the existence of two CP_2 and one CP_3. Therefore by using equations (12.14) and (12.15), the improved LB_3 is found to be equal to LB_2, having a value of *66*. The actual LB_4 is not calculated for this problem since the number of possible combinations is large, and these kinds of problems cannot be solved in real-time. If using only two MH resources available from the start, both delays type 2 and type 3 would occur at least at times $T = 36$, *37* and $T = 56$, *57*, respectively.

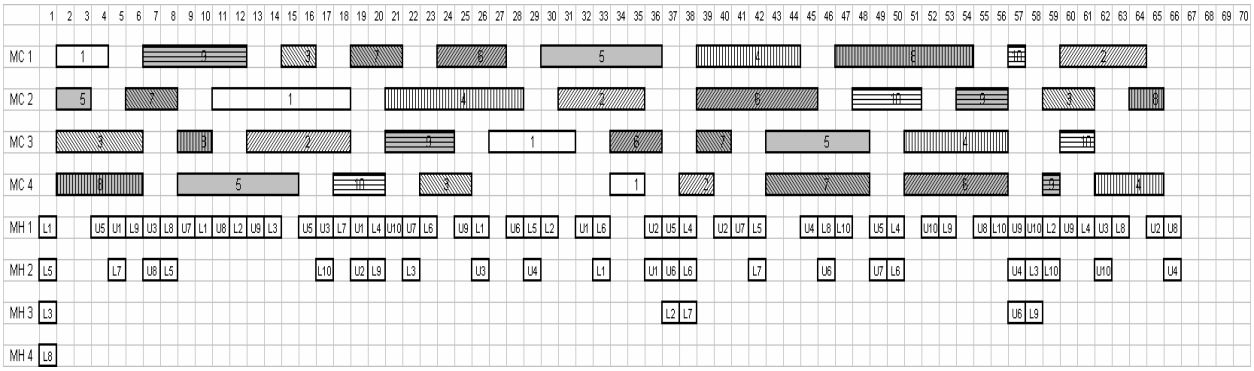

Figure 13.8. LB_2 on the makespan when considering the MH operations.

13.3.2. Initial Job Assignment to the Material Handling Resources

Using the Order and Material Handling Holons job evaluation and allocation algorithms presented before, the assignment of jobs to the three MH resources is presented in Table 13.11. Initially, jobs *1*, *5*, and *10* are assigned to MHH_1, jobs *2*, *3*, *4*, and *6* are assigned to MHH_2, and jobs *7*, *8*, and *9* are assigned to MHH_3. These assignments are presented in bold in Table 13.11.

Table 13.11. Holonic job assignment for the four machine problem.

Time	*Job*	*Order Holon*		*Material Handling Holon*					
				MH-ECT			*MH-OP*		
		OH-ECT	*OH-OP*	*MHH 1*	*MHH 2*	*MHH 3*	*MHH 1*	*MHH 2*	*MHH 3*
0	1	32	12.80	**32**	32	32	**12.80**	12.80	12.80
	2	53	21.20	53	**53**	59	21.20	**21.20**	43.20
	3	51	27.20	51	**51**	51	27.20	**27.20**	27.20
	4	53	7.07	63	**55**	57	14.07	**13.07**	12.07
	5	41	8.20	**41**	41	43	**8.20**	8.20	10.20
	6	48	15.36	54	**50**	56	23.36	**19.36**	39.36
	7	42	33.60	48	44	**42**	49.60	43.60	**33.60**
	8	51	13.60	51	55	**51**	13.60	21.60	**31.60**
	9	49	19.60	53	53	**49**	29.60	35.60	**19.60**
	10	50	8.00	**52**	54	54	**10.00**	22.00	16.00

13.3.3. Generation of the Initial System Level Schedule

Using the above MH job assignment and the System Level Schedule generation algorithm the resultant system schedule that takes into account the MH operations for the four-machine ten-job job-shop problem has a makespan value of *69* and is presented in Figure 13.9.

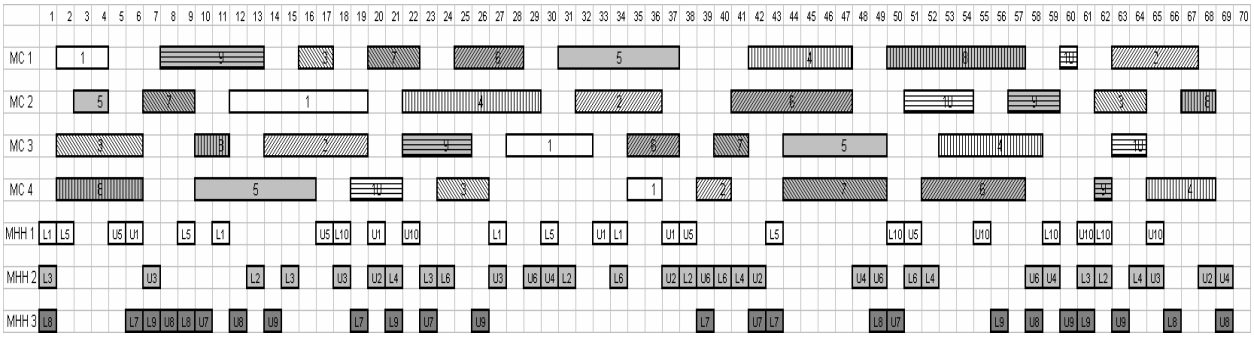

Figure 13.9. Assignment of MH operations obtained using the holonic approach.

In the above figure the operations performed by MHH_1 are presented in white, while the operations performed by MHH_2 and MHH_3 are presented in light and dark gray, respectively. There is an increase from *45* to *69* in the value of the schedule makespan from the initial job schedule to the System Level Schedule due to the insertion of the MH operations.

13.3.4. Inter-Material Handling Holons Cooperation Improvements to the Initial System Level Schedule

The initial schedule, before the breakdown occurs, for processing the eight jobs that takes in account the necessary MH operations, obtained after using the holonic evaluation and allocation algorithms and Inter-Material Handling Holon cooperation is presented in Figure 13.10 below:

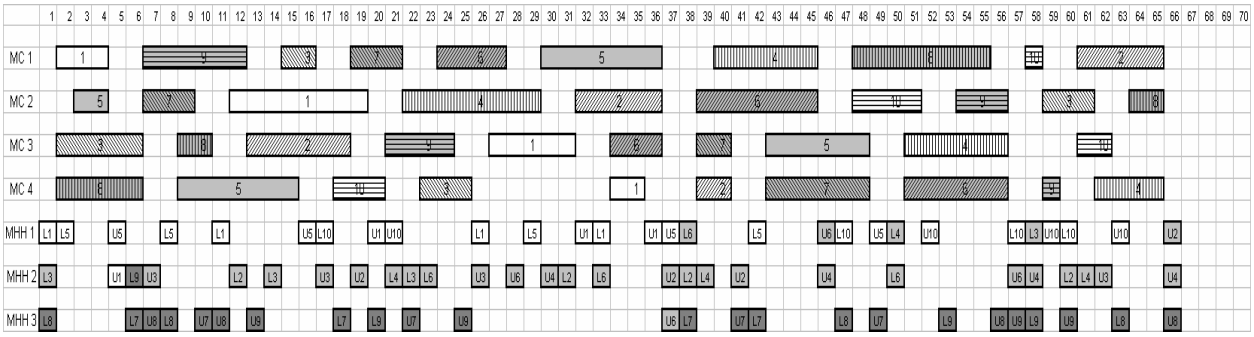

Figure 13.10. Initial schedule, before breakdown occurs, when considering MH operations.

It can be observed that MHH_1 is scheduled to perform five operations from the jobs initially assigned to MHH_2, while MHH_2 is scheduled to perform one operation from the ones initially assigned to both MHH_1 and MHH_3. Finally MHH_3 is scheduled to perform one operation from the ones initially assigned to MHH_2. This Inter-Material Handling Holons cooperation reduces the schedule makespan from *69* to *66*. This value is equal to the theoretical LB_3 calculated before, so this initial assignment of MH resources, before the breakdown occurs, is optimal. The theoretical value of the LB_3 is achieved even the assignment uses three MH resources compared to four used in the calculation of LB_3.

13.3.5. System Response to a Breakdown Occurrence

The breakdown of MHH_2 is simulated at time $T = 20$, and the jobs assigned to this Material Handling Holon are re-announced to the remaining two Material Handling Holons as available for sending L/U and transport offers. The loading operation of job *4* will not be performed at time $T = 20$, as previous scheduled because of the breakdown. The already completed and in-process operations performed by the other two MH resources are not considered for re-scheduling. All other MH operations are considered for re-scheduling. Using once again the Order and Material Handling Holons job evaluation and allocation algorithms the jobs initially assigned to MHH_2 are re-assigned as presented in Table 13.12. MHH_1 is now scheduled to perform also jobs *2* and *3*, while MHH_3 is now scheduled to perform another two jobs, *4* and *6*.

Table 13.12. Assignment of the jobs affected by the breakdown.

Time	*Job*	*Order Holon*		*Material Handling Holon*					
				MH-ECT			*MH-OP*		
		OH-ECT	*OH-OP*	*MHH 1*	*MHH 2*	*MHH 3*	*MHH 1*	*MHH 2*	*MHH 3*
20	2	53	21.20	**55**		59	**27.20**		43.20
	3	51	27.20	**51**		55	**27.20**		39.20
	4	53	7.07	58		**57**	26.20		**26.20**
	6	48	15.36	56		**54**	27.36		**41.36**

The MHH_2 breakdown will last until $T = 40$, when the MHH_2 is re-announced as available by the System Monitoring and Database module using the communication mechanisms presented before. The resulting System Level Schedule will change starting with $T = 20$, when the breakdown occurred, and is presented in Figure 13.11. The resulting makespan increased from *66* to *70* due to the greater load of the two remaining MH resources.

The use of the Inter-Material Handling Holon cooperation mechanisms during the operations will reduce the makespan to *68* as presented in Figure 13.12, very close to the optimal one, having a value of *66*, obtained before the breakdown. During the breakdown of MHH_2, the Inter-Material Handling Holon cooperation is used twice, when MHH_3 will execute one of the operations assigned from the start to MHH_1 (a loading for job *1*) and one operation assigned after

the breakdown to MHH_1 (a loading for job *2*).

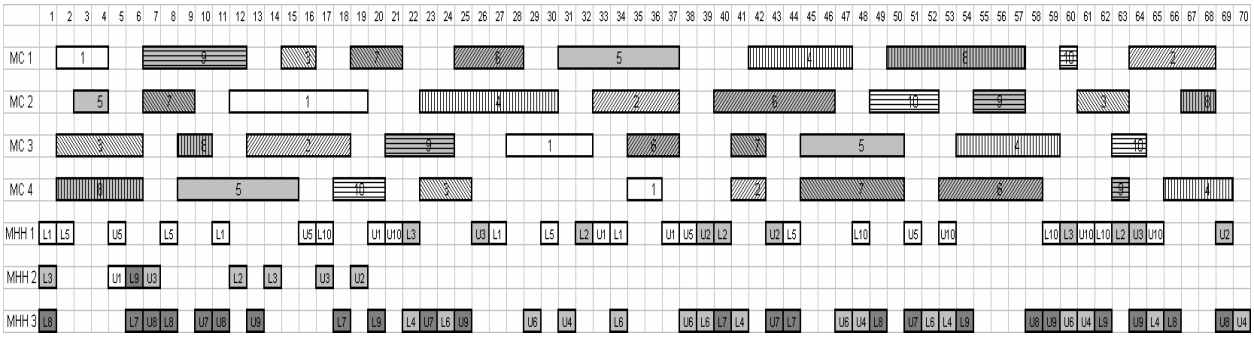

Figure 13.11. Assignment of the jobs affected by the breakdown.

At the time of recovery MHH_2 will be able to receive requests for any new jobs that enter the system, as well as help requests for the other two Material Handling Holons. As there is no new job entering the system, the recovered MHH_2 will respond to four help requests and perform four operations, two for MHH_1 and two for MHH_3.

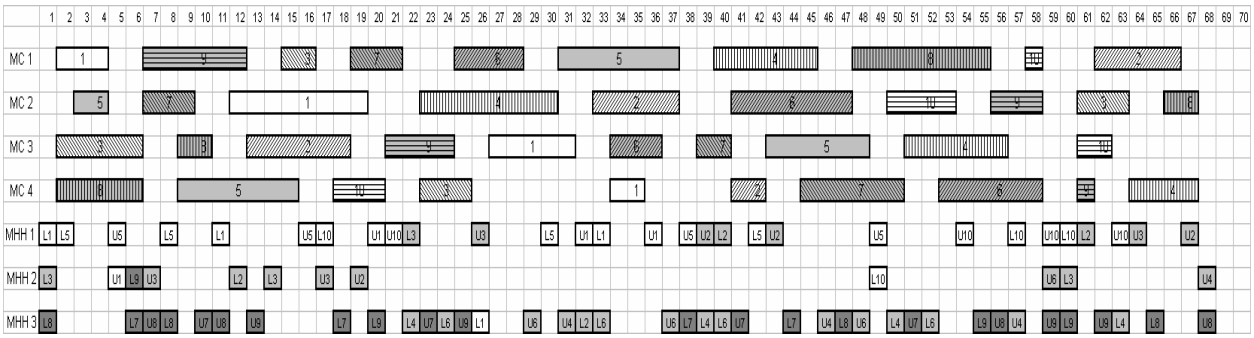

Figure 13.12. Final resulting System Level Schedule after the breakdown and the use of Inter-Material Handling Holon cooperation mechanisms.

13.3.6. Other Results Obtained when Simulating Breakdowns

Another example of a ten-machine job-shop problem ($J10 \mid\mid C_{max}$) with ten jobs and four

MH resources that illustrates the fault-tolerance and hardware reconfigurability capabilities is presented in Appendices D.1 - D.7. The processing jobs give a schedule makespan value of *96*. The theoretical LB_3 value of the schedule makespan that needs seven MH resources is calculated to be equal to *130*. The initial MH operations assignment gives a schedule with a makespan value of *136*. Improvements of this value by using Inter-Material Handling Holon cooperation mechanisms reduce the schedule makespan to *132*, a value very close to the LB_3 value. The simulated breakdown of MHH_4 occurs at time $T = 21$, and the assignment of affected jobs to the remaining MH resources leads to a schedule having a makespan of *138*. After the recovery from the breakdown of MHH_4 at time $T = 51$, which is equivalent with an introduction of a new resource, the use of Inter-Material Handling Holons cooperation mechanisms reduces the final schedule makespan to *133*, very close to the initial solution of *132*, obtained before the breakdown.

13.4. Holonic and Centralized Scheduling for the Ten Job-Shop Problems

To further evaluate the real-time response of the holonic system and the quality of the solution delivered, ten job-shop problems having a larger number of jobs and machines, generated using the framework described in an online database of scheduling problems (Beasley, 2005) are considered as pilot runs. Since the online framework is not considering the MH operations, the data related with the number of MH resources for each problem is considered a random integer in the *[2, 4]* interval. The results obtained by running the holonic and centralized scheduling algorithms, as well as by calculating the values or running the algorithms for the different LB defined in the previous chapters are presented in Tables 13.13 and 13.14 below.

Table 13.13 presents the values of the makespan when MH operations are not considered and the different theoretical LB on the schedule makespan. The results obtained from calculating the LB values show that in all ten job-shop problem cases studied, the algorithmic LB_2 is always greater than the intuitive LB_1 by a value varying between *2.51%* and *20.48%*, with an average of *9.93%*. When considering also the potential CPs existing in the schedule, the improved LB_3 can be slightly greater than the algorithmic LB_2. The difference between LB_3 and LB_2, when the former could be calculated, is very small, in most of the problems studied these two LB being equal. The largest percentage difference found is *0.10%*. Furthermore, if accounting for the

number of available MH resources in the system, the actual LB_4 is likely to increase. All these results are also presented graphically in Figure 13.13. The real-time response of the LB_2 algorithm needed as input for the Global Scheduler's optimal and heuristic algorithms is confirmed by the time to obtain any of the LB_2 values which was in every case less than two seconds.

Table 13.13. Schedule makespan when MH operations are not considered and the calculated LB for the ten job-shop problems considered.

Jobs	*mc*	*MH*	C_{max} *w/o MH*	LB_1	LB_2		LB_3
#	#	#	*Value*	*Value*	*Value*	*CPU Time*	*Value*
10	5	2	686	706	756	0.3	756
10	10	2	742	762	862	0.6	862
10	10	3	1094	1114	1172	0.5	1172
10	10	4	96	116	130	0.7	130
12	10	4	1039	1063	1090	0.7	1090
15	5	2	876	906	953	0.8	954
15	10	3	723	753	895	1.0	895
20	5	2	1235	1275	1307	1.2	1307
20	10	4	717	757	912	1.3	912
30	10	3	635	695	780	1.3	780

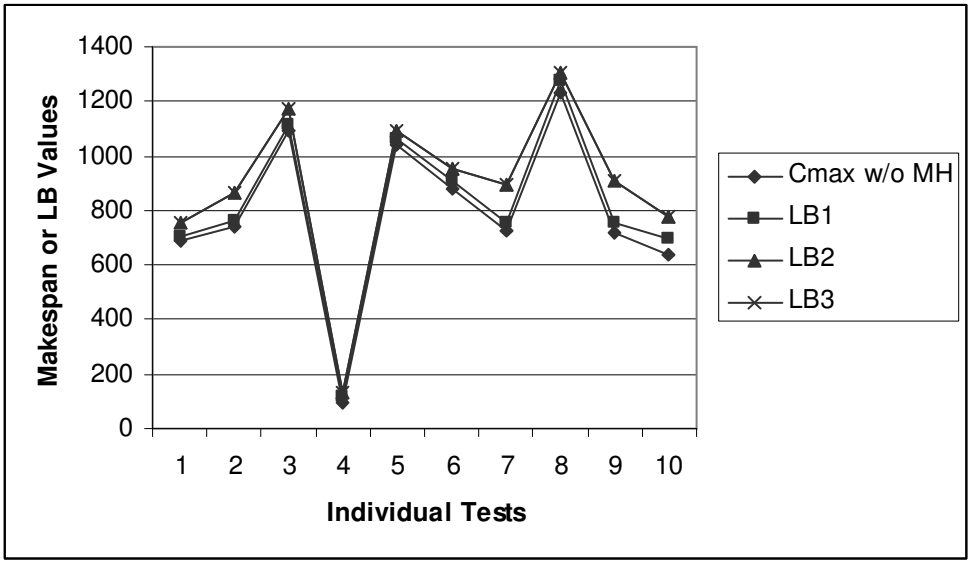

Figure 13.13. Makespan or calculated LB Values when MH operations are not considered for the ten job-shop problems.

Table 13.14 presents the values obtained after running both the algorithms for the holonic scheduling approach and the four Global Scheduler's algorithms in the case of the centralized scheduling approach, as well as the CPU time needed to obtain the results in each case. The same makespan values obtained for both control approaches are presented graphically in Figure 13.14.

Table 13.14. Comparison of the makespan values obtained using both the holonic and centralized scheduling approaches for the ten job-shop problems considered.

jobs	*mc*	*MH*	*Holonic Approach*		*EBFS Optimal Algorithm*		*JCTH Algorithm*		*MCTH Algorithm*		*MH-RCVH Algorithm*	
#	#	#	C_{max}	*CPU Time*	C_{max}	*CPU Time*	C_{max}	*CPU Time*	C_{max}	*CPU Time*	C_{max}	*CPU Time*
10	5	2	766	7.6	766	22.4	787	0.1	792	0.1	795	0.1
10	10	2	894	9.2	880	21.1	898	0.1	906	0.1	912	0.1
10	10	3	1216	10.5	1198	38.6	1232	0.1	1236	0.2	1229	0.2
10	10	4	138	7.4	130	25.9	157	0.2	162	0.1	171	0.2
12	10	4	1114	9.4	1107	34.5	1136	0.3	1143	0.2	1134	0.3
15	5	2	988	8.0	965	35.3	992	0.4	989	0.2	988	0.4
15	10	3	938	9.9	928	39.7	938	0.3	956	0.3	959	0.5
20	5	2	1389	10.3	1360	42.9	1393	0.4	1426	0.4	1432	0.5
20	10	4	971	11.5	945	44.5	968	0.7	982	0.6	972	0.8
30	10	3	878	12.3	852	48.2	886	0.7	898	0.6	893	0.8

The results show that the solutions delivered by the holonic scheduling approach are very close to the optimal results, the percentage difference between the two approaches varying from *0%* (the one case in which the holonic control approach reached the optimal solution) to *6.15%*, with an average of *2.13%.* The EBFS optimal algorithm gives in most of the cases results greater than the theoretical LB_3, varying between *0%* (when the LB_3 value proved to be the optimal solution) and *9.23%*, with an average of *2.89%*. Out of the three heuristics, the best performance is given by the first one, JCTH, with a percentage difference varying between *1.8%* and *20.77%* with an average of *4.37%* compared to the solutions given by the EBFS algorithm. Heuristic 2 has a percentage difference varying between *2.49%* and *24.61%* (average of *5.71%*), and heuristic 3 has a percentage difference varying between *2.38%* and *31.54%* (average of *6.27%*), both in comparison with the optimal solutions.

