Modeling, Scheduling, and Performance Evaluation for Deadlock-free Flexible Manufacturing Cells for a Dual Gripper Robot: A Constraint Programming Approach

NABIL EL KHAIRI

A thesis submitted to the Faculty of Graduate Studies of
The University of Manitoba
in partial fulfillment of the requirements of the degree of

MASTER OF SCIENCE

Department of Mechanical and Manufacturing Engineering
University of Manitoba
Winnipeg, Manitoba, Canada

Abstract

Deadlocks are critical events in Flexible Manufacturing Cells (FMC) that result from circular waits among a set of resources. Circular waits happen when a set of resources with finite capacity are in a permanent hold due to wait state to admit new jobs. Past literature examines the deadlock-free scheduling in FMCs considering many types of resources and techniques. This thesis proposes a new resource-oriented deadlock-free approach using a robot equipped with dual-grippers serving as a material handler in a FMC. The proposed methodology uses Constraint Programming (CP). The system performance is analyzed using different buffer configurations. Many test problems are generated to validate the developed models. The finding demonstrates that the proposed dual-gripper robot (DGR) can outperform the single-gripper robot (SGR) in many settings for FMCs. Likewise, the experience with the CP for the modeling and solving approach proposed in this research consolidates its application to FMC deadlock-free scheduling problems.

Acknowledgement

I would like to sincerely thank my advisors Dr. Tarek ElMekkawy and Dr. Subramaniam Balakrishnan for guiding my research work presented in this thesis. I have learnt a lot from them as experts and professionals. Special thanks to Jenifer Mohammed for providing valuable help in editing my thesis. I would like to express my gratitude to my laboratory mate Al Mehdi Ibrahem for his encouragement throughout my times in the Advanced Research Manufacturing Facility. Finally, I would like to acknowledge the great support that I have received from my parents, sister, friends, and family members.

Contents

List of Tables

List of Figures

Chapter 1 INTRODUCTION

1.1. Introduction to the Problem

The manufacturing environment is becoming more complex due to the competitive demand of the market place. The success of manufacturing systems depends on more efficient use of production systems. Advanced automated systems are nowadays present in most manufacturing industries. The advantages gained through automation are seen in today's manufacturing practice in terms of productivity improvement. The application of technological advancements such as buffers and robots has great impact on automated production systems.

Flexible manufacturing cells (FMCs) are modern automated manufacturing systems that use recent technologies to produce parts requiring specific manufacturing processes. FMCs benefit from the application of group technology philosophy. Several aspects of FMCs are well defined in the literature. An FMC consists usually of multiple computer numerical control (CNC) machines grouped in a manufacturing cell. The grouping of the CNCs is usually done at the process planning stage. Different types of CNCs are available in the market for manufacturers. Each CNC is characterised by certain machining abilities to perform certain manufacturing processes. Eventually, CNCs have a positive impact on the FMCs' productivity. In real life, FMCs are very widely used in the automotive industry.

In FMCs, disturbances lead to a reduction in the efficiency and effectiveness of the performance. Disturbance costs are high and account for production losses due to the inability of the manufacturing cell to finish the processing of jobs as per the manufacturing schedule. Deadlocks are an example of highly undesirable disturbance situations that must be addressed in FMCs. Deadlock situations are circular waits among the existing resources in the manufacturing cell. In case of a deadlock, jobs residing in the shared resources will never leave these resources. In turn, the system will be in a halt state.

Figure 1 depicts the scenario of a deadlock situation in an FMC. The manufacturing cell shown is composed of three CNC machines (CNC1, CNC2, and CNC3) characterized by different parameters and machining operation. The CNC machines are processing non-identical parts. The number of part types is five (P1...P5). The parts are represented in the figure as circles. The variability among routing of parts is significant and could create deadlocks if this type of situation is not properly managed. The processing times and routing for each part type is determined in the process plan. P1 and P2 require processing on CNC1, CNC2, and CNC3 in that order. P3 and P4 require processing on CNC2, CNC3, and CNC1 in that order. P5 requires processing on CNC3, CNC1, and CNC2 in that order. P2 is in the input/output device (I/O) waiting to enter the manufacturing cell. P4 is in the input/output device (I/O) and has already finished processing in the manufacturing cell. CNC1 has finished processing P1. The next processing of P1 must be done on CNC2. CNC2 has finished processing P3. The next processing of P3 must be done on CNC3. CNC3 has finished processing P5. The next processing of P5 must be done on CNC1. CNC1 cannot release P1 because CNC2 is held by P3. CNC2 cannot release P3 because CNC3 is held by P5. CNC3 cannot release P5 because CNC1 is held by P1. Consequently, a circular wait occurs between CNC1, CNC2, and CNC3.