The real-time response of the heuristic algorithms is confirmed by the time to obtain any of the solutions which was in every case less than one second. The real-time scheduling ability of the holonic approach is monitored by adding the individual CPU times needed to run all the

algorithms that are part of the holonic job evaluation, allocation and execution algorithms. Results show that by adding the CPU times for all the Order and Material Handling Holons algorithms, the solution obtained are delivered in a matter of seconds.

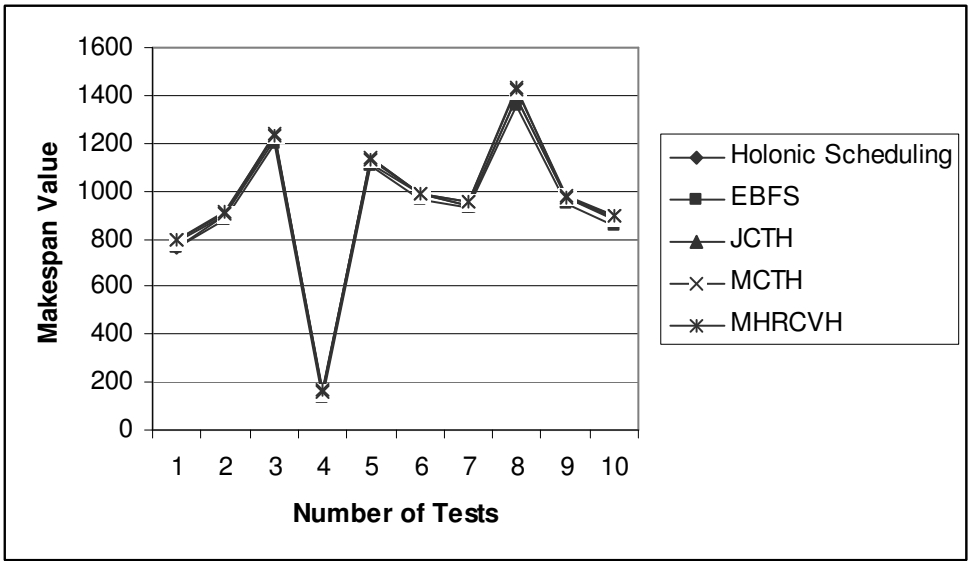

Figure 13.14. Makespan values obtained using both the holonic and centralized scheduling approaches for the ten job-shop problems

13.5. Simulation Study Results

The results obtained from the simulation study are validating the previous presented particular results and give reliable information for the performance of the proposed holonic system and its centralized scheduling extension incorporated in the Global Scheduler's internal architecture. Ten bunches of ten replications each are run for each of the two control approaches. Considering also the three heuristics designed for the Global Scheduler, the LB_2 algorithm, and the evaluation of the MH hardware reconfigurability capabilities, it results that each replication in every bunch is run seven times. The first run is performed to obtain the LB, then, in the next runs, the schedule makespan is calculated using the two control approaches, once for the holonic scheduling and four times for the Global Scheduler's algorithms. In the last run, the holonic scheduling approach is run once again, this time subject to randomly occurring breakdowns.

Each replication consists of 100 jobs having the arrival time, weight, due date, and processing time values as presented before in the design of experiments section.

13.5.1. Lower Bounds on the Schedule Makespan

Tables E.1 - E.10 in Appendix E.1 present the values for the schedule makespan for all replications in the study when MH operations are not considered, the values obtained for both LB_1 and LB_2, the number of time-steps in which the MH resource constraint is violated in the delivered LB_2 schedule, as well as the CPU time necessary to deliver the LB_2 values. The difference between the calculated LB_1 and LB_2 is significant, much larger than in the particular situations considered before, an average for the two LB calculated for all the replications being *1427* and *2166* respectively. This behavior is due to the larger number of operations of every job that is released in the system. Table 13.15 presents the averages of values for the makespan and LB for each bunch of replications for the case in which the MH operations are not considered and for the two types of LBs considered in the simulation study.

Table 13.15. Average makespan when MH operations are not considered and average LB values for each bunch of replications.

Bunches of Replications	*C_{max} w/o MH*	*LB_1*	*LB_2*		
	Value	*Value*	*Value*	*CPU Time*	*Violations*
1	1109.7	1509.7	2139.2	2.39	39.6
2	999.3	1399.3	2230.3	2.44	29.8
3	1008.3	1408.3	2163.2	2.38	35.8
4	984.5	1384.5	2177.2	2.22	35.5
5	1032.9	1432.9	2134.8	2.06	37.4
6	1041.2	1441.2	2228.0	1.98	33.3
7	948.2	1348.2	2137.8	1.97	34.7
8	1040.3	1440.3	2035.5	2.12	36.6
9	1012.0	1412.0	2205.0	1.95	29.5
10	1097.3	1497.3	2206.9	2.00	29.7
Average	1027.37	1427.37	2165.79	2.15	34.19

The same information is presented graphically in Figure 13.15. The average for the number of time-steps in which the MH resource constraint is violated over all the replications is found to be approximately equal to *34*. The CPU time to deliver the LB_2 schedule is in average

2.15 seconds, and in all replications is less than *5* seconds.

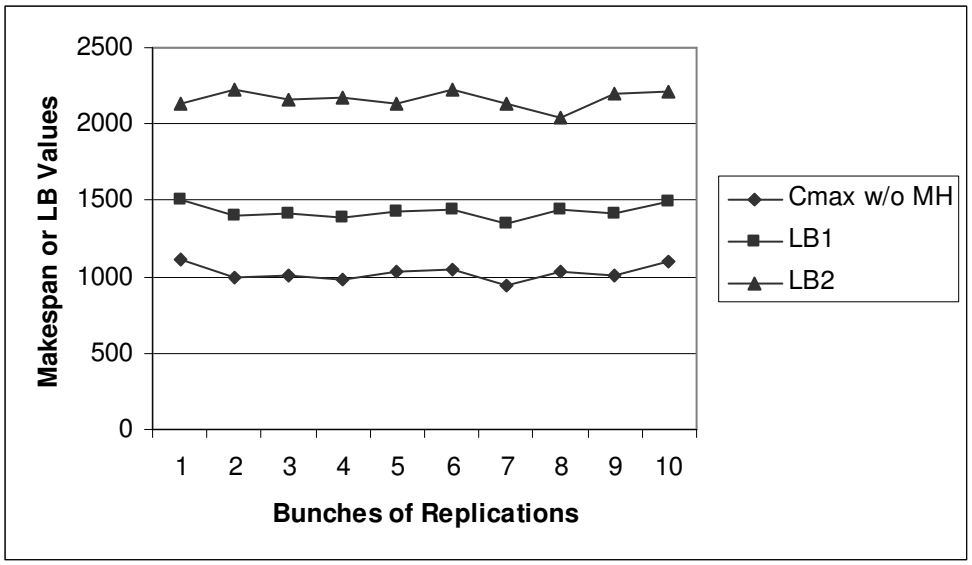

Figure 13.15. Average makespan when MH operations are not considered and LB values obtained in each bunch of replications.

13.5.2. Holonic Scheduling Control Approach Evaluation

The ten bunches of ten replications each performed for the two different control approaches gave the results presented in Tables E.11 - E.20 in Appendix E.2. At the end of each replication the performance measures monitored are the completion time of the last unloading job, along with the time needed to obtain it. The paired-t approach, presented before in the simulation scenarios section, is used to assess the performance of the holonic decentralized scheduling approach in comparison with the optimal solutions obtained by running the Global Scheduler's EBFS algorithm. Table 13.16 presents the averages of the makespan values for all ten bunches of replications in the case of holonic scheduling and the EBFS algorithm. Figure 13.16 presents the average makespan values for the two approaches considered in the case of each bunch of replications.

Using the paired-t approach, the sample mean and sample standard deviation for the

differences between the two approaches are calculated using equations (12.49) and (12.50) and have the values:

$\overline{X}_{(1-2)} = 20.04$ and $S_{(1-2)} = 5.01$

Table 13.16. Average makespan values and CPU times for the holonic scheduling and EBFS algorithm for each bunch of replications.

Bunches of Replications	*Holonic Approach*		*EBFS Optimal Algorithm*	
	C_{max}	*CPU Time*	C_{max}	*CPU Time*
1	2261.4	31.88	2238.8	128.36
2	2352.7	32.73	2324.7	130.14
3	2281.1	31.84	2259.1	123.95
4	2287.9	31.83	2269.3	111.87
5	2235.4	32.01	2224.6	138.47
6	2329.6	32.16	2313.3	132.29
7	2228.1	31.82	2208.3	126.08
8	2163.5	31.97	2146.3	132.19
9	2308.5	32.21	2282.1	130.53
10	2313.5	32.43	2294.8	132.27
Average	2276.17	32.09	2256.13	128.62

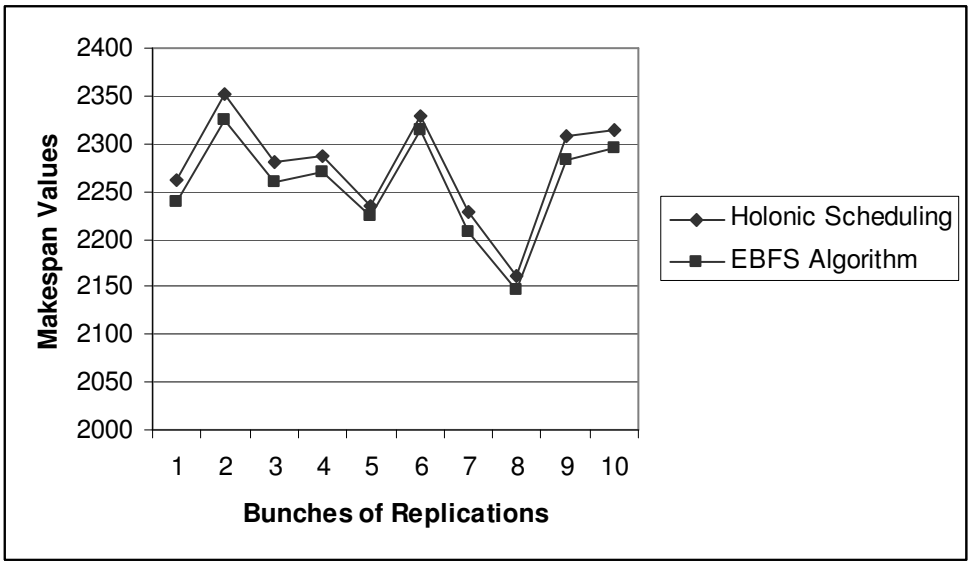

Figure 13.16. Average makespan values for the two control approaches obtained in each bunch of replications.

The averages values for both systems, as well as the differences in their averages

necessary to perform the paired-t approach are presented in Table 13.17.

Table 13.17. Average makespan values for the holonic scheduling and EBFS algorithm for each bunch of replications and their differences.

Bunches of Replications j	X_1	X_2	$X_{(1-2)}$
1	2261.4	2238.8	22.6
2	2352.7	2324.7	28.0
3	2281.1	2259.1	22.0
4	2287.9	2269.3	18.6
5	2235.4	2224.6	10.8
6	2329.6	2313.3	16.3
7	2228.1	2208.3	19.8
8	2163.5	2146.3	17.2
9	2308.5	2282.1	26.4
10	2313.5	2294.8	18.7
Mean $\overline{X}_{(1-2)}$			20.04
Standard Deviation $S_{(1-2)}$			5.01

The variance needed to form a $100(1 - \alpha)$ CI is calculated using equation (12.51) and is equal to:

$$Var\left[\overline{X}_{(1-2)}\right] = 2.51$$

By using expression (12.52) an approximate *90* percent confidence interval for the expected difference in the responses of the two systems, $E\left(X_{(1-2)_j}\right)$, is found to be:

$$(100 - \alpha)\%\ CI(\mu_{(1-2)}) = [17.14, 22.94]$$

Thus, with approximately *90* percent confidence, it can be stated that μ_1 and μ_2, the expected means of the responses of the two systems, are different. Since the constructed CI could not contain the zero value because there is no approach that can give better results than system S_2 represented by the EBFS optimal algorithm, this outcome is as expected. However, the holonic scheduling approach has the advantage of the real-time response obtained. Even the EBFS algorithm is better than the holonic approach in a statistical sense, the difference between them in a practical sense is less than *2%* for each bunch of replications, with an average of *0.89%*. By cumulating all the individual times for the holonic evaluation, allocation and execution algorithms, the average CPU time over all *100* replications is *32.09* seconds, whereas the same

average for the EBFS algorithm has a value a greater than *2* minutes per replication (*128.62* seconds). The maximum CPU time monitored for the EBFS algorithm was *276.6* seconds. With an increase in the number of operations this value is likely to further increase, much more than the increase in the CPU time for the holonic scheduling, due to the reasons related with the large number of possible combinations that could appear in the case of the EBFS algorithm.

To compare the performance of the holonic scheduling approach in respect to the three heuristics algorithms the same paired-t approach is used three times, and each time the holonic scheduling results are contrasted with the results given by each of the three heuristics. The ten bunches of ten replications performed for each of the three heuristics algorithms delivered the results presented in Tables E.21 - E.30 in Appendix E.3. At the end of each of the replication the performance measure monitored is the completion time of the last unloading job, along with the time needed to obtain it.

Table 13.18 presents the averages of the makespan values for all ten bunches of replications and for all the three heuristics studied, and the average CPU times needed to obtain them. A graphical comparison of these makespan values for the ten bunches of replications is presented in Figure 13.17.

Table 13.18. Average makespan values and CPU times for the three heuristics for each bunch of replications.

Bunches of Replications	*JCTH Algorithm*		*MCTH Algorithm*		*MH-RCVH Algorithm*	
	C_{max}	*CPU Time*	C_{max}	*CPU Time*	C_{max}	*CPU Time*
1	2271.7	2.17	2292.8	2.50	2311.1	2.31
2	2364.0	1.86	2392.6	2.70	2384.5	2.41
3	2292.4	1.70	2327.3	2.36	2303.5	1.97
4	2291.4	1.90	2325.3	2.51	2333.3	1.82
5	2261.0	1.89	2276.0	2.37	2275.0	1.95
6	2349.5	1.90	2360.2	1.88	2361.2	2.18
7	2235.8	2.05	2265.9	2.09	2262.9	2.01
8	2180.6	2.07	2215.6	2.04	2215.5	1.82
9	2317.3	1.71	2329.8	2.03	2338.4	1.82
10	2271.7	1.85	2292.8	2.05	2311.1	1.96
Average	2289.27	1.91	2311.56	2.25	2311.44	2.03

Table 13.19 gives the average responses obtained for the three systems, S_3, S_4, and S_5, and the differences between these responses and the response of the holonic scheduling algorithms

when running all the replications in the simulation study.

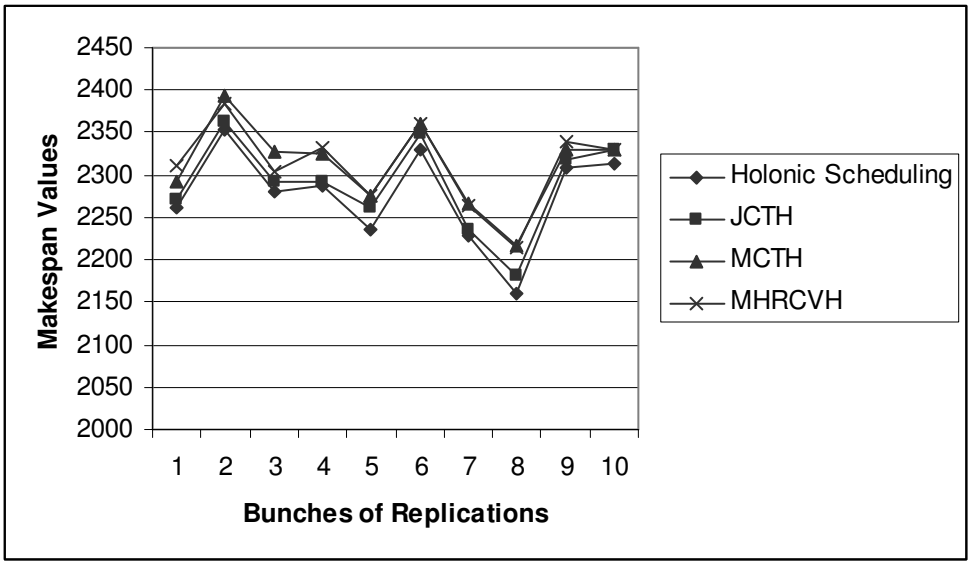

Figure 13.17. Average makespan values for the three heuristics compared with the holonic scheduling solution obtained in each bunch of replications.

Table 13.19. Average makespan values for the three heuristic algorithms and the differences in comparison to the holonic scheduling approach.

Bunches of Replications j	X_1	X_3	X_4	X_5	$X_{(1-3)}$	$X_{(1-4)}$	$X_{(1-5)}$
1	2261.4	2271.7	2292.8	2311.1	10.3	31.4	49.7
2	2352.7	2364.0	2392.6	2384.5	11.3	39.9	31.8
3	2281.1	2292.4	2327.3	2303.5	11.3	46.2	22.4
4	2287.9	2291.4	2325.3	2333.3	3.5	37.4	45.4
5	2235.4	2261.0	2276.0	2275.0	25.6	40.6	39.6
6	2329.6	2349.5	2360.2	2361.2	19.9	30.6	31.6
7	2228.1	2235.8	2265.9	2262.9	7.7	37.8	34.8
8	2163.5	2180.6	2215.6	2215.5	17.1	52.1	52
9	2308.5	2317.3	2329.8	2338.4	8.8	21.3	29.9
10	2313.5	2329.0	2330.1	2329.0	15.5	16.6	15.5
Mean $\overline{X}_{(i-k)}$					13.10	35.39	35.27
Standard Deviation $S_{(i-k)}$					6.48	10.80	11.64

Using the individual paired-t approach three times for comparing the three heuristics with

the holonic scheduling approach, the sample means and sample standard deviations for each of the comparisons, calculated using equations (12.49) and (12.50) are:

$\overline{X}_{(1-3)} = 13.10$ and $S_{(1-3)} = 6.48$

$\overline{X}_{(1-4)} = 35.39$ and $S_{(1-4)} = 10.80$

$\overline{X}_{(1-5)} = 35.27$ and $S_{(1-5)} = 11.64$

The variances needed to form $100(1 - \alpha)$ CIs are calculated using equation (12.51) and are equal to:

$Var\left[\overline{X}_{(1-3)}\right] = 4.20$

$Var\left[\overline{X}_{(1-4)}\right] = 11.60$

$Var\left[\overline{X}_{(1-5)}\right] = 13.56$

By using expression (12.52) the approximate *90%* CIs for the expected differences in the responses of the three heuristic algorithms, $E\left(X_{(1-3)_j}\right)$, $E\left(X_{(1-4)_j}\right)$, and $E\left(X_{(1-5)_j}\right)$ are found to be:

JCTH: $(100 - \alpha)\%\ CI(\mu_{(1-3)}) = [9.34, 16.86]$

MCTH: $(100 - \alpha)\%\ CI(\mu_{(1-4)}) = [29.15, 41.63]$

MHRCVH: $(100 - \alpha)\%\ CI(\mu_{(1-5)}) = [28.52, 48.02]$

From these CIs, it can be stated that the results given by each of the three heuristics are statistically different than the results deliverd by the holonic scheduling approach. Since all the CIs obtaines are positive the results given by the holonic scheduling are better in terms of the quality of the solution than those delivered by each of the three heuristics. However, the three heuristics performed better in terms of the time to deliver the solutions, so they can be used when for obtaining very fast feasible solutions for the assignment of MH operations.

13.5.3. Global Scheduler's Heuristics Evaluation

Using the results given by running the three heuristics presented above, pairwise comparisons using the paired-t approach are performed to contrast the three algorithms designed specifically to give real-time solutions. A graphical comparison of these makespan values for the ten bunches of replications presented in contrast to the EBFS algorithm's solution is presented in Figure 13.18. Table 13.20 gives the average responses obtained for all three systems, S_3, S_4, and

S_5, and the differences between these responses and the response of the EBFS algorithm when running all the replications in the simulation study.

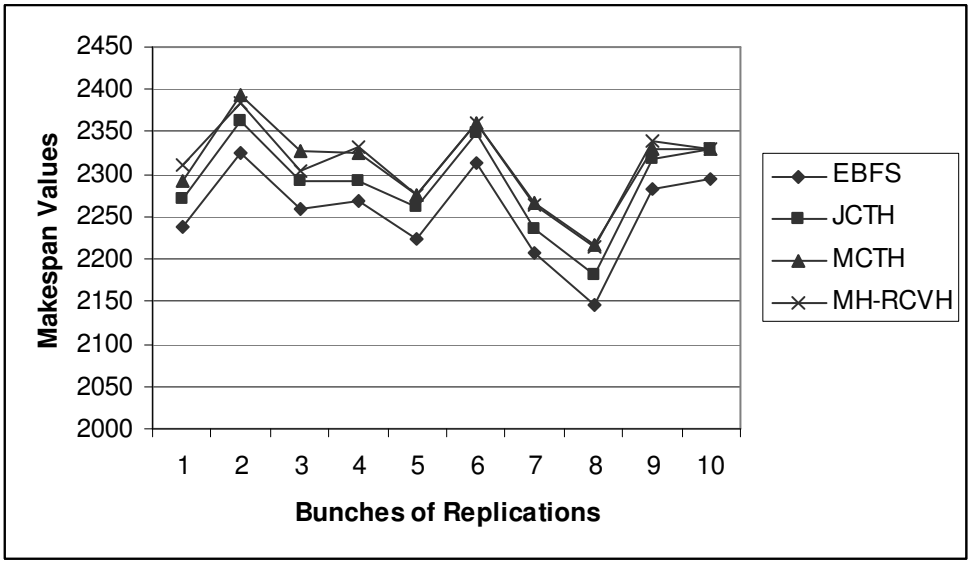

Figure 13.18. Average makespan values for the three heuristics compared with the optimal solution obtained in each bunch of replications.

Using the individual paired-t approach three times for comparing the three heuristics with the EBFS algorithm, the sample means and sample standard deviations for each of the comparisons, calculated using equations (12.49) and (12.50) are:

$\overline{X}_{(2-3)} = 33.14$ and $S_{(2-3)} = 4.93$

$\overline{X}_{(2-4)} = 55.43$ and $S_{(2-4)} = 10.91$

$\overline{X}_{(2-5)} = 55.31$ and $S_{(2-5)} = 11.65$

The variances needed to form $100(1 - \alpha)$ CIs are calculated using equation (12.51) and are equal to:

$Var\left[\overline{X}_{(2-3)}\right] = 2.43$

$Var\left[\overline{X}_{(2-4)}\right] = 11.91$

$Var\left[\overline{X}_{(2-5)}\right] = 13.58$

Table 13.20. Average makespan values for the three heuristic algorithms and the differences in comparison to EBFS algorithm.

Bunches of Replications j	X_2	X_3	X_4	X_5	$X_{(2-3)}$	$X_{(2-4)}$	$X_{(2-5)}$
1	2238.8	2271.7	2292.8	2311.1	32.9	54	72.3
2	2324.7	2364.0	2392.6	2384.5	39.3	67.9	59.8
3	2259.1	2292.4	2327.3	2303.5	33.3	68.2	44.4
4	2269.3	2291.4	2325.3	2333.3	22.1	56.0	64.0
5	2224.6	2261.0	2276.0	2275.0	36.4	51.4	50.4
6	2313.3	2349.5	2360.2	2361.2	36.2	46.9	47.9
7	2208.3	2235.8	2265.9	2262.9	27.5	57.6	54.6
8	2146.3	2180.6	2215.6	2215.5	34.3	69.3	69.2
9	2282.1	2317.3	2329.8	2338.4	35.2	47.7	56.3
10	2294.8	2329.0	2330.1	2329.0	34.2	35.3	34.2
Mean $\overline{X}_{(i-k)}$					33.14	55.43	55.31
Standard Deviation $S_{(i-k)}$					4.93	10.91	11.65

By using expression (12.52) the approximate *90%* CIs for the expected differences in the responses of the three heuristic algorithms, $E(X_{(2-3)_j})$, $E(X_{(2-4)_j})$, and $E(X_{(2-5)_j})$ are found to be:

JCTH: $(100 - \alpha)\%\ CI(\mu_{(2-3)}) = [31.28, 40.00]$

MCTH: $(100 - \alpha)\%\ CI(\mu_{(2-4)}) = [49.10, 61.76]$

MHRCVH: $(100 - \alpha)\%\ CI(\mu_{(2-5)}) = [48.55, 62.07]$

From the first three CIs, it can be stated that the best of the three heuristics is the first one, the JCTH, which is designed to choose, whenever there is a MH resource constraint violation, the job that has the largest completion time of all the jobs that need L/U operations at that particular time.

Since there is no much information about the performance of the second and third heuristic, a pairwise comparison approach that contrasts the performance of the three heuristics with every other one is run such that the other two heuristics can be ranked. Table 13.21 gives the average responses obtained for all three systems, S_3, S_4, and S_5, and the differences between these responses when running all the replications in the simulation study.

Using three times the paired-t approach, the sample means and sample standard deviations for the pairwise differences between the three heuristics, calculated using equations

(12.49) and (12.50) are:

$\overline{X}_{(3-4)} = -22.29$ and $S_{(3-4)} = 11.98$

$\overline{X}_{(3-5)} = -22.17$ and $S_{(3-5)} = 13.61$

$\overline{X}_{(4-5)} = 0.12$ and $S_{(4-5)} = 3.79$

The variances needed to form $100(1 - \alpha)$ CIs are calculated using equation (12.51) and are equal to:

$Var\left[\overline{X}_{(3-4)}\right] = 14.33$

$Var\left[\overline{X}_{(3-5)}\right] = 18.52$

$Var\left[\overline{X}_{(4-5)}\right] = 1.43$

Table 13.21. Average makespan values for the three heuristic algorithms and their pairwise differences.

Bunches of Replications j	X_3	X_4	X_5	$X_{(3-4)}$	$X_{(3-5)}$	$X_{(4-5)}$
1	2271.7	2292.8	2311.1	-21.1	-39.4	-18.3
2	2364.0	2392.6	2384.5	-28.6	-20.5	8.1
3	2292.4	2327.3	2303.5	-34.9	-11.1	23.8
4	2291.4	2325.3	2333.3	-33.9	-41.9	-8.0
5	2261.0	2276.0	2275.0	-15.0	-14.0	1.0
6	2349.5	2360.2	2361.2	-10.7	-11.7	-1.0
7	2235.8	2265.9	2262.9	-30.1	-27.1	3.0
8	2180.6	2215.6	2215.5	-35.0	-34.9	0.1
9	2317.3	2329.8	2338.4	-12.5	-21.1	-8.6
10	2329.0	2330.1	2329.0	-1.1	0.0	1.1
Mean $\overline{X}_{(i-k)}$				-22.29	-22.17	0.12
Standard Deviation $S_{(i-k)}$				11.98	13.61	3.79

By using expression (12.52) the approximate *90%* CIs for the expected differences in the responses of the three heuristic algorithms, $E\left(X_{(3-4)j}\right)$, $E\left(X_{(3-5)j}\right)$, and $E\left(X_{(4-5)j}\right)$ are found to be:

$(100 - \alpha)\%\ CI(\mu_{(3-4)}) = [-29.23, -15.35]$

$(100 - \alpha)\%\ CI(\mu_{(3-5)}) = [-30.06, -14.28]$

$(100 - \alpha)\%\ CI(\mu_{(4-5)}) = [-2.07, 2.31]$

The CI constructed for the difference in the expected response of the second and third heuristics contains the zero value, so these two heuristics are similar from statistical perspective, and when run they should give similar results. The average CPU times for all the three heuristics are in the range of a few seconds, the best time being recorded for the first algorithm with an overall average over all replications of *1.91* seconds. Running the other two heuristics gave an overall average of *2.25* and *2.03* seconds, respectively.

13.5.4. Fault-Tolerance and MH Hardware Reconfigurability Capabilities Evaluation

The same ten bunches of ten replications were run to evaluate the capability of the holonic system to respond in real-time to changes in the number of available MH resources. As stated before, resource breakdowns were considered during the simulation with the MTBF and MTTR times coming from the distributions defined before in the design of experiments section. The results obtained are presented in Appendix E.4 in Tables E.31 - E.40. The performance measures monitored were the completion time of the last unloading job, and the time needed to obtain it. Table 13.22 presents the averages of the makespan values after running all ten bunches of replications for the holonic scheduling approach when resource breakdowns are considered. Figure 13.19 presents the average makespan values after running the holonic scheduling approach in both situations, with or without changing the number of available MH resources.

The results obtained when considering the potential MH resource breakdowns in the system show an increase in the total completion time of all jobs in the system, as expected since the same number of L/U operations needs to be executed with less MH resources when breakdowns occur. However, the difference is not very large, the largest percentage difference in the averages of the ten bunches of replications is *2.91%,* with an overall average of *2.35%.* The CPU times needed to run all the holonic scheduling job evaluation, allocation, and execution algorithms is increasing in the case when MH resource breakdowns are considered, as expected because of the need to run the algorithms several more times, but overall the results are delivered in less than *50* seconds, so the real-time response is still obtained. There are individual replications in which the percentage difference in the makespan values is as high as *11.76%*, when more than one MH resource is down in the same time, but overall the holonic system is

capable of responding in real-time and can deliver good feasible solutions when the number of available MH resources is changing during operations.

Table 13.22. Average makespan values and CPU times for the holonic scheduling approach with and without considering MH resource breakdowns.

Bunches of Replications	*Holonic Scheduling w/o Resource Breakdowns*		*Holonic Scheduling w/ Resource Breakdowns*	
	C_{max}	*CPU Time*	C_{max}	*CPU Time*
1	2261.4	31.88	2316.7	44.16
2	2352.7	32.73	2405.7	44.55
3	2281.1	31.84	2347.4	43.80
4	2287.9	31.83	2349.6	43.80
5	2235.4	32.01	2287.0	44.14
6	2329.6	32.16	2376.8	44.17
7	2228.1	31.82	2282.6	43.67
8	2163.5	31.97	2223.1	43.98
9	2308.5	32.21	2350.7	44.43
10	2313.5	32.43	2356.4	44.60
Average	2276.17	32.09	2329.60	44.13

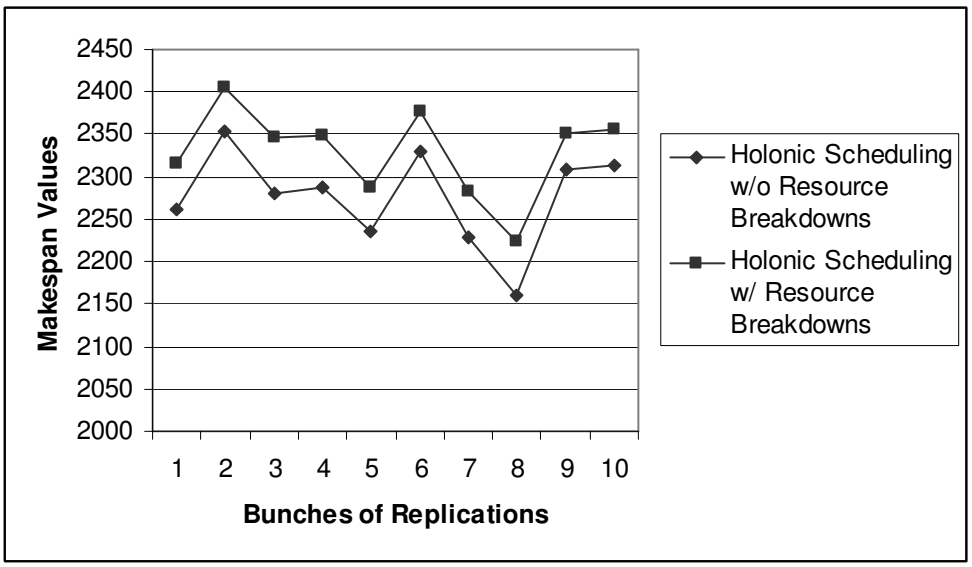

Figure 13.19. Comparison of the average makespan values for the holonic scheduling approach with and without considering resource breakdowns.

14. Conclusions

The fundamental difference between holonic control and conventional control comes from the distributed decision-making (multiple small decision spaces) existing in holonic systems as opposed to the centralized or hierarchical decision-making (single large decision space) in the case of conventional scheduling systems. The distributed decision-making for the job assignment problem translates in simple algorithms having reduced computational complexity that can be solved practically in real-time for any potential real-world situation.

14.1. Overview of the Results

As illustrated by the tests performed in the experimental part of this dissertation, MH does add additional time to the processing of jobs in manufacturing cell environments. When accounting for the MH operation times by merely adding them to the total completion time of the processing jobs, the results obtained are not realistic. The algorithmic LB_2 and calculated LB_3 give a much better value for estimating the schedule makespan when MH operations are considered. Evaluating the results obtained by calculating LB and taking in consideration the potential delays that may be introduced in the schedule by not selecting the best order of performing the MH operations can lead to improved solutions and a better utilization of the available resources.

The results obtained by running the algorithms designed in this research show that the proposed holonic system is capable of accommodating new arriving jobs and delivers good solutions in real-time. As presented in detail in the particular examples considered before the simulation study, the holonic scheduling approach chooses one of the three best assignments in terms of schedule makespan, and with the aid of the Inter-Material Handling Holon allocation procedure further improves this assignment to obtain an even better solution.

The simulation study results confirm the real-time capabilities and the good feasible solutions delivered by the proposed holonic system. There is a very small difference in the results obtained by running the holonic algorithms compared to the optimal solution. Moreover, considering the real-time response, confirmed by the time to obtain a feasible solution which is always less than one minute, the results given by the holonic approach show a significant

potential. Modifying the number of available MH resources, which includes fault-tolerance and scalability, is also supported in real-time by the holonic scheduling system. The LB_2 algorithm, and its LB_3 extension, can be successfully used to obtain real-time information about the theoretically possible LB values for the schedule makespan and for finding quick feasible solutions for job-shop problems when using one of the three Global Scheduler's heuristics.

Using simple algorithms that never lead to combinatorial explosion, the performance of the holonic approach will always have the strength of a real-time response. Even the optimality of the solution cannot be guaranteed, using the procedures described in this research, the solution can be compared with the LB_2 values, so its difference from the theoretical best possible solution can be calculated. Considering the large number of real-world scheduling problems for which searching for an optimal solution leads to combinatorial explosion, the lack of optimality of the holonic control approach is not an important issue overall.

14.2. Future Research Directions

Holonic control, as stated by Bongaerts *et al.* (2000) "tries to merge the best properties of both hierarchical and heterarchical systems, namely, a high and predictable performance with a high robustness against disturbances and unforeseen changes," and therefore it is appropriate for next generation manufacturing systems. As holonic concept is implemented using MAS, advances in MAS theory can offer valuable tools for future development and enhancement of holonic systems. Agent technology is a vast area of research and parts of it are especially promising for future application in the holonic systems theory. Different types of learning mechanisms, self-organization of societies of agents, or different types of emergent behavior are just a few promising research areas within DAI having potential applications to holonic systems. As an example, Cardon *et al.* (2000) suggested that incorporating genetic algorithms as the driver for the agents in MAS and considering the AI behavior motivation for agents, newly formed agents become a completely autonomous genetic entity that "can lead to a drastically new kind of emergence phenomenon in self-organizing multi-agent systems."

While the holonic system theory evolves and new systems are being developed and implemented in research centers, a needed step forward is to refine and prepare the new control systems for the big real-world challenge in which holonic systems must demonstrate an

improved performance beyond that of traditional control systems. In this regard there are some open issues in both holonic control system design and implementation that need to be addressed to obtain a larger industrial acceptance. For the holonic control system design, McFarlane and Bussmann (2003) pointed out the need for proven design methodologies that can provide consistency and reliability in holonic systems, clearly evidence of improvements delivered by the holonic control mechanisms and migration to full holonic control algorithms. On the other hand, for the holonic control system implementation, McFarlane and Bussmann (2003) indicated the need for adapting the holonic control systems to be used with the available computing systems, and the need for standardization of data exchange, internal algorithms and architectures of the holonic control systems.

Presently, the industrial acceptance of holonic systems is relatively low due to reasons inherent to the holonic system theory and as a consequence of a company's business strategy, as presented below:

- Agent-based technology on which holonic systems are built, and especially its manufacturing applications, is not completely mature.
- Difficulties in adapting the existing equipment to work using the holonic approach, or, the large investments needed to build a holonic production system.
- Resistance to change existing in any organization, as always.

However, some of these adverse issues can be overcome as agent technology and its applications will achieve maturity and the computational power cost decreases every year. Resistance to change factor is not discussed here, as it virtually affects all aspects of social and economic life. In theory, the holonic control approach is considered a viable solution for the next generation manufacturing systems. By addressing the issues above, the holonic control approach could make the important step from theory to accepted practice.

References

1. Agility Forum (1997) *Next Generation Manufacturing Project: A Framework for Action*, Agility Forum, Leaders for Manufacturing, Technologies Enabling Agile Manufacturing.
2. Alonso, E., d'Inverno, M., Kudenko, D., Luck, M. and Noble, J. (2001) Learning in multi-agent systems. *The Knowledge Engineering Review*, Vol. 16:3, 277-284.
3. Arai, T., Aiyama, Y., Sugi, M. and Ota, J. (2001) Holonic assembly system with Plug and Produce. *Computers in Industry,* 46, 289-299.
4. Aydin, M. E. and Oztemel, E. (2000) Dynamic job-shop scheduling using reinforcement learning agents. *Robotics and Autonomous Systems* 33, pp. 169-178.
5. Balasubramanian, S., Zhang, X. and Norrie, D. H. (2000) Intelligent control for holonic manufacturing systems, in *Proceedings of the Institute of Mechanical Engineers,* Vol. 214, Part B, 953-961.
6. Balasubramanian, S., Brennan, R. W. and Norrie, D. H. (2001) An architecture for metamorphic control of holonic manufacturing systems. *Computers in Industry,* 46, 13-31.
7. Barto A. G. and Dietterich T. G. (2004) Reinforcement learning and its relationship to supervised learning, in *Si, J., Barto, A. G., Powell W. B., Wunsch II. D. (ed.), Handbook of learning and approximate dynamic programming,* Institute of Electrical and Electronics Engineers, Inc.
8. Bauer, A., Bowden, R., Browne, J., Duggan, J. and Lyons, G. (1994) *Shop floor control systems - from design to implementation,* Chapman & Hall, London.
9. Beasley, J. E. (2005) OR - Library, Available online at http://people.brunel.ac.uk/~mastjjb/jeb/info.html (Accessed: June 4, 2005).
10. Berlinski, D. (2000) *The advent of the algorithm, the idea that rules the world*, Harcourt, Inc., New York.
11. Bongaerts, L., Jordan, P., Timmermans, P., Valckenaers, P. and Wyns, J. (1997) Evolutionary development in shop floor control. *Computers in Industry,* 33, 295-304.
12. Bongaerts, L., Monostori, L., McFarlane, D. and Kadar, B. (2000) Hierarchy in distributed shop floor control. *Computers in Industry,* 43, 123-137.
13. Brennan, R. W. and O, W. (2000) A simulation test-bed to evaluate multi-agent control of manufacturing systems, in *Proceedings of the 2000 Winter Simulation Conference,* 1747-1756.
14. Brennan, R.W., Norrie, D. H. (2001) Evaluating the performance of reactive control architectures for manufacturing production control. *Computers in Industry,* 46, 235-245.
15. Brennan, R. W., Fletcher, M. and Norrie, D. H. (2001) Reconfiguring real-time holonic manufacturing systems, in *Proceedings of the 12th International Workshop on Database and Expert Systems Application*s, 611-615.
16. Bruckner, S., Wyns, J., Peeters, P. and Kollingbaum, M. (1998) Designing agents for the manufacturing control, in *Proceedings of Artificial Intelligence and Manufacturing Research Planning Workshop - State of the Art & State of the Practice,* 40-46.
17. Bussmann, S. and Sieverding, J. (2001) Holonic control of an engine assembly plant - an industrial

evaluation, in *Proceedings of the 2001 IEEE Systems, Man, and Cybernetics Conference,* Tucson, AZ.