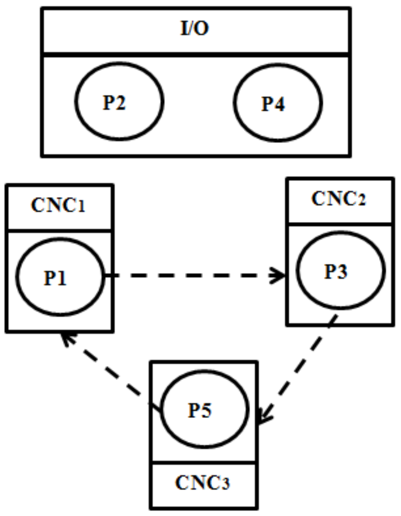


Figure 1. Case of Deadlock in an Automated Facility

From this example, it is clear that the functions of the given FMC are affected by the deadlock occurrence. Deadlocks result in very low productivity. Moreover, they are difficult to manage and cannot be solved easily. In large scale operation, FMCs have a limited capacity of resources and are most likely to have non-identical part production. Therefore, deadlocks are almost always present in such cases. That being said, deadlock-freeness is highly needed.

1.2. Problem Statement

The work presented in this thesis is a special case of a FMC defined in the literature as flexible job shop scheduling problem (F-JSSP). The F-JSSP is a class of optimization problem. The problem for F-JSSP is a two decision-making problem, which is as follows: allocation of operation to the machines and generation of sequences to route parts through the machines. The job shop scheduling problem (JSSP) which is a special case of F-JSSP is also studied in this thesis. For JSSP, the problem is only to determine the sequence of operations on the machines. The objective is to find optimal (or near optimal) deadlock-free schedules. The configuration of the manufacturing cell in this thesis will permit rearrangement of the different resources (machine-type, buffer-type, and gripper-type). A framework for the proposed scenario is defined in the classification scheme for a FMC in Figure 2. This framework originated from the traditional classification scheme for robotic cells presented in Dawande (2007). Level 1 represents the machine environment including existence of buffers, types of buffers, and types of robot grippers. In addition, Level 1 represents the routing criterion that includes the existence of routing flexibility of parts, or no routing flexibility of the parts where only unique route of the jobs are assumed. Further, Level 2 represents the pickup criterion and part-types. The pickup criterion represented in the scheme is a free-pickup. Such criterion means that jobs do not have to be moved immediately after an operation is completed. In other words, intermediate delay is permissible between two operations of a job. Moreover, the part-type is multiple which assumes that the manufacturing cell processes non-identical parts (mixed parts). Finally, the Level 3 represents the targeted performance measure. The performance measure considered in this thesis is the mean flow time (MFT). The MFT is the sum of the completion time of jobs divided by the number of jobs. The conditions and assumptions considered in this thesis are defined as follows:

- The machines process one job at any given time,
- Each robot gripper can handle one job at any given time,
- Capacity of the buffers cannot be exceeded,
- The processing of a job on a machine cannot be pre-empted,
- A job should not leave a resource until the next resource needed is available,
- The transportation times are not considered and will vary based on the task handled.