18. Bussmann, S., Jennings, N. R. and Wooldridge, M. (2004) *Multi-agent Systems for Manufacturing Control: A Design Methodology,* Springer-Verlag, Berlin.
19. Buttazzo, G. C. (2002) *Hard real-time computing systems; predictable scheduling algorithms and applications*, Kluwer Academic Publishers, Norwell, Massachusetts.
20. Cantamessa, M. (1997) Hierarchical and heterarchical behavior in agent-based manufacturing systems. *Computers in Industry*, 33, 305-316.
21. Cardon, A., Galinho, T. and Vacher, J.-P. (2000) Genetic algorithms using multi-objectives in a multi-agent system. *Robotics and Autonomous Systems,* 33, 179-190.
22. Cheng, F.-T., Wu, S.-L. and Chang, C.-F. (2001) Systematic approach for developing holonic manufacturing execution systems, in *Proceedings of the 27th Annual Conference of the IEEE Industrial Electronics Society,* 261-266.
23. Cheung, H. M. E., Yeung, W. H. R., Ng, H. C. A. and Fung, S. T. R. (2000) HSCF: a holonic shop floor control framework for flexible manufacturing systems. *International Journal of Computer Integrated Manufacturing,* Vol. 13, No. 2, 121-138.
24. Chirn, J.-L. and McFarlane, D. C. (2000a) A holonic component-based approach to reconfigurable manufacturing control architecture, in *Proceedings of the 11th International Workshop on Database and Expert Systems Applications*, 219-223.
25. Chirn, J.-L. and McFarlane, D. C. (2000b) A component-based approach to the holonic control of a robot assembly cell, in *Proceedings of 2000 IEEE International Conference on Robotics and Automation*, Vol. 4, 3701-3706.
26. Chokshi, N. N. and McFarlane, D. C. (2001) Rationales for holonic manufacturing systems in chemical process industries, in *Proceedings of the 12th International Workshop on Database and Expert Systems Applications*, 616-620.
27. Christensen, J. H. (1994) Holonic manufacturing systems: initial architecture and standards directions, in *Proceedings of First Conference on Holonic Manufacturing Systems,* Hanover, Germany.
28. Christensen, J. H. (2003) HMS/FB Architecture and its implementation, in *Deen, S. M. (ed.), Agent-Based Manufacturing: Advances in the Holonic Approach*, Springer-Verlag, Berlin, 53-87.
29. Claus, C. and Boutilier, C. (1998) The dynamics of reinforcement learning in cooperative multi-agent systems, in *Proceedings of the Fifteenth National Conference on Artificial Intelligence,* AAAI-98.
30. Cowling, P. and Johansson, M. (2002) Using real time information for effective dynamic scheduling. *European Journal of Operational Research,* 139, 230-244.
31. Cselenyi, J. and Toth, T. (1998) Some questions of logistics in the case of holonic systems. *Journal of Intelligent Manufacturing,* 9, 113-118.
32. Deen, S. M. (2003) An investigation into a computational model for HMS, in *Deen, S. M. (ed.), Agent-Based Manufacturing: Advances in the Holonic Approach*, Springer-Verlag, Berlin, 147-161.
33. Depke, R., Heckel, R. and Kuster, J. M. (2002) Formal agent-oriented modeling with UML and graph

transformation. *Science of Computer Programming,* 44, 229-252.

34. Dilts, D. M., Boyd, N. P. and Whorms, H. H. (1991) The evolution of control architectures for automated manufacturing systems. *Journal of Manufacturing Systems,* Vol. 10, No. 1, 79-93.
35. Doran, J. E., Franklin, S., Jennings, N. R. and Norman, T. J. (1997) On cooperation in multi-agent systems. *Knowledge Engineering Review,* 12(3), 309-314.
36. Duffie, N. A. (1990) Synthesis of heterarchical manufacturing systems. *Computers in Industry* 14, 167-174.
37. Fletcher, M., Garcia-Herreros, E., Christensen, J. H., Deen, S. M. and Mittmann, R. (2000) An open architecture for holonic cooperation and autonomy, in *Proceedings of the 11th International Workshop on Database and Expert Systems Applications,* 224-230.
38. Fletcher, M., Brennan, R. W. and Norrie, D. H. (2001) Deadline control in holonic manufacturing using mobile agents, in *Proceedings of the 12th International Workshop on Database and Expert Systems Applications*, 623-627.
39. Fletcher, M. and Deen, S. M. (2001) Fault-tolerant holonic manufacturing systems. *Concurrency and Computation: Practice and Experience,* 13, 43-70.
40. Gayed, N., Jarvis, D. H. and Jarvis, J. H. (1998) A strategy for the migration of existing manufacturing systems to holonic systems, in *Proceedings of the 1998 IEEE International Conference on Systems, Man, and Cybernetics*, Vol. 1, 319-324.
41. Giebels, M. M. T., Kals, H. J. J. and Zjim, W. H. M. (2001) Building holarchies for concurrent manufacturing planning control in EtoPlan. *Computers in Industry* 46, 301-314.
42. Glanzer, K., Hammerle, A. and Geurts, R. (2001) The application of ZEUS agents in manufacturing environments, in *Proceedings of the 12th International Workshop on Database and Expert Systems Applications,* 628-632.
43. Gou, L., Hasegawa, T., Luh, P. B., Tamura, S. and Oblak, J. M. (1994) Holonic planning and scheduling for a robotic assembly testbed, in *Proceedings of the 4th International Conference on Computer Integrated Manufacturing and Automation Technology*, 142-149.
44. Gou, L., Luh, P. B. and Kyoya, Y. (1998) Holonic manufacturing scheduling: architecture, cooperation, mechanism, and implementation. *Computers in Industry* 37, 213-231.
45. Hammerle, A., Geurts, R. and Auinger, F. (2000) Internal communication architecture for resource holons, in *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, Vol. 3, 1721-1726.
46. Hatvany, J. (1985) Intelligence and cooperation in heterarchical manufacturing systems. *Annals of CIRP,* Vol. 14, No. 1, 5-10.
47. Heikkila, T., Kollingbaum, M., Valckenaers, P. and Bluemink, G.-J. (2001) An agent architecture for manufacturing control: manAge. *Computers in Industry*, 46, 315-331.
48. Heikkila, T., Rannanjarvi, L., Sallinen, M., Rintala, M. (2003) A holonic shot-blasting system, in *Deen, S. M. (ed.), Agent-Based Manufacturing: Advances in the Holonic Approach*, Springer-Verlag, Berlin, 255-302.

49. Heragu, S. S., Graves, R. J., Kim, B.-I. and St. Onge, A. (2002) Intelligent agent based framework for manufacturing systems control. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans,* Vol. 32, No. 5, 560-573.
50. Hirose, M., Ishii, T. and Ikei, Y. (1986) Development of holonic manipulator, in *Proceedings of the Japan-USA Symposium on Flexible Automation*, 269-273.
51. Hirose, M., Ishii, T., Ikei, Y. and Amari, H. (1988) Software environment for holonic manipulator. *Advances in Flexible Automation and Robotics,* Vol. 1, 317-324.
52. Hirose, M. (1990) Development of the holonic manipulator and its control, in *Proceedings of the 29th IEEE Conference on Decision & Control,* 91-96.
53. Hsieh, F.-S. (2002) Modeling and control of holonic manufacturing systems based on extended contract net protocol, in *Proceedings of the American Control Conference,* 5037-5042.
54. Hsieh, F.-S. (2004) Modeling and control holonic manufacturing systems based on fusion of contract net and Petri nets. *Automatica*, 40, 51-57.
55. Huang, B., Gou, H., Liu W., Li, Y. and Xie, M. (2002) A framework for the virtual enterprise control with the holonic manufacturing paradigm. *Computers in Industry,* 49, 299-310.
56. d'Inverno, M. and Luck, M. (2001) *Understanding agent systems*, Springer-Verlag, Berlin.
57. Jarvis, D. H. and Jarvis, J. H. (2003) Holonic diagnosis for an automotive assembly line, in *Deen, S. M. (ed.), Agent-Based Manufacturing: Advances in the Holonic Approach*, Springer-Verlag, Berlin, 193-206.
58. Jarvis, J., Jarvis, D. and McFarlane, D. (2003) Achieving holonic control - an incremental approach. *Computers in Industry,* 51, 211-223.
59. Jennings, B., Brennan, R., Gustavsson, R., Feldt, R., Pitt, J., Prouskas, K. and Quantz, J. (1999) FIPA-compliant agents for real-time control of intelligent network traffic. *Computers Networks,* 31, 2017-2036.
60. Johnson, C. A. (2003) Towards a formalized HMS model, in *Deen, S. M. (ed.), Agent-Based Manufacturing: Advances in the Holonic Approach*, Springer-Verlag, Berlin, 193-206.
61. Joshi, S. B. and Smith, J. S. (1994) *Computer control of flexible manufacturing systems*, Chapman & Hall, London.
62. Karageorgos, A., Mehandjiev, N., Weichhart, G. and Hammerle, A. (2003) Agent-based optimization of logistics and production planning. *Engineering Applications of Artificial Intelligence* 16, 335-348.
63. Kim, B.-Y., Heragu, S. S., Graves, R. J. and St. Onge, A. (2004) Intelligent agent based control for warehouse control, in *Proceedings of the IEEE 37th Hawaii International Conference on System Sciences,* 1-10.
64. Koestler, A. (1968) *The ghost in the machine*, The Macmillan Company, New York.
65. Kretchmar, R. M. (2002) Parallel reinforcement learning, in *Proceedings of the Sixth World Conference on Systemics, Cybernetics, and Informatics,* Orlando, Florida.
66. Law, A. M. and Kelton, W. D. (2000) *Simulation Modeling and Analysis* 3rd edition, McGraw-Hill Co, Inc., Boston.
67. Lee, C.-Y., Lei, L. and Pinedo, M. (1997) Current trends in deterministic scheduling. *Annals of Operations*

Research, 70, 1-41.

68. Leitao, P. and Restivo, F. (2000) A framework for distributed manufacturing applications, in *Proceedings of the Annual Conference of ICIMS-NOE, Bordeaux, France.*
69. Liu, S., Gruver, W. A., Kotak, D. and Bardi, S. (2000) Holonic manufacturing system for distributed control of automated guided vehicles, in *Proceedings of the 2000 IEEE International Conference on Systems, Man, and Cybernetics*, Vol. 3, 1727-1732.
70. Lun, M. and Chen, F. F. (2000) Holonic concept based methodology for part routing on flexible manufacturing systems. *The International Journal of Advanced Manufacturing Technology,* 16, 483-490.
71. Marik, V. and Pechoucek, M. (2001) Holons and agents: recent developments and mutual impacts, in *Twelfth International Workshop on Database and Expert Systems Applications,* IEEE Computer Society, 605-607.
72. Marik, V., Pechoucek, M., Vrba, P. and Hrdonka, V. (2003) FIPA standards and Holonic Manufacturing, in *Deen, S. M. (ed.), Agent-Based Manufacturing: Advances in the Holonic Approach*, Springer-Verlag, Berlin, 89-119.
73. Mathews, J. (1995) Organizational foundations of intelligent manufacturing systems - the holonic viewpoint. *Computer Integrated Manufacturing Systems,* Vol. 8, No. 4, 237-243.
74. Mathews, J. (1996) Organizational foundations of object-oriented programming. *Journal of Systems and Software,* 34, 247-253.
75. Maturana, F., Shen, W. and Norrie, D. H. (1999) MetaMorph: an adaptive agent-based architecture for intelligent manufacturing, *International Journal of Production Research,* 37(10), 2159-2174.
76. McFarlane, D. C. and Bussmann. S. (2000) Developments in holonic production planning and control. *International Journal of Production Planning and Control*, Vol. 11, No. 6, 522-536.
77. McFarlane, D. C. and Bussmann, S. (2003) Holonic manufacturing control: rationales, developments and open issues, in *Deen, S. M. (ed.), Agent-Based Manufacturing: Advances in the Holonic Approach*, Springer-Verlag, Berlin, 303-326.
78. Mehrabi, M. G., Ulsoy, A. G. and Koren, Y. (1998) Reconfigurable manufacturing systems: key to future manufacturing, in *1998 Japan-USA Symposium on Flexible Automation,* Otsu, Japan.
79. Mehrabi, M. G., Ulsoy, A. G. and Koren, Y. (2000) Reconfigurable manufacturing systems: key to future manufacturing. *Journal of Intelligent Manufacturing,* 11 (4), 403-419.
80. Meystel, A. M. and Albus, J. S. (2002) *Intelligent Systems, Architecture, Design and Control,* John Wiley & Sons, Inc., New York.
81. Mill, F. and Sherlock, A. (2000) Biological analogies in manufacturing. *Computers in Industry* 43, 153-160.
82. Mitidieri, C., Pereira, C. E. and Penz, L. L. (2002) Real time issues on coordinating cooperative autonomous objects, in *Proceedings of the IEEE Seventh International Workshop on Object-Oriented Real-Time Dependable Systems*, 131-138.
83. Monostori, L. and Kadar, B. (1999) Agent-based control of manufacturing systems, in *Proceedings of the*

Second International Conference on Intelligent Processing and Manufacturing of Materials, Volume: 1, 131-137.

84. Monostori, L. (2003) AI and machine learning techniques for managing complexity, changes and uncertainties in manufacturing, *Engineering Applications of Artificial Intelligence* 16, 277-291.
85. National Research Council (1998) *Visionary Manufacturing Challenges for 2020 Report*, Committee on Visionary Manufacturing Challenges, Board of Manufacturing and Engineering Design, Commission on Engineering and Technical System of the National Research Council, National Academy of Sciences.
86. Neligwa, T. and Fletcher, M. (2003) An HMS operational model, in *Deen, S. M. (ed.), Agent-Based Manufacturing: Advances in the Holonic Approach*, Springer-Verlag, Berlin, 163-191.
87. Ouelhadj, D., Hanachi, C., Bouzouia, B., Moualek, A. and Farhi, A. (1999) A multi-contract net protocol for dynamic scheduling in flexible manufacturing systems, in *Proceedings of the 1999 IEEE International Conference on Robotics & Automation,* 1114-119.
88. Overmars, A. H. and Toncich, D. J. (1996) Hybrid FMS control architectures based on holonic principles. *The International Journal of Flexible Manufacturing Systems,* 8 (3), 263-278.
89. Paolucci, M. and Sacile, R. (2005) *Agent-Based Manufacturing and Control Systems: New Agile Manufacturing Solutions for Achieving Peak Performance*, CRC Press, Boca Raton, Florida.
90. Parunak, V. D. (1987) Manufacturing experience with the contract net. *Distributed Artificial Intelligence,* 285-310.
91. Pendharkar, P. C. (1999) A computational study on design and performance issues of multi-agent intelligent systems for dynamic scheduling environments. *Expert Systems with Applications,* 16, 121-133.
92. Pinedo, M. (2002) *Scheduling theory, algorithms, and systems*, Prentice-Hall, Inc., Upper Saddle River, New Jersey.
93. Rahimifard, S., Newman, S. T. and Bell, R. (1999) Distributed autonomous real time planning and control of SME's. *Journal of Engineering Manufacture: Proceedings of the Institution of Mechanical Engineering Part-B,* Vol. 213.
94. Ramos, C., (1996) A holonic approach for task scheduling in manufacturing systems, in *Proceedings of the 1996 IEEE International Conference on Robotics and Automation,* 2511-2516.
95. Rannanjarvi, L. and Heikkila, T. (1998) Software development for holonic manufacturing systems. *Computers in Industry,* 37, 233-253.
96. Ritter, A., Braatz, A., Hopf, M., Schaeffer, C. and Westkamper, E. (2003) Transport agents - specification and development, in *Deen, S. M. (ed.), Agent-Based Manufacturing: Advances in the Holonic Approach*, Springer-Verlag, Berlin, 241-253.
97. Russell, S. J. and Norvig, P. (2003) *Artificial intelligence: a modern approach, Second edition,* Pearson Education, Inc., Upper Saddle River, New Jersey.
98. Ryu, K., Shin, M. and Jung, M. (2001) A methodology for implementing agent-based controllers in the fractal manufacturing systems, in *Proceedings of the 5th Conference of Engineering Design and Automation,* 91-96.

99. Ryu, K., Son, Y. and Jung, M. (2003) Modeling and specifications of dynamic agents in fractal manufacturing systems. *Computers in Industry,* 52, 161-182.

100. Saad, A., Kawamura, K., Biswas, G., Johnson, M. E. and Salama, A. (1995) Evaluating a contract net-based heterarchical scheduling approach for flexible manufacturing, in *Proceedings of the IEEE International Symposium on Assembly and Task Planning*, 147-152.

101. Sallans, H. and Hinton, G. (2001) Using free energies to represent q-values in a multi-agent reinforcement learning task. *Advances in Neural Information Processing Systems,* Vol. 13, MIT Press.

102. Schoop, R., Neubert, R. and Colombo, A. W. (2001) A multiagent-based distributed control platform for industrial flexible production systems, in *Proceedings of the 27th Annual Conference of the IEEE Industrial Electronics Society,* Vol. 1, 279-284.

103. Sedgewick, R. (1998) *Algorithms in C++, part 1-4: fundamentals, data structures, sorting, searching,* Addison-Wesley Publishing Company, Inc., Boston.

104. Shen, W., Xue, D. and Norrie, D. H. (1997) An agent-based manufacturing enterprise infrastructure for Distributed integrated intelligent manufacturing systems, in *Proceedings of the Third International Conference on the Practical Application of Intelligent Agents and Multi-Agents,* 1-16.

105. Shen, W., Norrie, D. H. and Barthes, J.-P. A. (2001) *Multi-agent systems for concurrent intelligent design and manufacturing,* Taylor & Francis Inc., London.

106. Shen, W. (2002) Distributed manufacturing scheduling using intelligent agents. *Intelligent Systems*, Vol. 17, Issue 1, 88-94.

107. Shu, S., Wilkes, M. and Kawamura, K. (2000) Development of reusable, configurable, extensible holonic manufacturing system, in *2000 IEEE International Conference on Systems, Man, and Cybernetics*, Vol. 3, 8-11.

108. Silva, N. and Ramos, C. (1999) Infrastructures and scheduling method for holonic manufacturing systems, in *Proceedings of the 1999 IEEE International Symposium on Assembly and Task Planning,* 442-447.

109. Smith, R. G. (1980) The contract net protocol: high-level communication and control in a distributed problem solver. *IEEE Transactions on Computers,* Vol. C-29, No. 12, 1104-1113.

110. Smith, J. S., Peters, B. A. and Sirinvasan A. (1999) Job shop scheduling considering material handling. *International Journal of Production Research,* Vol. 37, No. 7, 1541-1560.

111. Sousa, P. and Ramos, C. (1998) A dynamic scheduling holon for manufacturing orders. *Journal of Intelligent Manufacturing,* 9, 107-112.

112. Sousa, P. and Ramos, C. (1999a) Distributed task scheduling, in *Proceedings of the 1999 IEEE International Symposium on Assembly and Task Planning,* 436-441.

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

114. Sugimura, N., Hino, R. and Moriwaki, T. (2001) Integrated process planning and scheduling in holonic manufacturing systems, in *Proceedings of the 4th IEEE International Symposium on Assembly and Task Planning*, 250-255.

115. Sun, J. and Xue, D. (2001) A dynamic reactive scheduling mechanism for responding to changes of production orders and manufacturing resources. *Computers in Industry,* 46, 189-207.

116. Sutton, R. S. and Barto, A. G., (1998) *Reinforcement learning: An Introduction*, Bradford Books, MIT Press, Cambridge, MA.

117. Talukdar, S. N. (1999) Collaboration rules for autonomous software agents. *Decision Support Systems,* 24, 269-278.

118. Tamura, S., Toshibumi, S. and Hasegawa, T. (2003) HMS development and implementation environments, in *Deen, S. M. (ed.), Agent-Based Manufacturing: Advances in the Holonic Approach*, Springer-Verlag, Berlin, 209-240.

119. Tharumarajah, A., Wells, A. J. and Nemes, L. (1996) Comparison of the bionic, fractal and holonic manufacturing system concepts. *International Journal of Computer Integrated Manufacturing,* Vol. 9, No. 3, 217-226.