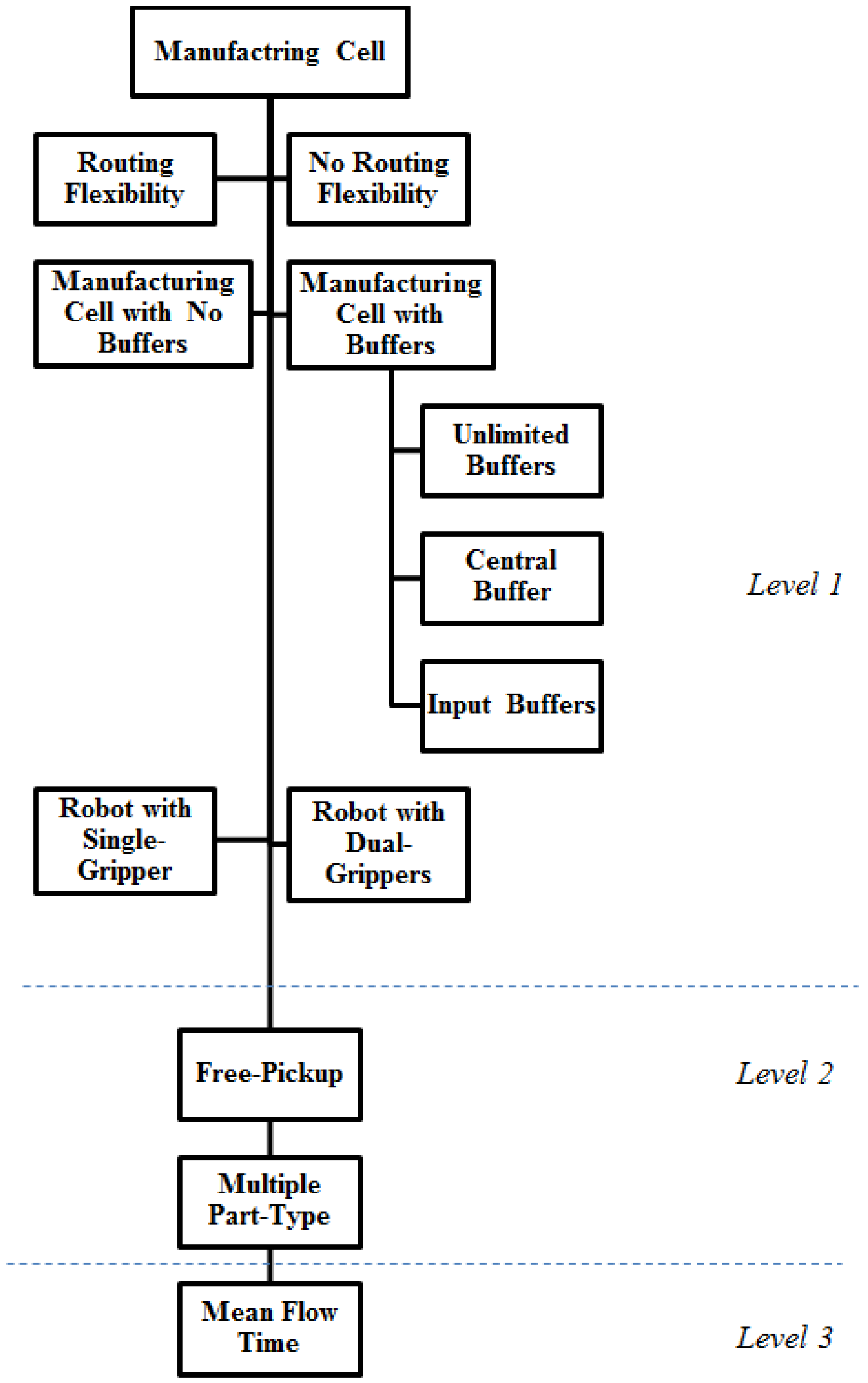


Figure 2. A Classification of the Studied Manufacturing Cells

1.3. Research Objectives

Deadlock-freeness, cell design, and scheduling for optimal production are the scope of many studies that have emphasis on FMCs' applications. The objective of this thesis is to investigate the deadlock resolution problem in FMCs with different configurations. These configurations include manufacturing cell with unlimited buffer capacity; with no buffer and single-gripper robot; with central buffer and single-gripper robot; with input buffers and single-gripper robot; with no buffer and dual-gripper robot; with central buffer and dual-gripper robot; and with input buffers and dual-gripper robot. In this regard, this thesis aims at producing a systematic analysis of the deadlock resolution problem in the following cases.

1. Developing constraint programming models for the deadlock-free scheduling problem in the manufacturing cells with the different configurations using IBM ILOG CP Optimizer,
2. Solving the developed models with small size test problems and validating the results through Gantt charts,
3. Generating large test problems for the different manufacturing cell configurations to study the effect of the following:
    a) The various resources in a manufacturing cell with an emphasis on dual-gripper robots, and
    b) The routing flexibility on a manufacturing cell when jobs have alternative routes.

1.4. Thesis Structure

Literature review is presented in Chapter 2 to discuss the previous work related to deadlock-free scheduling approaches, dual-gripper robots, and constraint programming methodology. Chapter 3 outlines the deadlock-free scheduling approaches. The complete models of the proposed configurations are also presented. A validation of the developed models is presented in Chapter 4, based on a small size test problem. The test outcomes are then interpreted based on numerical results and Gantt charts for each configuration. In Chapter 5, an extensive performance evaluation is established. First, a comparison of system performance between robots with single and dual grippers for different problems, as presented in the literature, is established. Second, different groups of test problems that represent situations in an industrial environment are solved. A numerical evaluation of routing flexibility and configuration design are provided. Finally, Chapter 6 presents the conclusions arising from this work and suggestions for future research.

Chapter 2 SURVEY OF LITERATURE

2.1. Introduction

Deadlock occurrence has its origin in the field of computer science. Some of the earliest studies involving deadlock occurrence in manufacturing systems were done by Wysk et al. (1991). Many later studies examined different deadlock strategies and the complexity of the scheduling problem especially with the emerging advanced technologies in FMCs in the last two decades. Traditional scheduling approaches were not responding well to the need of the FMC when one has to deal with deadlocks. More recently, a number of researchers began studying and presenting their results on deadlock-free scheduling in a way that is aligned with the rising complexity of FMCs. These deadlock-free solutions are now being applied in many FMC configurations and industries. A common agreement between researchers is that machines, buffers, and material handlers all contribute to the occurrence of deadlocks. These resources are highly associated. Of course, not all possible configurations of cells and resource types have been studied in the literature and hence deadlocks present a rich area of research.