120. Toh, K. T. K., Newman, S. T. and Bell, R. (1998) An information systems architecture for small metal-working companies. *Journal of Engineering Manufacture: Proceedings of the Institute of Mechanical Engineers,* Vol. 212, No. 2.

121. Toh, K. T. K., Harding, J. A. and Thompson, D. (1999) An holonic approach for the modeling of enterprise functionality and behavior. *International Journal of Computer Integrated Manufacturing,* Vol. 12, No. 6, 541-558.

122. Ulieru, M. and Norrie, D. (2000) Fault recovery in distributed manufacturing systems by emergent re-configuration: A fuzzy multi-agent modeling approach. *Information Sciences,* 127, 101-123.

123. Ulieru, M., Stefanoiu, D. and Norrie, D. (2000) Holonic self-organization of multi-agent systems by fuzzy modeling with application to intelligent manufacturing, in *Proceedings of the 2000 IEEE International Conference on Systems, Man, and Cybernetics*, Vol. 3, 1661-1666.

124. Ulieru, M. (2001a) The holonic enterprise as a collaborative information ecosystem, *FIPA Meeting,* London, UK.

125. Ulieru, M. (2001b) Web-centric diagnosis and prediction system for global manufacturing, in *Joint 9th IFSA World Congress and 20th NAFIPS International Conference,* Vol. 1.

126. Ulieru, M., Walker, S. S. and Brennan, R. W. (2001) The holonic enterprise as a collaborative information ecosystem, in *Proceedings of the Workshop on Holons: Autonomous and Cooperating Agents for Industry,* Montreal, Canada.

127. Unver, H. O. and Anlagan, O. (2000) Design and implementation of an agent-based shop floor control system using Windows-DNA. *International. Journal of Computer Integrated Manufacturing.* Available online at http://www.me.metu.edu.tr/imtrg/Papers/ijcimpaper.pdf (Accessed: June 4, 2005).

128. Usher, J. M. and Wang, Y.-C. (2000) Intelligent agents architectures for manufacturing control, in *Proceedings of the 2000 Industrial Engineering Research Conference,* Cleveland, Ohio.

129. Valckenaers, P., Van Brussel, H., Wyns, J., Bongaerts, L. and Peeters, P. (1998) Designing holonic manufacturing systems. *Robotics and Computer-Integrated Manufacturing,* 14, 455-464.

130. Van Brussel, H., Wyns, J., Valckenaers, P., Bongaerts, L. and Peeters, P. (1998) Reference architecture for holonic manufacturing systems: PROSA. *Computers in Industry,* 37, 255-274.

131. Vancza, J. and Markus, A. (2000) An agent model for incentive-based production scheduling. *Computers in Industry,* 43, 173-187.

132. Veeramani, D., Bhargava, B. and Barash, M. M. (1993) Information system architecture for heterarchical control of large FMSs. *Computer Integrated Manufacturing Systems,* 6(2), 76-92.

133. Vrba, P. and Hrdonka, V. (2001) Material handling problem: FIPA compliant agent implementation, in *Proceedings of the 12th International Workshop on Database and Expert Systems Applications*, 635-639.

134. Wang, L., Balasubramanian, S., Norrie, D. H. and Brennan, R. W. (1998a) Agent-based control system for next generation manufacturing, in *Proceedings of the 1998 IEEE International Symposium on Intelligent Control,* 78-83.

135. Wang, L., Balasubramanian, S. and Norrie, D. H. (1998b) Agent-based intelligent control system design for real-time distributed manufacturing environments, in *Proceedings of the Agent-based Manufacturing Workshop – Autonomous Agents '98,* 152-159.

136. Wang, L. (2001) Integrated design-to-control approach for holonic manufacturing systems. *Robotics and Computer Integrated Manufacturing,* 17, 159-167.

137. Wang, Y.-C. and Usher, J. M. (2004) A reinforcement learning approach for developing routing policies in multi-agent production scheduling, in *Proceedings of the 2004 Industrial Engineering Research Conference,* Houston, Texas.

138. Warnecke, H. J. (1993) *The fractal company,* Springer-Verlag, Berlin.

139. Weiss, G. (1996) Adaptation and learning in multi-agent systems: some remarks and a bibliography, *in Weiss G. and Sen, S. (ed.), Adaptation and Learning in Multi-Agent Systems* Springer-Verlag, Berlin.

140. Wullink, G., Giebels, M. M. T. and Kals, H. J. J. (2002) A system architecture for holonic manufacturing planning and control (EtoPlan). *Robotics and Computer Integrated Manufacturing,* 18, 313-318.

141. Zhang, X. and Norrie, D. H. (1999) Dynamic reconfiguration of holonic lower level control, in *Proceedings of the Second International Conference on Intelligent Processing and Manufacturing of Materials*, Vol. 2, 887-893.

142. Zhang, X., Balasubramanian, S., Brennan, R. W. and Norrie, D. H. (2000) Design and implementation of a real-time holonic control system for manufacturing. *Information Sciences,* 127, 23-44.

143. Zhang, X., Brennan, R. W., Yuefei, X. and Norrie, D. N. (2001) Runtime adaptability of a concurrent function block model for a real-time holonic controller, in *IEEE International Conference on Systems, Man, and Cybernetics,* Vol. 1, 164-168.

144. Zhang, W. and Dietterich, T. G. (1995) A reinforcement learning approach to job-shop scheduling, in *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pp. 1114-1120.

145. Zhang, W. and Dietterich, T. G. (1996) High-performance job-shop scheduling with a time-delay TD(λ) network, in *Proceedings of the 1995 Conference in Advances in Neural Information Processing Systems*, MIT Press, Cambridge, MA, pp. 1024-1030.

Appendices

Appendix A. Pseudocodes for the Job Evaluation, Allocation, and Execution Algorithms

The following notations are used in the pseudocodes of the holonic job evaluation, allocation, and execution algorithm besides the already listed notations in the corresponding chapters where these algorithms are presented:

- *ST* = System time taken from the internal clock
- *CT* = Checks the actual time
- *Offers* = Number of offers received by a OH from the MHHs in the system
- *Requests* = Number of requests received by a MHH from the OHs in the system
- *List* = List of jobs to be processed by an individual MHH
- *min(STP(j))*: Sorts the offers received by a OH in the increasing order of their MHH-ECT values
- *min(MHH-OP(j))*: Sorts the offers received by a OH in the increasing order of their MHH-OP values
- *min(MHH(J))*: Sorts the offers received by a OH in the increasing order of their number of jobs already assigned to a particular MHH
- *Random( )*: Sorts the argument randomly
- *min(STP(j))*: Sorts the requests received by a MHH in the increasing order of their starting time of the processing operations
- *max(OH-ECT(j))*: Sorts the requests received by a MHH in the decreasing order of their OH-ECT values
- *max(OH-OP(j))*: Sorts the requests received by a MHH in the decreasing order of their OH-OP values
- *Counter* = Measures how many times the program goes through a specific step

A.1. Order Holon Evaluation and Allocation Algorithm

```
Start
Initialize c ← 0 and r ← 0
Set d(f) = d(c)
Calculate OH-ECT(j) and OH-OP(j)
Communication (RE: MHH, GS)
Calculate ST
While (ST < d(f))
  Re-calculate ST
Communication (SE: MHH)
If (ST = d(f))
   If (Offers = 0)
      d(c+1) = d(c) + r
   OH Ranking Procedure ()
   Communication (RE: MHH)
   Communication (RE: GS)
   Stop
```

A.1.1. Order Holon Ranking Procedure Algorithm

```
Start
Calculate min(MHH-ECT(j))
If (Tie)
   Calculate min(MHH-OP(j))
   If (Tie)
      Calculate min(MHH(J))
      If (Tie)
         Random (MHH)
Select MHH
Stop
```

A.2. Material Handling Holon Evaluation Algorithm

```
Start
Set T ← constant
Calculate ST
Set CT = ST
Calculate ST
While (ST < CT + T)
   Re-calculate ST
Communication (SE: OH)
If (ST = CT + T)
   CT = ST
   If (Requests > 0)
      MHH Ranking Procedure ()
      While (List > 0)
         Select and Remove the First Job on the List
         Calculate MHH-ECT(j) and MHH-OP(j)
         Calculate ST
         If (ST • d(f))
            Communication (RE: OH)
            OH Evaluation and Allocation ()
            Follow SLS
      Follow SLS
   Follow SLS
```

A.2.1. Material Handling Holon Ranking Procedure Algorithm

```
Start
Calculate min(STP(j))
If (Tie)
   Calculate max(OH-ECT(j))
   If (Tie)
      Calculate max(OH-OP(j))
```

```
      If (Tie)
         Random (Job)
Select Job
Stop
```

A.3. Individual Schedules Generation Algorithm

```
Start
Initialize k ← 0
Set k ← k + 1
Set CTP(0) = UT(0) = CTU(0) = 0
While (k < K)
   If (STP(k) – CTP(k–1) < LT(k))
      Move k to the Right with T(1) = CTP(k–1) + LT(k) – STP(k)
   Insert LT(k) to the Left of k
   If (STL(k+1) – CTP(k) < UT(k))
      Move k+1 to the Right with T(2) = CTP(k) + UT(k) – STL(k+1)
   Insert UT(k) to the Right of k
   k ← k + 1
If (STP(k) – CTP(k–1) < LT(k))
   Move k to the Right with T(1) = CTP(k–1) + LT(k) – STP(k)
Insert LT(k) to the Left of k
Insert UT(k) to the Right of k
Stop
```

A.4. System Level Schedule Generation Algorithm

```
Start
Communication (SE: SMD)
Initialize M ← constant, and MH ← constant
Initialize OH ← 0
```

```
Communication (SE: SMD)
No new OH?
   Generate SLS
If (OH > Existing OH)
   OH ← OH + 1
   Communication (SE: OH)
   While (IS < OH)
      Wait
   Communication (SE: MHH)
   If (IS = OH)
      Generate SLS
      Communication (RE: SMD)
      Stop
```

A.5. System Level Schedule Constraint Satisfaction Algorithm

```
Start
Job Precedence Constraint ()
Operation Precedence Constraint ()
MH Resource Constraint ()
Stop
```

A.5.1. Algorithm for Satisfying Job Precedence Constraints on Machines

```
Start
Initialize m ← 0
Set m ← m + 1
While (m • M)
   CTU(0) = 0
   Initialize j ← 0
   Set j ← j + 1
```

```
   While (j • J)
      If (STL(j) – CTU(j–1) < 0)
         Move j to the Right with T(3) = CTU(j–1) – STL(j)
      j ← j + 1
   m ← m + 1
Stop
```

A.5.2. Algorithm for Satisfying Operation Precedence Constraints

```
Start
Initialize j ← 0
Set j ← j + 1
While (j • OH)
   Initialize k ← 0
   Set k ← k + 1
   While (k • K)
      Set CTU(0) = 0
      If (STL(k) – CTU(k–1) < 0)
         Move k to the Right with T(4) = CTU(k–1) – STL(k)
      k ← k + 1
j ← j + 1
Stop
```

A.5.3. Algorithm for Satisfying Material Handling Resource Constraints

```
Start
Initialize T ← –1, MT ← max(C(j))
Set T ← T + 1
While (T • MT)
```

```
   Initialize mh ← 0
   Set mh ← mh + 1
      While (mh • MH)
      Set OP = Σ(#ST(L/U)(k))
      If (OP > 0)
         If (mh idle?)
            If (OP > 1)
               GS Ranking Procedure ()
         Set (ST(L/U)(k)) = T + 1
         Set MT ← MT + 1
         Job Precedence Constraint ()
         Operation Precedence Constraint ()
      mh ← mh + 1
T ← T + 1
Stop
```

A.5.4. Global Scheduler Ranking Procedure Algorithm

```
Start
If (CTU(q-1) = STL(q))
   Set T = STL(q)
Else If (CTU(q-1) • STL(q))
   Set RPJ(j) = Σ(PT+LT+UT)(j)
   Set RPM(m) = Σ(PT+LT+UT)(m)
   Calculate all (T + RPJ(j) and T + RPM(m))
   Set Counter ← 0
   If ((T + RPJ(j) = MT) or (T + RPM(m) = MT))
      Counter ← Counter + 1
   If (Counter = 0 or Counter > 1)
      Calculate max(CTL(L/U)(k-1) - ST(L/U)(k))
      Calculate max(CTL(L/U)(j-1) - ST(L/U)(j))
      If (Tie)
         Random Job ()
```

```
   Set ST(L/U)(1) = T
Set all ST(L/U) = T + 1
Stop
```

A.6. Global Scheduler Decision Algorithm

```
Start
Generate SLS ()
If (UF(Global Schedule) > UF(SLS))
   Set SLS = Global Schedule
Communication (RE: SMD)
Stop
```

A.7. Inter-Material Handling Holons Cooperation Algorithm

```
Start
Initialize T ← -1, MT = C(max)
Set T ← T + 1
While (T • MT)
   Set CTP(0) = CTU(0) = 0
   If (STU(j(k)) - CTP(j(k)) • 0)
      If (STL(j(k)) - CTU((j-1)(k)) > 0)
         If (STL(j(k)) - CTU(j(k-1)) > 0)
   Calculate MHH-HOP
   Communication (RE: MHH)
   Calculate ST
   Communication (SE: MHH)
   If (Offers > 0)
      Award Operation
      Communication (RE: MHH)
      Communication (RE: GS)
   If (ST = d(k))
```

```
      Follow SLS
T ← T + 1
Stop
```

A.8. Order Holon Job Completion Algorithm

```
Start
Communication (SE: SMD)
Communication (SE: SMD)
If (Resource Breakdown?)
   Re-send requests
   Communication (RE: MHH)
Set P ← 0
Calculate ST
While (ST < C(j))
   Re-calculate ST
Communication (SE: MHH)
If (MHH Job Execution Not Received?)
   Communication (RE: MHH)
   Communication (SE: MHH)
   Set C(j) = new C(j)
   Set P = w(j)(ACTU(j) - CTU(j))
Calculate MHH Acc. = MHH-HOP - P
Communication (RE: SMD)
Stop
```

A.9. Inter-Material Handling Holons Sub-Contracted Operations Completion Algorithm

```
Start
Set P ← 0
```

```
Calculate ST
While (ST < C(k))
   Re-calculate ST
Communication (SE: MHH)
If (MHH Sub-Contractor Job Execution Not Received)
   Communication (RE: MHH)
   Set P = w(j)(ACTU(k) – CTU(k))
   Communication (RE: MHH)
Calculate MHH Acc. = MHH-HOP – P
Communication (RE: SMD)
Stop
```

Appendix B. Pseudocodes for the Algorithms Used in the Performance Evaluation Process

The following notations are used in the pseudocodes for the algorithm used in the performance evaluation process besides the already listed notations in the corresponding chapters where these algorithms are presented:

- *L(mh)* = Number of loading operations to be executed by MH, *mh*
- *U(mh)* = Number of unloading operations to be executed by MH, *mh*
- *(MH)C(op)* = Number of possible combinations of scheduling the MH operations
- *min(IC(max)(N)*: Sorts the *IC(max)* values in increasing order
- *MIN(s1):* The minimum of the *IC(max)* values at a particular time
- *max(C(j)(jb)*): Sorts the jobs in the decreasing order of their completion time
- *max(PT(k(j))(jb)):* Sorts the operations in the decreasing order of their next value of their processing time
- *Random( ):* Sorts the argument randomly
- *max(CTP(j(m))(mc)):* Sorts operations in the decreasing order of their corresponding job completion time on any machine
- *max(C(max)(j(m))(mc)):* Sorts operations in the decreasing order of their corresponding job makespan values on any machine
- *min(V(j(m))(mc)):* Sorts operations in the increasing order of the number of violations generated by their selection
- *max(PT(k(m))(mc)):* Sorts operations in the decreasing order of their next value of their processing time

B.1. Algorithmic Lower Bound

The first part of the pseudocode which assures that the loading and unloading operations are inserted in the schedule and the operations precedence constraints on machines is satisfied is presented below:

```
Initialize m ← 0
   Set m ← m + 1
      While (m ≤ M)
         Initialize CTP(0) ← 0, UT(0) ← 0, CTU(0) ← 0
         Initialize j ← 0
         Set j ← j + 1
            While (j < J)
               If (STP(j) – CTP(j-1) < LT(j))
                   Move all op to the right with
                   T(5) = CTP(j-1) + LT(j) – STP(j)
               Insert LT to the left of j
                 If (STL(j+1) – CTP(j) < UT(j))
                  Move all op to the right with
                  T(6) = CTP(j) + UT(j) – STL(j+1)
               Insert UT to the right of j
           j ← j + 1
           If (j = J)
              If (STP(j) – CTP(j-1) < LT(j))
                 Move all op to the right with
                 T(5) = CTP(j-1) + LT(j) – STP(j)
              Insert LT to the left of j
              Insert UT to the right of j
      m ← m + 1
```

The second part of the pseudocode which assures that the operation precedence constraints for all jobs are satisfied, while not disrupting the operation precedence on machines is presented below:

```
Initialize Counter1 ← 0 and Counter2 ← 0
   Do {
      Set Counter 2 = Counter 1
      Initialize j ← 0
      Set j ← j + 1
      While (j ≤ J)
```

```
        Initialize k ← 0
        Set k ← k + 1
        While (k ≤ K)
           Initialize CTU(0) ← 0, STL(0) ← 0
           If (STL(k) – CTU(k–1) < 0)
              Move all op to the right with
              T(7) = CTU(k–1) – STL(k)
              Set Counter1 = Counter1 + 1
           If (STL(k) – CTU(k–1) > 0)
              Move all op to the left with
              T(8) = STL(k) – CTU(k–1)
              Set Counter1 = Counter1 + 1
        k ← k + 1
    j ← j + 1
  } While Counter2 not equal with Counter1
```

B.2. Enhanced Best-First Search Algorithm

```
Start with the LB3 schedule
Initialize t ← 0
Set t ← t + 1
While (t ≤ T)
   If (Σ(L(mh)+U(mh)) < MH)
       Counter ← 0
      Calculate N = (MH)C(op)
      If ((STL(k(j))(jb) – (CTU(k(j)–1))(jb) < 0)
         Move the op to the right with T(9) = (STL(k(j))(jb) + 1
         Counter = Counter + 1
      If ((STL(k(m))(mc) – (CTU(k(m)–1))(mc) ≥ 0)
         Move the op to the right with T(10) = (STL(k(m))(mc) + 1
         Counter = Counter + 1
      If (Counter > 0)
         Calculate IC(max)(N) = EC(max)(N) + PC(max)(N)
```

```
        Calculate min(IC(max)(N))
           Set MIN(s1) = min(IC(max)(N))
           Move k(j) to the right with
                          T(11) = (STL(U)(k(j)(jb)) + 1
           Re-calculate IC(max)(s1)
           If (MIN(s1) < MIN(s2))
              Set MIN(s1) = MIN(s2)
t ← t + 1
```

B.3. Job Completion Time Heuristic Algorithm

```
Start with the LB3 schedule
Initialize t ← 0
Set t ← t + 1
While (t ≤ T)
    If ((STL(k(j))(jb) – (CTU(k(j)-1))(jb) < 0)
       Move the op to the right with
       T(12) = (CTU(k(j)-1))(jb)- (STL(k(j))(jb)
    If ((STL(k(m))(mc) – (CTU(k(m)-1))(mc) ≥ 0)
       Move the op to the right with
       T(13) = (CTU(k(m)-1))(mc) – (STL(k(m))(mc)
    If (Σ(L(mh)+U(mh))> MH)
       Calculate max(C(j)(jb))
        If (Tie)
           Calculate max(PT(k(j))(jb))
           If (Tie)
              Random (k(j))
        Move k(j) to the Right with T(14)= (ST(L/U)(k(j)(jb)) + 1
t ← t + 1
```

B.4. Machine Completion Time Heuristic Algorithm

```
Start with the LB3 schedule
Initialize t ← 0
Set t ← t + 1
While (t ≤ T)
    If ((STL(k(j))(jb) – (CTU(k(j)–1))(jb) < 0)
       Move the op to the right with
       T(12) = (CTU(k(j)–1))(jb)– (STL(k(j))(jb)
    If ((STL(k(m))(mc) – (CTU(k(m)–1))(mc) ≥ 0)
       Move the op to the right with
       T(13) = (CTU(k(m)–1))(mc) – (STL(k(m))(mc)
    If (Σ(L(mh)+U(mh))> MH)
       Calculate max(CTP(j(m))(mc))
        If (Tie)
            Calculate max(C(max)(j(m))(mc))
            If (Tie)
               Random (k(m))
         Move k(j) to the Right with T(15)= (ST(L/U)(k(m)(mc)) + 1
t ← t + 1
```

B.5. Material Handling Resource Constraint Violation Heuristic Algorithm

```
Start with the LB3 schedule
Initialize t ← 0
Set t ← t + 1
While (t ≤ T)
    If ((STL(k(j))(jb) – (CTU(k(j)–1))(jb) < 0)
       Move the op to the right with
       T(12) = (CTU(k(j)–1))(jb)– (STL(k(j))(jb)
    If ((STL(k(m))(mc) – (CTU(k(m)–1))(mc) ≥ 0)
       Move the op to the right with
```

$T(13) = (CTU(k(m)-1))(mc) - (STL(k(m))(mc)$

If $(\Sigma(L(mh)+U(mh)) > MH)$

Calculate $max(V(j(m))(mc))$

If (Tie)

Calculate $max(PT(k(m))(mc))$

If (Tie)

$Random\ (k(m))$

Move $k(j)$ to the Right with $T(16) = (ST(L/U)(k(m)(mc)) + 1$

$t \leftarrow t + 1$

Appendix C. Schedules Generated when Replicating Dynamic Manufacturing Environments

C.1. Schedules generated at time T = 0

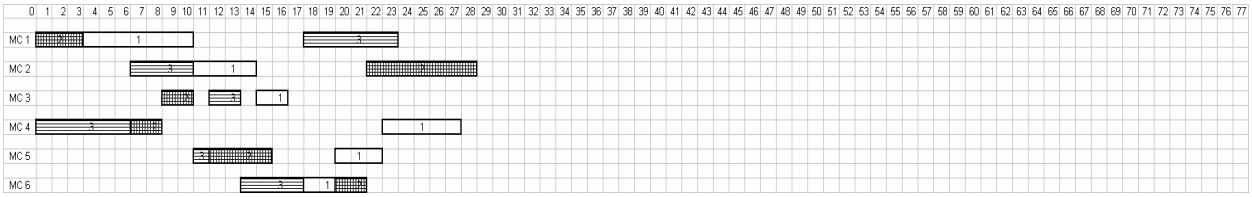

Figure C.1. Job schedule at time T = 0, when MH operations are not considered.