Interested groups in manufacturing recognize that scheduling for a deadlock-free production in FMCs is a computational NP-hard problem where the computational complexity increases with the size of the problem, (Ponnambalam et al. (2009)). Several optimization methods were used to solve the problem. The results indicate that its level of complexity depends on the methods used.

The intention of the literature review presented in this section is to show the scholarly basis for this research. This section examines the most relevant literature in the field of deadlock-free resolution in FMCs. This overview focuses more on literature that uses new approaches to solve this problem. The dual-gripper robot approach will be used for the first time to resolve deadlock-free scheduling. However, dual-gripper robots have been used in robotic cells for the material handling task in many industrial applications. Many examples could be found in the semiconductor fabrication industry. Therefore, this thesis will discuss relevant literature related to scheduling problems in manufacturing systems with an aim to attain greater productivity. Constraint programming methodology, which is gaining more popularity in manufacturing related problems in academia, has never been used before for solving the deadlock-free scheduling problem. This section will examine the contribution of this method in manufacturing problems especially with scheduling and modeling as the focus. Conclusions arising from the review are given at the end of this section.

2.2. Scheduling with Deadlock Resolution

The deadlock-handling policies represent the strategies that deal with the deadlocks. There are three policies for deadlock resolution in FMCs: deadlock detection and recovery, deadlock avoidance, and deadlock prevention. Zhiwu et al. (2009) discusses the deadlock control policies as follows: A deadlock detection and recovery approach permits the occurrence of deadlocks. When a deadlock occurs, it is detected and then the system is put back to a deadlock-free state, by simply reallocating the resources. In deadlock avoidance, at each system state, an on-line control policy is used to make a correct decision to proceed among the feasible evolutions. The main purpose of this approach is to keep the system free of deadlock states. Deadlock prevention is considered to be a well-defined problem in deadlock resolution scheduling literature. It is usually achieved by using an off-line computational mechanism to control the request for resources to ensure that deadlocks never occur. The goal of a deadlock prevention approach is to impose constraints on a system to prevent it from reaching deadlock states. In this case, the computation is carried out offline in a static way and once the control policy is established, the system can no longer reach undesirable deadlock states. The efficiency of each policy is different. Many factors can be taken into consideration when comparing these three policies such as computation algorithms, cell productivity, implementation, resource utilization, and costs. In the work of Zhiwu et al. (2009) and Fanti et al. (2004) the various aspects of deadlock-free policies are presented.

Ramaswamy et al. (1996) reported that the deadlock-handling problem differs considerably from the traditional scheduling problem. The deadlock-free problem in FMCs has another component beside the control policies. This important component is the scheduling aspects. The scheduling function is to optimize tasks in the manufacturing cell to find best objective values. Zhiwu et al. (2012) show that many studies neglect the optimization function and focus more on the control strategies. However, these types of studies do not satisfy the manufacturing objectives, which have created a gap between deadlock-handling policies and implementation in real industry as mentioned by Yoon (2010) and Xin et al (2012). There are many studies, listed in Table 1, which do consider deadlock handling policies and optimising strategies for the scheduling problem. These studies focus on finding optimal or near optimal deadlock-free schedules for modeled FMCs. Many operation research techniques are used including exact methods, heuristic algorithms, and the meta-heuristic algorithms. They include branch-and-bound, genetic algorithm, mathematical modelling, Petri Nets, Automata, insertion algorithms with rank matrices, Lagrangian relaxation, hybrid algorithms, Taboo Search, and Beam Search.

A significant number of optimization approaches deal with deadlocks-free scheduling generally combined with graphical and mathematical tools such as Petri Nets. The optimization approaches are used to find the deadlock-free schedules by employing Petri Nets for representing the structure of the manufacturing cell and the existing resources. For example, Yoon (2010) developed deadlock-free genetic scheduling algorithms for automated manufacturing systems based on deadlock avoidance policy. The manufacturing system and the flexible routes for the jobs are modeled with Petri Nets. Deadlock-freeness is ensured using genetic scheduling procedures. The design of genetic algorithms dealing with the scheduling problems is based on a chromosome generating mechanism. The feasible deadlock-free schedules are selected from feasible chromosomes exclusively. The search is guided by the best objective value. The authors conclude that the proposed theoretical method is applicable to all automated manufacturing systems.Zhiwu et al. (2012), Zhiwu et al. (2009) and Murata (1989) recognized Petri Nets as one of the most powerful formal methods for modeling FMCs. However, Deering (2012) mentioned that strategies using Petri Nets do receive some criticism in their limitation to appropriately represent manufacturing systems where there is more variance in the type of resources. The problem sizes are also found to be very large. That is why when Petri Nets are used, priority rules are added to the solving algorithm to solve larger instances, Zhang et al. (2009).

There are still some studies that do not use any Petri Nets such as the work done by Fahmy et al. (2009), Fahmy et al. (2008), Fahmy et al. (2010), Fahmy et al. (2007), Ramaswamy et al. (1996), and Sabuncuoglu et al. (2003). Fahmy et al. (2009) propose a generic deadlock-free reactive scheduling method based on a job insertion algorithm using rank matrices for flexible job shops. Operations of new jobs are inserted in the manufacturing operation. Corresponding rank matrix representing the schedule are verified to detect any circular waits as and when they form. When all jobs are inserted and different combinations are generated, the combination that gives the best performance measure in the schedule is chosen. The algorithm is developed to ensure the deadlock-freeness of the production system. It has the capability to react to disturbing situations such as machine breakdowns, process time variation, urgent jobs, and cancellation of orders while gaining the advantage of the flexible routing of the jobs and buffer capacities. Since that study deals with deadlock-free reactive scheduling, the authors demonstrate the efficiency and stability of the schedules. Fahmy et al. (2008) propose deadlock-free scheduling of flexible job shops with limited capacity buffers. Mixed integer programming is used to find optimal solutions. The deadlock prevention policy for minimising the MFT (or makespan (MS)) are both mathematically modeled. The work demonstrates the effective utilisation of the buffer's capacities for preventing the deadlock even though the mathematical models are difficult to solve. This study also proposes a heuristic approach based on the insertion algorithm and rank matrices. It concludes that the proposed heuristic can solve a larger size problem in a timely and efficient manner. The authors report that this method can be used for other cases beside deadlock-freeness, buffer's existence, material handling, and job flexible routing. Fahmy et al. (2010) propose mathematical formulations for scheduling in manufacturing cells with limited capacity buffers using mixed integer programming. Optimal deadlock-free schedules for minimizing the MFT are found. The authors establish a performance evaluation for the different models proposed by comparing the solutions. They conclude that the solution quality and solution times are the trade-off factors in the designed manufacturing cell. Sabuncuoglu et al. (2003) propose a deadlock-free reactive scheduling methodology in a dynamic and stochastic FMC based on filtered beam search technique. The proposed scheduling solution assumes multiple machines, material handlers, buffer capacities, and flexible routing. The authors demonstrate the performance of the system by considering processing time variations and machine breakdowns to draw conclusions about the robustness and efficiency of the scheduling solutions. The work of Ramaswamy et al. (1996) deals with generating deadlock-free schedules for automated manufacturing workstations built in a job shop context. They examine various types of resources and cell configurations using mixed integer programming and Lagrangian relaxation heuristics. Restrictions are imposed on the resources to prevent circular waits from happening. It demonstrates that the complex manufacturing cell structure and complex resource interactions makes the system deadlock-prone. Therefore, Mati et al. (2001) report that any discrepancy between the current level of complexity in the manufacturing cell and poor control policies may lead to undesirable deadlocks.

Table 1. Literature Survey of Deadlock-Free Scheduling

Reference	**Scope**	**Objective Function**	**Approach**
Fahmy et al. (2009)	Deadlock-free scheduling capable of reacting to uncertainties and disruptions for flexible job shops while utilising the available flexibility in the system	MS	RM
Yoon (2010)	Deadlock avoidance policy with machine breakdown and flexible job routing in robotic manufacturing cells	MS	GA, PN
Xing et al. (2012)	Deadlock-free scheduling for manufacturing systems with shared resources and route flexibility	MS	GA, PN
Zhang et al. (2009)	Deadlock-free job shop scheduling with sequence-dependent setup times and no flexible job routing	MS	BB, TPN
Fahmy et al. (2008)	Deadlock-free scheduling of flexible job shops with limited capacity buffers	MFT MS	MP, RM