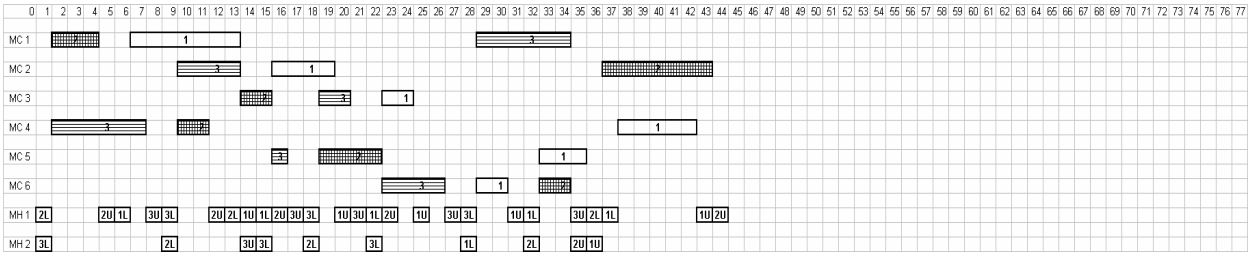

Figure C.2. Lower bound on makespan at time T = 0.

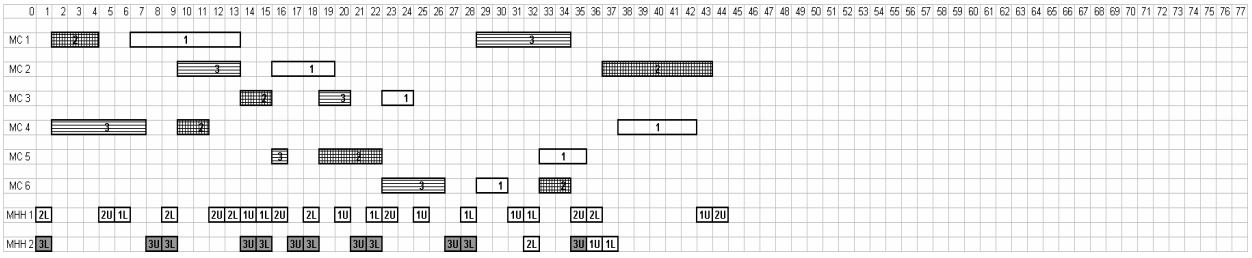

Figure C.3. Final schedule at time T = 0, when considering MH operations.

C.2. Schedules generated at time T = 3

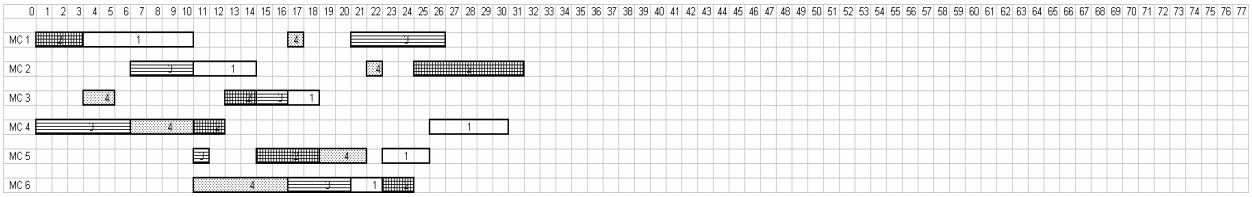

Figure C.4. Job schedule at time T = 3, when MH operations are not considered.

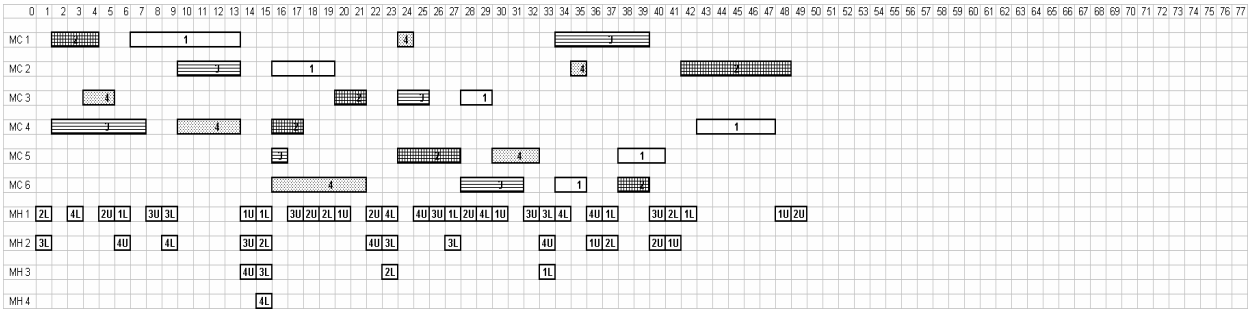

Figure C.5. Lower bound on makespan at time T = 3.

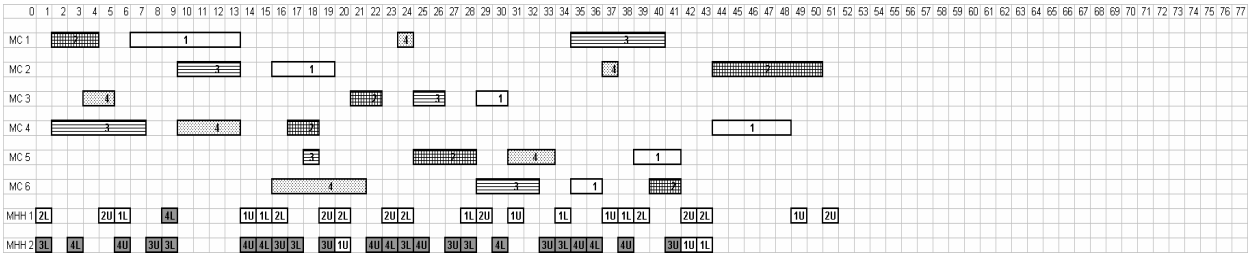

Figure C.6. Final schedule at time T = 3, when considering MH operations.

C.3. Schedules generated at time T = 6

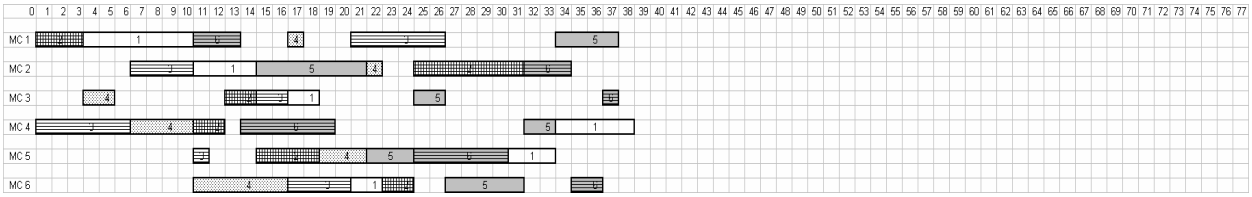

Figure C.7. Job schedule at time T = 6, when MH operations are not considered.

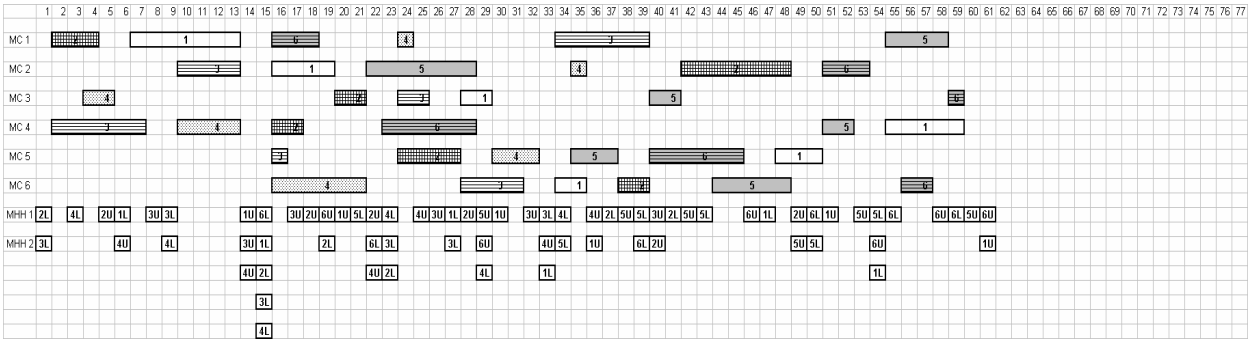

Figure C.8. Lower bound on makespan at time T = 6.

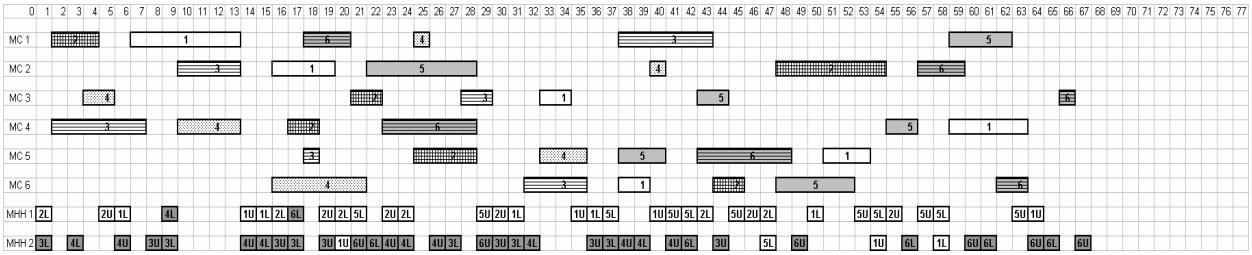

Figure C.9. Final schedule at time T = 6, when considering MH operations.

C.4. Schedules generated at time T = 10

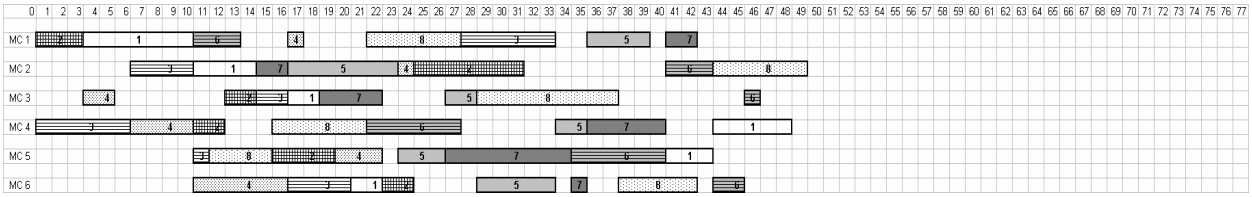

Figure C.10. Job schedule at time T = 10, when MH operations are not considered.

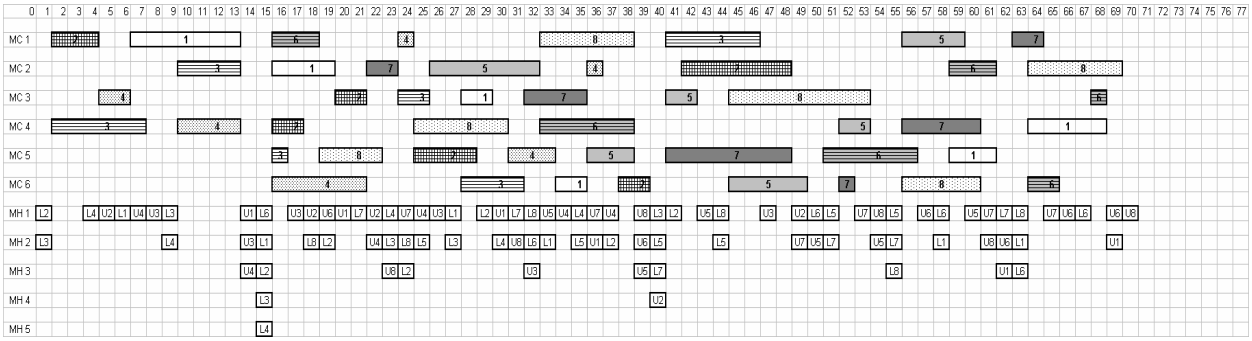

Figure C.11. Lower bound on makespan at time T = 10.

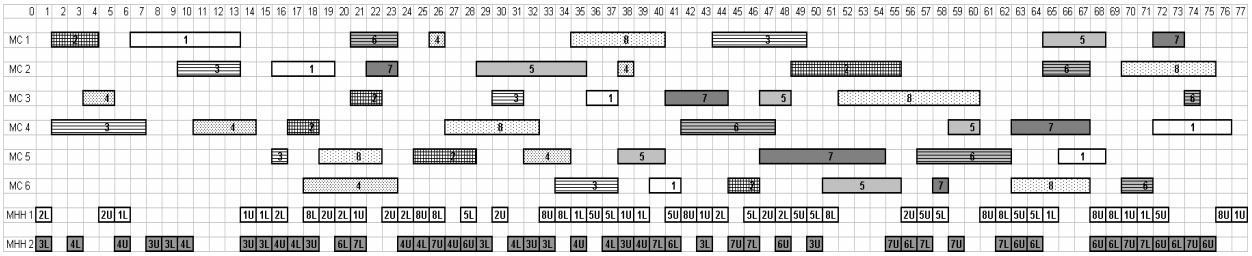

Figure C.12. Intermediate schedule at time T = 10, before Inter-MHH cooperation process.

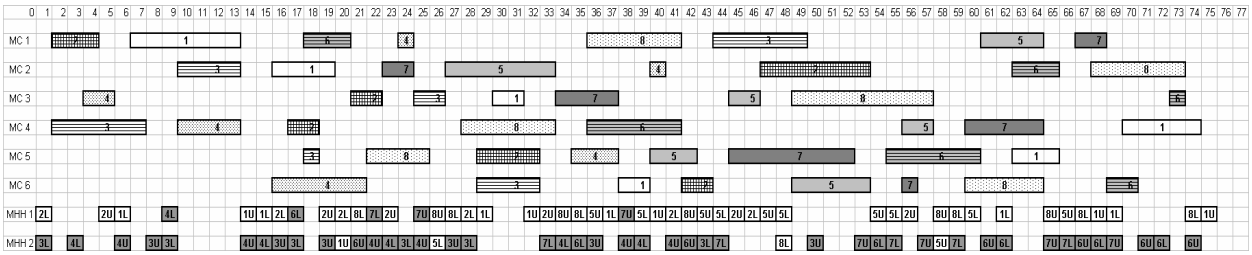

Figure C.13. Final schedule at time T = 10, when considering MH operations.

Appendix D. Schedules Generated when Varying the Number of Available Material Handling Resources in the Holonic System

D.1. Job Information

Table D.1. Characteristics, sequence of operations and processing times for the ten jobs.

Job j	*Rel. Date* r_j	*Weight* w_j	*Due Date* d_j	***Sequence of Operations / Processing Times***																			
				1		*2*		*3*		*4*		*5*		*6*		*7*		*8*		*9*		*10*	
1	0	1	100	8	6	9	2	6	3	4	8	5	5	7	4	3	8	1	9	10	2	2	7
2	0	2	100	6	5	3	10	9	9	10	2	2	9	5	3	8	6	4	8	1	8	7	7
3	0	3	100	2	5	8	5	7	2	3	3	10	4	1	7	5	4	4	3	9	8	6	10
4	0	3	100	5	9	9	8	6	7	10	9	4	4	7	8	3	6	2	4	1	8	8	9
5	0	2	100	9	6	10	2	8	3	4	6	5	8	1	5	2	3	6	10	7	5	3	9
6	0	5	100	4	9	10	7	6	5	3	10	5	7	2	9	7	2	9	6	1	8	8	7
7	0	3	100	6	8	3	5	8	6	7	7	10	8	1	4	5	3	9	5	4	6	2	7
8	0	1	100	5	2	7	9	6	2	9	8	10	6	3	3	1	3	2	8	8	3	4	5
9	0	5	100	10	7	7	7	6	10	9	9	8	7	1	6	4	8	2	10	5	5	3	8
10	0	3	100	1	3	4	5	8	3	3	6	2	8	6	3	9	8	5	4	10	5	7	4

D.2. Job Schedule

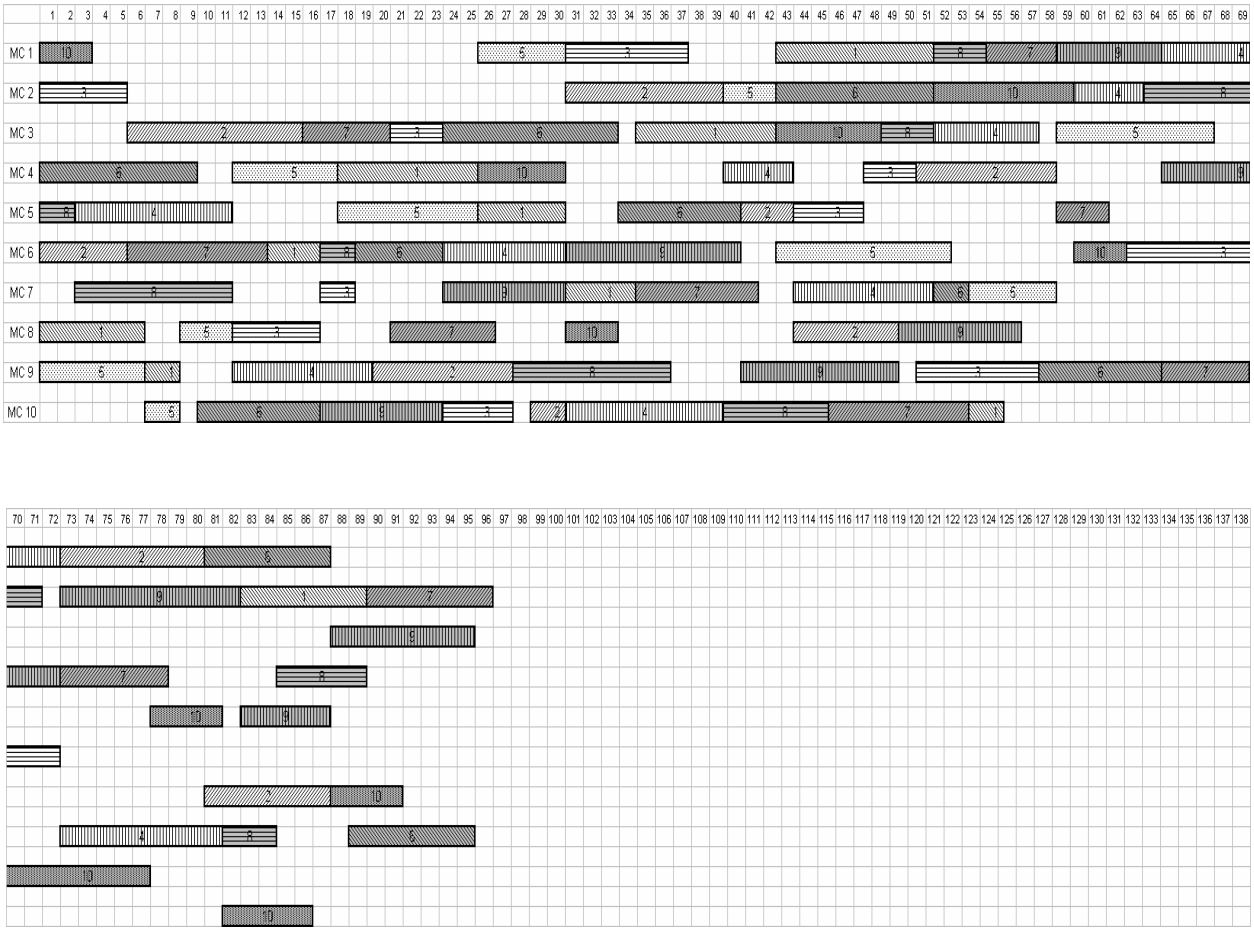

Figure D.1. Job schedule, when MH operations are not considered.