Fahmy et al. (2010)	Deadlock-free scheduling in manufacturing cells with different scenarios of capacity buffers and no flexible routing	MFT	MP
Mejía et al. (2009)	Scheduling for variety of manufacturing environments (classical job shop, flexible job shop, and flexible manufacturing scheduling problems) with consideration to deadlock-freeness and blocking	MS	A*, PN
Dashora et al. (2007)	Deadlock-free scheduling in an automated manufacturing system with dynamic routing and limited buffer capacity	MS	TPN, Automata
Huang et al. (2008)	Deadlock-free job-shop scheduling problem for challenging and complex system with TPN	MS	TPN
Xiong et al. (1996)	Search for an optimal or near-optimal deadlock-free schedule with hybrid algorithm representing flexible manufacturing systems with routing flexibility	MS	HH, PN
Xiong et al. (1997)	Explore the advantage of PN to find deadlock-free scheduling of an automated manufacturing system when considering routing flexibility and many resources in the manufacturing cell	MS	HH, PN
Damasceno et al. (1998)	Scheduling for deadlock-free in manufacturing systems with multiple resources	MS	Heuristic, PN
Huang et al. (2004a)	Deadlock-free scheduling method for automated manufacturing systems with limited central buffers which gives the ability to determine the number of buffers which help in the design process	MS	GT, GA
Mati et al. (2001)	Deadlock-free scheduling using model that represents multiple resources and the inherent hold while wait condition	MS	TS, DG
Liljenvall (1999)	Deadlock-free scheduling algorithm for production systems with limited buffers	MS	Automata

Golmakani et al. (2006)	Scheduling and control of FMCs considering different part-processing requirements	MS	Automata
Fahmy et al. (2007)	Deadlock-free scheduling job shop problem with limited capacity buffers	AFT MS	MP, RM
Ben Abdallah et al. (1998)	Deadlock-free scheduling with efficient search algorithm utilizing PN	MS	BB, PN
Gang et al. (2004)	Deadlock-free scheduling for automated production cell with limited buffer	MS	GA, PN
Shi et al. (2005)	Deadlock-free scheduling problems in FMC with no buffers	MS	BS, PN
Gang et al. (2002)	Deadlock-free schedules in an flexible manufacturing system with no flexible routing	MS	GA, PN
He et al. (2006)	Deadlock-free scheduling flexible manufacturing systems model using PN	MS	A*, PN
Ramaswamy et al. (1996)	Deadlock-free schedules for automated manufacturing workstations considering buffers and material handling	MFT	MP, LR
(Ben Abdallah et al. (2002)	Deadlock-free scheduling problem in flexible manufacturing systems	MFT	TPN
Chen et al. (1994)	Reactive scheduling in flexible manufacturing systems with disturbing events and deadlocks when considering many resources	MS	A*, PN
ElMekkawy et al. (2003)	Real-time reactive scheduling for flexible manufacturing systems considering deadlocks, flexible routing, and disturbing situations	FT MS AMU	TPN
Sabuncuoglu et al. (2003)	Simulation based reactive scheduling problems for flexible manufacturing systems with consideration to deadlock-freeness	MFT	BS

* AMU: Average Machine Utilisation, FT: flow time, MS: Makespan, MFT: Mean Flow Time, TPN: Timed Petri Nets, PN: Petri Nets, BB: Branch and bound, GA: Genetic Algorithm, MP: Mathematical Programming, RM: Rank matrices, HH: Hybrid Heuristic, TS: Taboo Search, BS:

Beam Search, LR: Lagrangian relaxation heuristic, GT: Graph-theoretic approach, DG: Disjunctive Graphs

The most common objective functions considered in the literature are the minimization of mean flow time (MFT) and makespan (MS). The MFT is considered in Ramaswamy et al. (1996), Fahmy et al. (2008), Fahmy (2009), and Fahmy et al. (2007). Minimising the MS is considered in Shi et al. (2005), Damasceno et al. (1998), Golmakani et al. (2006) and Dashora et al. (2007). These two objectives are best known to reflect many manufacturing indicators as mentioned by Fahmy (2009). Overall, the literature review reveals that the performance measures have an impact on the computational complexity and resources' utilization. From a computational point of view, Yang et al. (2011) report that optimization of a given performance measures is not always easy. In FMCs especially, El-Tamimi et al. (2012) mention the difficulty in calculating the performance due to the complex interaction between the resources in the manufacturing. For example, the MFT is represented in the literature using explicit expression, which requires a certain number of iterations, Yang et al. (2011). When the performance measure requires a considerable number of iterations, it increases the complexity of the problem and results in extensive computation. Specifically, the flow time is the sum of the completion times of all the jobs' last operations. Therefore, MFT requires many calculations which means that MFT is not easy to compute. As found in the literature that deal with optimisation algorithms, when the computational effort is different for various performance measures, there will be varying costs for the scheduling problem. On the other hand, the existing literature confirms that the performance measure of a FMC is directly related to the availability of resources and the cell configuration. This is because the components of a FMC are tightly interconnected, (Montazeri et al. (1990)), and resources such as buffers are used for supportive tasks in the manufacturing cells. Therefore, performance measures can have a huge influence on the resource utilization. This influence indirectly means that the type of configuration of a FMC should align with the manufacturing objectives. For example, the MFT is reflective for built-up inventories in the system. If a process planner or a production manager aims to reduce the number of inventory on the production floor, the best objective is to minimize the MFT. A minimum MFT usually implies less delay of the jobs in the system.

When considering the scheduling problem, the FMC problems in the literature are formulated as (*j* x *m*) problems where *j* is the number of jobs and *m* is the number of machines. The data are given as *j* jobs {1, 2... *j*} that have to be processed in *m* machines {1, 2...*m*}. Each job *j* has multiple operations {1, 2... *o*}. The number of operations in the system is equivalent to the number of machines. A (10 *x* 10) problem is the largest problem considered in the literature as found in Liljenvall (1999), Huang et al. (2004b), and Huang et al. (2004c). However, in common practice, the number of machine in FMCs is less than or equal to five as reported by Fahmy (2009) or six as reported by Sabuncuoglu et al. (2003). Matta et al. (2005) report that FMCs with more than eight machines are uncommon and difficult to design. Thus, most research considers a smaller group of machines. The parameters consider processing times and jobs routing. The justification is based on the author’s experience in real-world situations similar to the actual data used in Ramaswamy et al. (1996). This data has been duplicated in other studies such as the work done by Fahmy et al. (2010) and Mejía et al (2009).

An examination of the literature shows that most literature that had studied the deadlock problem in FMCs did not consider the routing flexibility. This lack of consideration is because the number of alternative routes for the jobs increases the number of variables; thus, routing flexibility increases the complexity of the problem from a computational point of view. However, routing flexibility has significant benefits for FMCs in the deadlock-free scheduling problems as concluded in the literature that focuses more on the effect of routing flexibility in FMCs.

Many studies assume buffer capacities in FMC to efficiently guarantee deadlock-free schedules such as ElMekkawy et al. (2003), Sabuncuoglu et al. (2003), Liljenvall (1999), Huang et al. (2004a), Fahmy et al. (2008), Fahmy et al. (2010), Fahmy et al. (2007), and Ramaswamy et al. (1996). Piroddi et al. (2008) find that the existence of limited buffers in the system results in heavy resource sharing in the manufacturing cell. The scheduling problem with buffers solved in past literature considers many conditions such as the limited capacity of the buffers (buffer size) and the type of buffers (intermediate, central, input). The buffer related factors has a significant effect on the computational complexity and the performance of the manufacturing cell. Obviously, a model with an infinite capacity buffer and a model with very limited buffer capacity will not react in the same way. Similarly, a bufferless model and a model with limited buffer capacity will not react in the same way. Sabuncuoglu et al. (2003) measure the performance of generated deadlock-free schedules for FMCs with limited buffer capacity and test different buffer sizes from a computational aspects. The authors show that limited capacity buffers may result in an excessive amount of computation times. Further, when considering different configurations of the buffer, Ramaswamy et al. (1996) propose three scenarios for the manufacturing cell: infinite capacity buffers, a bufferless cell, and the existence of intermediate buffers to swap jobs between machines. In the proposed model for the intermediate buffers, utilization is ensured by the machine constraints that allow jobs to be swapped between pairs of machines through the intermediate buffers at any time. It concludes that the introduction of the buffer increases the computational complexity of the problem; however, the gain on the solution value is clearly noticeable compared to the bufferless manufacturing cell. Fahmy et al. (2008) propose unit capacity input buffers at each machine and a central buffer with finite capacity that is utilized by all the machines. In the performance evaluation, the authors note that the input buffers and central buffer models obtain better solutions than the intermediate buffer scenario presented by Ramaswamy et al. (1996). The input buffers and the central buffer take relatively more time to find the solutions especially for larger problem sizes. This increase in time is attributed to the additional number of variables added to represent the input buffers and the central buffer. That study shows also that there are some cases where the input buffer model finds better solutions than the central buffer. The authors conclude that the limited capacity of the buffers prevent jobs from holding the machines and serve for deadlock-free schedules more efficiently. Fahmy et al. (2010) models the intermediate buffers with arbitrary capacity and then a central buffer with arbitrary capacity. Computational experiments are established to study the efficiency and computational performance of the cell configurations with the proposed arbitrary capacity intermediate buffers and arbitrary capacity central buffer models. The experiments also include the model with intermediate buffers to swap jobs between machines developed by Ramaswamy et al. (1996), an improved model of the intermediate buffers to swap jobs between machines developed by Ramaswamy et al. (1996), and the unit capacity input buffers proposed by Fahmy et al. (2008). From the result, the authors conclude that the objective function values and the solution time make a difference between the studied models.

2.3. Dual Gripper Robots in Robotic Cells

A single-gripper robot, which holds only one job at a time, is widely used in many manufacturing tasks. Many manufacturers also use dual grippers for robots. However, the gains attainable are unsupported by concrete scientific findings or analysis for deadlock-free scheduling problems. All the available literature dealing with the deadlock situation for robots as a material handlingfunctionconsider a single gripper for the robots. Furthermore, to the best of the author'sknowledge, no literature is seen that presents any analysis for dual-gripper robots for the scheduling problem with deadlock resolution in FMCs.