D.3. Lower Bound on Makespan

Figure D.2. Lower Bound on makespan.

D.4. Initial Material Handling Operations Assignment

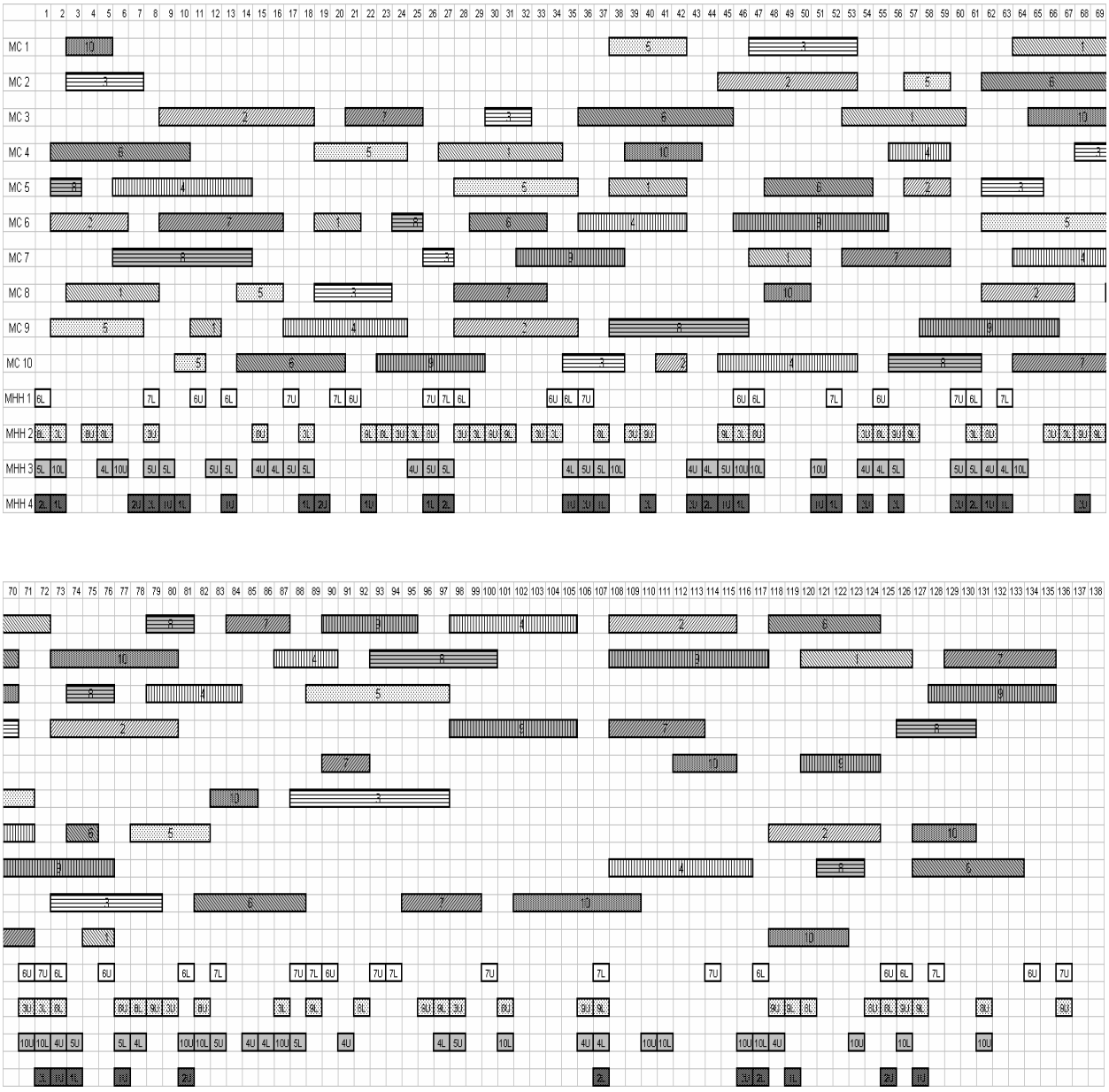

Figure D.3. Schedule generated after considering MH operations.

D.5. Initial System Level Schedule after using Inter-Material Handling Holon Cooperation Mechanisms

Figure D.4. System Level Schedule before the breakdown.

D.6. System Level Schedule Resulted after the Breakdown Occurrence

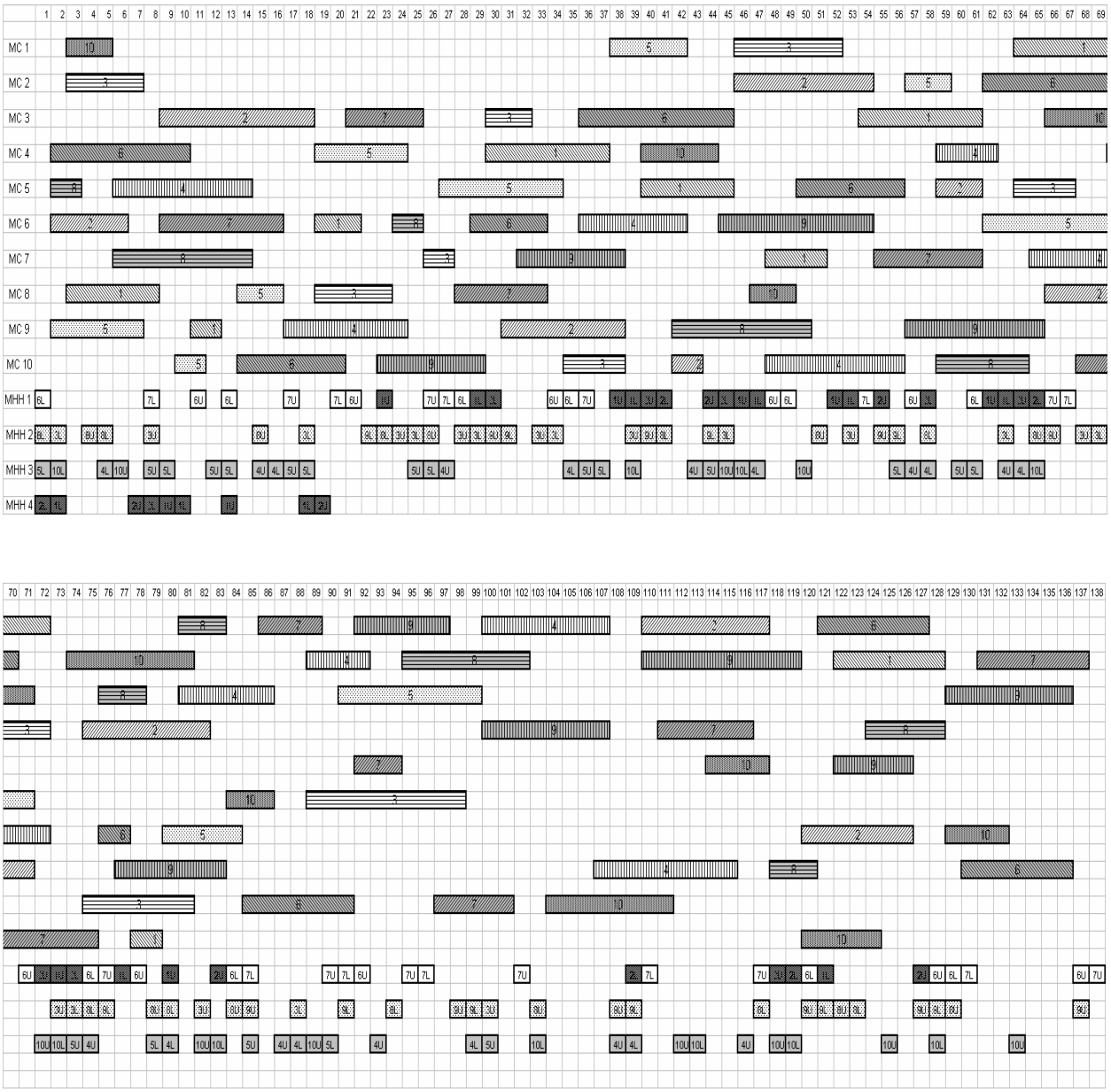

Figure D.5. Assignment of the jobs affected by the breakdown.

D.7. Final System Level Schedule Generated after the Breakdown and the use of Inter-Material Handling Holon Cooperation Mechanisms

Figure D.6. Final System Level Schedule.

Appendix E. Simulation Study Results

E.1. Average Makespan Values for the Processing Operations and LB Information for all Bunches of Replications

Table E.1. Average makespan when MH operations are not considered and average LB values for replication bunch 1.

Replication	C_{max} *w/o MH*	LB_1	LB_2		
	Value	*Value*	*Value*	*CPU Time*	*Violations*
1-1	1009	1409	1990	2.3	51
1-2	1219	1619	2328	3.2	28
1-3	1229	1629	2321	1.8	36
1-4	1465	1865	1917	1.3	39
1-5	1052	1452	2251	3.9	39
1-6	863	1263	2099	4.1	32
1-7	1383	1783	2346	1.2	43
1-8	1025	1425	2159	1.5	36
1-9	1051	1451	1735	1.7	71
1-10	801	1201	2246	2.9	21
Average	1109.7	1509.7	2139.2	2.39	39.6

Table E.2. Average makespan when MH operations are not considered and average LB values for replication bunch 2.

Replication	C_{max} *w/o MH*	LB_1	LB_2		
	Value	*Value*	*Value*	*CPU Time*	*Violations*
2-1	1065	1465	2109	1.9	31
2-2	936	1336	2224	2.9	43
2-3	934	1334	2198	3.8	37
2-4	1137	1537	2329	1.8	22
2-5	951	1351	2285	2.2	29
2-6	1048	1448	2265	2.0	29
2-7	779	1179	2160	2.1	24
2-8	863	1263	2234	3.1	28
2-9	1130	1530	2360	2.4	27
2-10	1150	1550	2139	2.2	28
Average	999.3	1399.3	2230.3	2.44	29.8

Table E.3. Average makespan when MH operations are not considered and average LB values for replication bunch 3.

Replication	C_{max} *w/o MH*	LB_1	LB_2		
	Value	*Value*	*Value*	*CPU Time*	*Violations*
3-1	831	1231	2119	2.5	34
3-2	1009	1409	1880	2.6	64
3-3	1034	1434	2138	2.2	38
3-4	997	1397	2266	2.6	29
3-5	960	1360	2247	2.3	29
3-6	929	1329	2168	1.9	40
3-7	1007	1407	2247	2.2	22
3-8	1086	1486	2059	2.2	43
3-9	1011	1411	2206	3.1	29
3-10	1219	1619	2302	2.2	30
Average	1008.3	1408.3	2163.2	2.38	35.8

Table E.4. Average makespan when MH operations are not considered and average LB values for replication bunch 4.

Replication	C_{max} *w/o MH*	LB_1	LB_2		
	Value	*Value*	*Value*	*CPU Time*	*Violations*
4-1	1016	1416	2331	2.7	28
4-2	1012	1412	2171	2.1	45
4-3	814	1214	2101	2.4	29
4-4	991	1391	2162	2.5	38
4-5	1008	1408	2120	2.3	36
4-6	984	1384	1651	3.6	67
4-7	1271	1671	2333	1.4	30
4-8	893	1293	2229	1.6	35
4-9	826	1226	2264	2.2	21
4-10	1030	1430	2410	1.4	26
Average	984.5	1384.5	2177.2	2.22	35.5

Table E.5. Average makespan when MH operations are not considered and average LB values for replication bunch 5.

Replication	C_{max} *w/o MH*	LB_1	LB_2		
	Value	*Value*	*Value*	*CPU Time*	*Violations*
5-1	1122	1522	2266	1.7	32
5-2	1120	1520	2258	2.2	52
5-3	909	1309	2262	2.1	25
5-4	1069	1469	2153	2.0	29
5-5	775	1175	2253	2.3	24
5-6	1105	1505	1622	2.7	69
5-7	1133	1533	2322	2.3	25
5-8	1002	1402	2268	1.6	22
5-9	949	1349	2134	1.7	28
5-10	1145	1545	1810	2.0	68
Average	1032.9	1432.9	2134.8	2.06	37.4

Table E.6. Average makespan when MH operations are not considered and average LB values for replication bunch 6.

Replication	C_{max} *w/o MH*	LB_1	LB_2		
	Value	*Value*	*Value*	*CPU Time*	*Violations*
6-1	1067	1467	2254	4.6	29
6-2	1036	1436	2015	1.8	53
6-3	1309	1709	2422	1.5	26
6-4	930	1330	2179	2.0	34
6-5	1064	1464	2109	2.0	34
6-6	936	1336	2224	1.7	41
6-7	934	1334	2198	1.5	37
6-8	1137	1537	2329	1.4	22
6-9	951	1351	2285	1.8	28
6-10	1048	1448	2265	1.5	29
Average	1041.2	1441.2	2228.0	1.98	33.3

Table E.7. Average makespan when MH operations are not considered and average LB values for replication bunch 7.

Replication	C_{max} *w/o MH*	LB_1	LB_2		
	Value	*Value*	*Value*	*CPU Time*	*Violations*
7-1	779	1179	2227	1.6	24
7-2	863	1263	2234	1.5	28
7-3	1130	1530	2160	2.1	33
7-4	950	1350	2139	1.8	28
7-5	831	1231	1919	1.9	34
7-6	1009	1409	1880	2.2	64
7-7	1034	1434	2138	1.6	38
7-8	997	1397	2266	2.3	29
7-9	960	1360	2247	1.5	29
7-10	929	1329	2168	3.2	40
Average	948.2	1348.2	2137.8	1.97	34.7

Table E.8. Average makespan when MH operations are not considered and average LB values for replication bunch 8.

Replication	C_{max} *w/o MH*	LB_1	LB_2		
	Value	*Value*	*Value*	*CPU Time*	*Violations*
8-1	1007	1407	2247	2.0	22
8-2	1086	1486	2058	3.1	43
8-3	1011	1411	2206	2.3	29
8-4	1219	1619	2302	1.4	30
8-5	1016	1416	2031	1.9	28
8-6	1012	1412	2171	1.5	45
8-7	1069	1469	1608	2.9	65
8-8	991	1391	1961	1.5	38
8-9	1008	1408	2120	2.2	36
8-10	984	1384	1651	2.4	30
Average	1040.3	1440.3	2035.5	2.12	36.6

Table E.9. Average makespan when MH operations are not considered and average LB values for replication bunch 9.

Replication	C_{max} *w/o MH*	LB_1	LB_2		
	Value	*Value*	*Value*	*CPU Time*	*Violations*
9-1	1271	1671	2333	1.4	30
9-2	893	1293	2229	2.5	35
9-3	826	1226	2264	1.7	21
9-4	1030	1430	2410	1.5	26
9-5	1122	1522	2266	1.7	32
9-6	1120	1520	2258	2.6	52
9-7	909	1309	2262	1.7	25
9-8	1069	1469	2153	2.2	29
9-9	775	1175	2253	1.6	24
9-10	1105	1505	1622	2.6	21
Average	1012.0	1412.0	2205.0	1.95	29.5

Table E.10. Average makespan when MH operations are not considered and average LB values for replication bunch 10.

Replication	C_{max} *w/o MH*	LB_1	LB_2		
	Value	*Value*	*Value*	*CPU Time*	*Violations*
10-1	1133	1533	2322	1.6	25
10-2	1002	1402	2268	2.2	22
10-3	949	1349	2134	2.8	28
10-4	1145	1545	1810	1.6	32
10-5	1067	1467	2254	2.3	41
10-6	1036	1436	2015	1.8	20
10-7	1309	1709	2422	2.0	26
10-8	930	1330	2179	1.7	34
10-9	1340	1740	2365	1.4	41
10-10	1062	1462	2300	2.6	28
Average	1097.3	1497.3	2206.9	2.0	29.7

E.2. Makespan Values and CPU Times Obtained for the Holonic Scheduling and EBFS Algorithm for all Bunches of Replications

Table E.11. Average makespan values and CPU times for the holonic scheduling and EBFS algorithm for replication bunch 1.

Replication	*Holonic Approach*		*EBFS Optimal Algorithm*	
	C_{max}	*CPU Time*	C_{max}	*CPU Time*
1-1	2152	32.0	2131	174.3
1-2	2476	31.0	2466	167.3
1-3	2429	32.4	2418	124.5
1-4	2026	31.8	1991	59.4
1-5	2366	32.1	2349	133.8
1-6	2208	31.6	2175	96.4
1-7	2467	31.9	2445	169.8
1-8	2271	31.3	2244	101.8
1-9	1912	32.2	1882	183.4
1-10	2307	32.5	2287	72.9
Average	2261.4	31.88	2238.8	128.36

Table E.12. Average makespan values and CPU times for the holonic scheduling and EBFS algorithm for replication bunch 2.

Replication	*Holonic Approach*		*EBFS Optimal Algorithm*	
	C_{max}	*CPU Time*	C_{max}	*CPU Time*
2-1	2245	32.8	2163	176.2
2-2	2338	32.6	2331	171.3
2-3	2342	32.3	2299	126.4
2-4	2411	32.1	2393	81.6
2-5	2424	32.6	2401	175.1
2-6	2392	33.9	2379	131.7
2-7	2289	32.6	2272	118.5
2-8	2325	32.7	2311	94.4
2-9	2490	32.5	2449	99.0
2-10	2271	33.2	2249	127.2
Average	2352.7	32.73	2324.7	130.14

Table E.13. Average makespan values and CPU times for the holonic scheduling and EBFS algorithm for replication bunch 3.

Replication	*Holonic Approach*		*EBFS Optimal Algorithm*	
	C_{max}	*CPU Time*	C_{max}	*CPU Time*
3-1	2196	31.8	2183	95.9
3-2	2045	32.4	2045	276.6
3-3	2286	31.9	2272	146.9
3-4	2353	31.4	2332	77.4
3-5	2362	31.8	2324	89.0
3-6	2283	31.6	2267	157.8
3-7	2340	32.2	2314	74.6
3-8	2192	31.9	2171	98.5
3-9	2331	32.0	2302	125.1
3-10	2423	31.4	2381	97.7
Average	2281.1	31.84	2259.1	123.95

Table E.14. Average makespan values and CPU times for the holonic scheduling and EBFS algorithm for replication bunch 4.

Replication	*Holonic Approach*		*EBFS Optimal Algorithm*	
	C_{max}	*CPU Time*	C_{max}	*CPU Time*
4-1	2437	31.0	2409	105.9
4-2	2278	32.1	2278	163.6
4-3	2215	31.9	2189	64.1
4-4	2294	31.7	2279	159.0
4-5	2258	32.1	2230	128.3
4-6	1836	31.9	1831	172.2
4-7	2412	31.5	2398	96.0
4-8	2317	32.0	2293	95.1
4-9	2318	32.0	2316	63.5
4-10	2514	32.1	2470	71.0
Average	2287.9	31.83	2269.3	111.87

Table E.15. Average makespan values and CPU times for the holonic scheduling and EBFS algorithm for replication bunch 5.

Replication	*Holonic Approach*		*EBFS Optimal Algorithm*	
	C_{max}	*CPU Time*	C_{max}	*CPU Time*
5-1	2349	32.1	2348	85.9
5-2	2362	31.8	2344	164.2
5-3	2348	32.9	2337	120.4
5-4	2225	31.6	2209	164.9
5-5	2285	31.9	2280	201.4
5-6	1816	32.5	1806	181.3
5-7	2414	31.6	2392	83.6
5-8	2331	31.6	2324	75.4
5-9	2261	32.2	2253	137.8
5-10	1963	31.9	1953	169.8
Average	2235.4	32.01	2224.6	138.47

Table E.16. Average makespan values and CPU times for the holonic scheduling and EBFS algorithm for replication bunch 6.

Replication	*Holonic Approach*		*EBFS Optimal Algorithm*	
	C_{max}	*CPU Time*	C_{max}	*CPU Time*
6-1	2349	32.4	2333	116.7
6-2	2116	33.3	2105	200.3
6-3	2511	31.5	2494	67.1
6-4	2291	31.8	2265	91.7
6-5	2245	32.9	2243	167.3
6-6	2348	32.0	2331	153.2
6-7	2342	31.8	2299	145.5
6-8	2431	31.8	2413	76.3
6-9	2334	31.3	2321	176.6
6-10	2329	32.8	2329	128.2
Average	2329.6	32.16	2313.3	132.29

Table E.17. Average makespan values and CPU times for the holonic scheduling and EBFS algorithm for replication bunch 7.

Replication	*Holonic Approach*		*EBFS Optimal Algorithm*	
	C_{max}	*CPU Time*	C_{max}	*CPU Time*
7-1	2295	31.2	2272	123.4
7-2	2320	31.9	2311	91.9
7-3	2221	32.3	2199	88.5
7-4	2246	32.1	2229	135.6
7-5	2005	32.0	1983	101.8
7-6	1986	32.0	1954	263.0
7-7	2293	31.6	2272	132.7
7-8	2372	31.7	2332	85.9
7-9	2303	31.7	2294	86.1
7-10	2240	31.7	2237	151.9
Average	2228.1	31.82	2208.3	126.08

Table E.18. Average makespan values and CPU times for the holonic scheduling and EBFS algorithm for replication bunch 8.

Replication	*Holonic Approach*		*EBFS Optimal Algorithm*	
	C_{max}	*CPU Time*	C_{max}	*CPU Time*
8-1	2312	31.8	2304	69.6
8-2	2171	31.6	2171	94.1
8-3	2323	32.6	2302	125.5
8-4	2347	32.0	2341	102.1
8-5	2149	31.5	2109	106.5
8-6	2287	32.0	2267	156.0
8-7	1828	32.2	1809	181.8
8-8	2094	31.8	2079	169.9
8-9	2258	32.1	2230	137.3
8-10	1866	32.1	1851	179.1
Average	2163.5	31.97	2146.3	132.19

Table E.19. Average makespan values and CPU times for the holonic scheduling and EBFS algorithm for replication bunch 9.

Replication	*Holonic Approach*		*EBFS Optimal Algorithm*	
	C_{max}	*CPU Time*	C_{max}	*CPU Time*
9-1	2460	32.9	2438	84.7
9-2	2297	32.1	2293	94.9
9-3	2318	32.0	2306	120.6
9-4	2474	31.6	2470	141.6
9-5	2389	32.0	2348	125.7
9-6	2379	32.2	2334	141.8
9-7	2341	32.8	2337	123.8
9-8	2285	31.7	2209	162.0
9-9	2326	32.7	2280	132.4
9-10	1816	32.1	1806	177.8
Average	2308.5	32.21	2282.1	130.53

Table E.20. Average makespan values and CPU times for the holonic scheduling and EBFS algorithm for replication bunch 10.

Replication	*Holonic Approach*		*EBFS Optimal Algorithm*	
	C_{max}	*CPU Time*	C_{max}	*CPU Time*
10-1	2424	34.0	2392	122.1
10-2	2351	32.1	2324	81.7
10-3	2261	32.0	2253	144.5
10-4	1993	32.6	1953	149.7
10-5	2339	32.3	2333	127.7
10-6	2125	32.6	2105	203.2
10-7	2501	32.6	2494	117.1
10-8	2291	32.3	2265	122.9
10-9	2433	32.1	2433	117.2
10-10	2417	31.7	2396	136.6
Average	2313.5	32.43	2294.8	132.27

E.3. Makespan Values Obtained for the Three Heuristics for all Bunches of Replications

Table E.21. Average makespan values and CPU times for the three heuristics for replication bunch 1.

Bunches of Replications	*JCTH Algorithm*		*MCTH Algorithm*		*MH-RCVH Algorithm*	
	C_{max}	*CPU Time*	C_{max}	*CPU Time*	C_{max}	*CPU Time*
1-1	2216	2.6	2246	2.5	2180	2.9
1-2	2468	2.8	2497	2.5	2476	2.3
1-3	2428	2.3	2435	2.2	2491	2.4
1-4	2029	1.9	2049	1.7	2069	2.5
1-5	2370	2.3	2370	2.5	2372	2.2
1-6	2201	2.8	2217	2.4	2266	1.8
1-7	2454	1.3	2488	2.4	2547	2.2
1-8	2253	1.7	2299	3.1	2326	2.1
1-9	1993	2.5	2006	3.6	2089	2.0
1-10	2305	1.5	2321	2.1	2295	2.7
Average	2271.7	2.17	2292.8	2.5	2311.1	2.31

Table E.22. Average makespan values and CPU times for the three heuristics for replication bunch 2.

Bunches of Replications	*JCTH Algorithm*		*MCTH Algorithm*		*MH-RCVH Algorithm*	
	C_{max}	*CPU Time*	C_{max}	*CPU Time*	C_{max}	*CPU Time*
2-1	2268	1.7	2287	1.9	2327	2.5
2-2	2343	2.0	2343	2.6	2405	2.8
2-3	2425	2.0	2349	3.9	2342	3.1
2-4	2413	1.6	2416	2.3	2444	1.5
2-5	2433	1.7	2426	2.3	2457	2.2
2-6	2407	1.5	2396	3.2	2417	2.3
2-7	2292	1.8	2296	2.5	2294	2.5
2-8	2312	2.9	2655	3.8	2332	2.8
2-9	2475	1.8	2484	2.3	2492	2.0
2-10	2272	1.6	2274	2.2	2335	2.4
Average	2364	1.86	2392.6	2.7	2384.5	2.41

Table E.23. Average makespan values and CPU times for the three heuristics for replication bunch 3.

Bunches of Replications	*JCTH Algorithm*		*MCTH Algorithm*		*MH-RCVH Algorithm*	
	C_{max}	*CPU Time*	C_{max}	*CPU Time*	C_{max}	*CPU Time*
3-1	2201	1.8	2248	1.9	2234	2.5
3-2	2116	1.4	2146	2.3	2173	1.7
3-3	2321	1.7	2327	2.0	2303	1.8
3-4	2333	2.1	2397	2.1	2368	1.8
3-5	2328	2.0	2369	2.7	2342	1.9
3-6	2284	1.8	2363	3.4	2297	2.2
3-7	2349	1.7	2323	2.0	2347	1.8
3-8	2205	1.5	2313	3.3	2191	1.9
3-9	2362	1.6	2372	2.1	2343	2.4
3-10	2425	1.4	2415	1.8	2437	1.7
Average	2292.4	1.7	2327.3	2.36	2303.5	1.97

Table E.24. Average makespan values and CPU times for the three heuristics for replication bunch 4.

Bunches of Replications	*JCTH Algorithm*		*MCTH Algorithm*		*MH-RCVH Algorithm*	
	C_{max}	*CPU Time*	C_{max}	*CPU Time*	C_{max}	*CPU Time*
4-1	2437	1.9	2447	2.2	2440	1.6
4-2	2282	1.7	2310	3.0	2347	1.7
4-3	2232	1.9	2215	2.5	2233	1.9
4-4	2297	1.7	2323	2.2	2302	2.0
4-5	2284	2.1	2337	2.0	2332	1.8
4-6	1858	2.2	2017	3.1	1977	2.0
4-7	2427	1.8	2414	2.3	2482	1.6
4-8	2322	1.9	2334	2.2	2327	1.9
4-9	2316	1.9	2372	3.6	2368	2.2
4-10	2459	1.9	2484	2.0	2525	1.5
Average	2291.4	1.90	2325.3	2.51	2333.3	1.82

Table E.25. Average makespan values and CPU times for the three heuristics for replication bunch 5.

Bunches of Replications	*JCTH Algorithm*		*MCTH Algorithm*		*MH-RCVH Algorithm*	
	C_{max}	*CPU Time*	C_{max}	*CPU Time*	C_{max}	*CPU Time*
5-1	2369	1.7	2357	2.6	2400	2.5
5-2	2373	1.7	2393	2.2	2387	1.6
5-3	2349	1.8	2377	2.9	2358	1.9
5-4	2265	1.6	2276	2.6	2290	2.2
5-5	2326	2.1	2338	2.5	2330	2.0
5-6	1889	2.0	1914	1.9	1927	2.0
5-7	2426	1.9	2445	2.1	2446	1.9
5-8	2368	1.9	2371	2.3	2374	1.8
5-9	2273	2.4	2291	2.3	2203	2.0
5-10	1972	1.8	1998	2.3	2035	1.6
Average	2261.0	1.89	2276.0	2.37	2275.0	1.95

Table E.26. Average makespan values and CPU times for the three heuristics for replication bunch 6.

Bunches of Replications	*JCTH Algorithm*		*MCTH Algorithm*		*MH-RCVH Algorithm*	
	C_{max}	*CPU Time*	C_{max}	*CPU Time*	C_{max}	*CPU Time*
6-1	2340	2.1	2416	1.6	2382	2.4
6-2	2143	1.8	2159	1.7	2179	2.2
6-3	2516	1.8	2527	2.1	2502	2.1
6-4	2297	1.9	2313	2.0	2277	2.0
6-5	2268	1.7	2287	1.8	2327	2.4
6-6	2343	1.8	2343	1..9	2405	2.7
6-7	2425	2.2	2349	1.7	2342	2.6
6-8	2433	1.6	2416	1.6	2444	1.8
6-9	2373	1.7	2396	1.9	2357	1.9
6-10	2357	2.4	2396	2.5	2397	1.7
Average	2349.5	1.90	2360.2	1.88	2361.2	2.18

Table E.27. Average makespan values and CPU times for the three heuristics for replication bunch 7.

Bunches of Replications	*JCTH Algorithm*		*MCTH Algorithm*		*MH-RCVH Algorithm*	
	C_{max}	*CPU Time*	C_{max}	*CPU Time*	C_{max}	*CPU Time*
7-1	2272	1.7	2296	2.5	2294	2.1
7-2	2329	2.3	2355	2.1	2332	2.8
7-3	2235	3.4	2284	1.8	2292	1.9
7-4	2262	2.2	2274	2.0	2295	1.9
7-5	2001	2.2	2048	2.2	2034	2.2
7-6	2016	1.7	2046	1.8	2073	2.0
7-7	2321	1.5	2327	2.0	2302	1.8
7-8	2333	1.7	2397	2.0	2368	1.9
7-9	2305	1.8	2369	2.4	2342	1.6
7-10	2284	2.0	2263	2.1	2297	1.9
Average	2235.8	2.05	2265.9	2.09	2262.9	2.01

Table E.28. Average makespan values and CPU times for the three heuristics for replication bunch 8.

Bunches of Replications	*JCTH Algorithm*		*MCTH Algorithm*		*MH-RCVH Algorithm*	
	C_{max}	*CPU Time*	C_{max}	*CPU Time*	C_{max}	*CPU Time*
8-1	2309	2.3	2323	2.3	2347	1.7
8-2	2205	3.6	2213	2.5	2191	1.8
8-3	2342	1.9	2372	1.7	2343	2.0
8-4	2364	2.4	2399	1.8	2397	1.7
8-5	2197	1.6	2207	1.7	2220	1.7
8-6	2282	1.5	2310	2.1	2347	1.8
8-7	1868	1.5	1890	1.9	1899	1.9
8-8	2097	2.1	2123	2.3	2102	1.8
8-9	2284	1.8	2337	2.0	2332	1.7
8-10	1858	2.0	1982	2.1	1977	2.1
Average	2180.6	2.07	2215.6	2.04	2215.5	1.82

Table E.29. Average makespan values and CPU times for the three heuristics for replication bunch 9.

Bunches of Replications	*JCTH Algorithm*		*MCTH Algorithm*		*MH-RCVH Algorithm*	
	C_{max}	*CPU Time*	C_{max}	*CPU Time*	C_{max}	*CPU Time*
9-1	2467	1.6	2454	2.0	2482	1.6
9-2	2311	1.8	2334	2.3	2327	1.8
9-3	2306	1.8	2372	2.3	2368	2.0
9-4	2489	1.6	2484	2.1	2525	1.7
9-5	2398	1.6	2357	1.9	2400	1.9
9-6	2373	1.4	2392	1.8	2387	1.6
9-7	2349	1.8	2377	1.8	2358	2.0
9-8	2265	1.6	2276	1.8	2280	1.7
9-9	2326	1.8	2338	2.4	2330	2.0
9-10	1889	2.1	1914	1.9	1927	1.9
Average	2317.3	1.71	2329.8	2.03	2338.4	1.82

Table E.30. Average makespan values and CPU times for the three heuristics for replication bunch 10.

Bunches of Replications	*JCTH Algorithm*		*MCTH Algorithm*		*MH-RCVH Algorithm*	
	C_{max}	*CPU Time*	C_{max}	*CPU Time*	C_{max}	*CPU Time*
10-1	2426	1.8	2445	2.1	2416	1.6
10-2	2368	1.8	2344	1.6	2374	2.5
10-3	2273	2.0	2291	1.7	2273	2.2
10-4	2042	2.2	2038	2.9	2035	1.7
10-5	2340	1.7	2367	1.7	2347	1.8
10-6	2143	1.9	2159	3.4	2179	2.0
10-7	2516	1.6	2527	1.5	2505	1.4
10-8	2297	2.0	2283	1.7	2277	1.9
10-9	2480	1.8	2442	1.9	2469	2.4
10-10	2405	1.7	2405	2.0	2415	2.1
Average	2329.0	1.85	2330.1	2.05	2329.0	1.96

E.4. Makespan Values Obtained for the Holonic Scheduling when Resource Breakdowns are Considered for All Bunches of Replications

Table E.31. Average makespan values and CPU times for the holonic scheduling approach when resource breakdowns are considered for replication bunch 1.

Bunches of Replications	*Holonic Scheduling w/ Resource Breakdowns*	
	C_{max}	*CPU Time*
1-1	2183	44.1
1-2	2492	43.1
1-3	2501	45.1
1-4	2145	44.4
1-5	2399	44.6
1-6	2264	43.5
1-7	2516	44.2
1-8	2382	43.1
1-9	1937	44.5
1-10	2348	45.0
Average	2316.7	44.16

Table E.32. Average makespan values and CPU times for the holonic scheduling approach when resource breakdowns are considered for replication bunch 2.

Bunches of Replications	*Holonic Scheduling w/ Resource Breakdowns*	
	C_{max}	*CPU Time*
2-1	2269	44.8
2-2	2402	44.4
2-3	2384	44.0
2-4	2456	43.2
2-5	2463	44.3
2-6	2512	46.2
2-7	2332	44.3
2-8	2364	44.4
2-9	2503	44.4
2-10	2372	45.5
Average	2405.7	44.55

Table E.33. Average makespan values and CPU times for the holonic scheduling approach when resource breakdowns are considered for replication bunch 3.

Bunches of Replications	*Holonic Scheduling w/ Resource Breakdowns*	
	C_{max}	*CPU Time*
3-1	2259	43.7
3-2	2280	44.9
3-3	2292	43.9
3-4	2395	43.2
3-5	2385	43.6
3-6	2410	43.3
3-7	2352	44.4
3-8	2198	44.0
3-9	2446	44.0
3-10	2457	43.0
Average	2347.4	43.80

Table E.34. Average makespan values and CPU times for the holonic scheduling approach when resource breakdowns are considered for replication bunch 4.

Bunches of Replications	*Holonic Scheduling w/ Resource Breakdowns*	
	C_{max}	*CPU Time*
4-1	2439	42.4
4-2	2311	44.0
4-3	2220	43.8
4-4	2306	44.0
4-5	2379	44.1
4-6	2052	43.8
4-7	2432	43.0
4-8	2348	44.2
4-9	2384	44.3
4-10	2625	44.4
Average	2349.6	43.80

Table E.35. Average makespan values and CPU times for the holonic scheduling approach when resource breakdowns are considered for replication bunch 5.

Bunches of Replications	*Holonic Scheduling w/ Resource Breakdowns*	
	C_{max}	*CPU Time*
5-1	2396	44.2
5-2	2386	44.1
5-3	2392	44.9
5-4	2294	43.7
5-5	2365	44.0
5-6	1919	45.3
5-7	2434	43.2
5-8	2385	43.3
5-9	2301	44.5
5-10	1998	44.2
Average	2287.0	44.14

Table E.36. Average makespan values and CPU times for the holonic scheduling approach when resource breakdowns are considered for replication bunch 6.

Bunches of Replications	*Holonic Scheduling w/ Resource Breakdowns*	
	C_{max}	*CPU Time*
6-1	2396	44.7
6-2	2178	45.5
6-3	2579	43.0
6-4	2312	44.1
6-5	2364	45.3
6-6	2364	43.6
6-7	2349	43.6
6-8	2454	44.0
6-9	2435	42.8
6-10	2337	45.1
Average	2376.8	44.17

Table E.37. Average makespan values and CPU times for the holonic scheduling approach when resource breakdowns are considered for replication bunch 7.

Bunches of Replications	*Holonic Scheduling w/ Resource Breakdowns*	
	C_{max}	*CPU Time*
7-1	2366	42.6
7-2	2399	44.1
7-3	2280	44.7
7-4	2255	44.2
7-5	2120	43.9
7-6	2088	43.7
7-7	2306	43.2
7-8	2385	43.2
7-9	2323	43.5
7-10	2304	43.6
Average	2282.6	43.67

Table E.38. Average makespan values and CPU times for the holonic scheduling approach when resource breakdowns are considered for replication bunch 8.

Bunches of Replications	*Holonic Scheduling w/ Resource Breakdowns*	
	C_{max}	*CPU Time*
8-1	2322	44.0
8-2	2251	43.8
8-3	2394	44.6
8-4	2391	43.7
8-5	2207	43.0
8-6	2329	43.9
8-7	1932	44.3
8-8	2196	43.4
8-9	2311	44.9
8-10	1898	44.2
Average	2223.1	43.98

Table E.39. Average makespan values and CPU times for the holonic scheduling approach when resource breakdowns are considered for replication bunch 9.

Bunches of Replications	*Holonic Scheduling w/ Resource Breakdowns*	
	C_{max}	*CPU Time*
9-1	2521	45.4
9-2	2372	44.1
9-3	2385	44.4
9-4	2519	43.4
9-5	2399	45.0
9-6	2384	44.2
9-7	2389	44.7
9-8	2296	43.7
9-9	2361	45.2
9-10	1881	44.2
Average	2350.7	44.43

Table E.40. Average makespan values and CPU times for the holonic scheduling approach when resource breakdowns are considered for replication bunch 10.

Bunches of Replications	*Holonic Scheduling w/ Resource Breakdowns*	
	C_{max}	*CPU Time*
10-1	2448	46.3
10-2	2410	44.6
10-3	2275	43.7
10-4	2038	44.4
10-5	2396	44.4
10-6	2165	44.7
10-7	2571	45.0
10-8	2355	44.8
10-9	2482	44.3
10-10	2424	43.8
Average	2356.4	44.6

Vita

Radu F. Babiceanu started his Ph.D. studies in the Grado Department of Industrial and Systems Engineering at Virginia Tech, Manufacturing Systems Engineering Option in August 2001. At the same time he joined the Center for High Performance Manufacturing, where he worked as a Graduate Research Assistant under the direction of Dr. F. Frank Chen. His main research interests include the agent and holonic-based approaches to controlling manufacturing systems and the applications of artificial intelligence techniques in manufacturing and service sectors. Prior to coming to Virginia Tech, he received a B.S. degree in mechanical engineering from the Polytechnic University of Bucharest, Romania, and a M.S. degree in mechanical engineering from the University of Toledo, Ohio. In between these two degrees, he worked as a manufacturing and design engineer in the Romanian industry, having responsibilities in both shop floor production and engineering design areas.