There is extensive literature for dual-gripper robots in FMCs in flow shop problems. The scheduling problems in robotic cells are built around the classification scheme scheduling problem given by Dawande (2007). The type of parts is either identical parts with the same processing times or multiple parts with different processing times. However, whether identical or multiple parts, the parts have the same route. The objective in most of the earlier literature is to find the optimal k-unit cycle for the robotic cell when maximising the throughput rate. Dawande (2007) stated that the scheduling problem in robotic cells is strongly NP-hard.

Su et al. (1996) are pioneers in the use of dual-gripper robots in FMCs. Their goal is to provide insights for manufacturing system designers in the use of a dual-gripper robot. From the solution provided and the numerical examples presented, it is clear that the dual-gripper robot is more effective than a single-gripper robot. The authors recommend the use of the dual-gripper robot in most production systems to ensure the material handling function. Since then, many studies are addressing the scheduling problems for dual-gripper robots in a FMC environment such as the analysis done by Drobouchevitch et al. (2006), Geismar et al. (2006), Sethi et al. (2001), and Sriskandarajah et al. (2004).

Drobouchevitch et al. (2006) study the scheduling problem of the cyclic production of a family of identical parts in a FMC served by a dual-gripper robot. The authors conclude that dual-gripper robots improve cell productivity. The authors prove that a dual-gripper robot in a bufferless configuration is more efficient than a single-gripper robot with output buffers. Geismar et al. (2006) propose an optimal cycle to maximize the throughput for dual-gripper robotic cells with or without parallel machines producing identical parts at each processing stage. The authors stress the productivity gains from implementing dual-gripper robots. Sethi et al. (2001) consider the problem of scheduling robot moves in dual-gripper bufferless robotic cells to maximize the throughput. Through an examination of the results, the authors find that the productivity of dual-gripper robots can double the productivity of a single-gripper robot. Sriskandarajah et al. (2004) schedule multiple parts in a traditional bufferless robotic cell served by a dual-gripper robot to maximize the throughput. From the solution procedure and test results to estimate the productivity improvement, the authors provide strong proof that using the dual-gripper robots results in improved productivity over the use of single-gripper robots. Su et al. (1996) analyze the scheduling problem of a proposed tightly-coupled serial production system with deterministic processing using a dual-gripper robot. The authors conclude that the dual-gripper robot is more effective than the single-gripper robot. Recently, Geismar et al. (2008) provide optimal cycles for single-gripper robot and dual-gripper robot moves to address the problem of scheduling identical parts in bufferless robotic cells. The results show an improvement in productivity that can be gained by using a dual-gripper robot rather than a single-gripper robot. Geismar et al. (2011) study the productivity improvement for identical parts production when using a dual-gripper robot in robotic cells with unit-capacity input and output buffers at each machine. When qualifying the cyclic production solution, the results prove that a robotic cell with a dual-gripper robot and input/output buffers performs much better than a dual-gripper robot without buffers at the machines. Dawande et al. (2009) address multiple part-type production in FMCs with two different models. The first model is a single-gripper robot with a unit-capacity output buffer at each machine. The second model is a bufferless robotic cell with a dual-gripper robot. It concludes that the two scenarios have the same maximum throughput. However, an economic analysis shows that the purchasing and maintenance of the output buffer is less than the cost of a dual-gripper robot. This difference is justified by the advanced features of the dual-gripper robot compared to output buffers.

2.4. Constraint Programming

The constraint programming emerges from recent developments in the artificial intelligence science. Constraint programming is introduced in the operation research field to solve manufacturing problems. Constraint programming is a flexible modeling tool used to express model constraints in a general logical structure. Baptiste et al. (2001) mention that constraint programming has proven its effectiveness for scheduling applications. More importantly, Khayat et al. (2006) find that the constraint programming computational studies for the scheduling problem show their excellence and effectiveness even for larger problems. Pinedo (2012) provides a good background on constraint programming. After a survey of the literature, only a few studies are found that consider the problem of scheduling in FMCs using constraint programming as a tool such as in the work by Zeballos (2010), Zeballos et al. (2010), and Khayat et al. (2006). Khayat et al. (2006) propose a constraint programming model for obtaining integrated production and material handling schedules in a production system that have machines and automated guided vehicles. The proposed model is for a job shop with the goal of minimizing the MS. The constraint programming modeling and solving are done through the OPL constraint programming language. The results show the suitability of constraint programming in this type of production scheduling problems. Interestingly, this study also conducts a comparison between a constraint programming and a mathematical programming approach developed for the same scheduling problem. The results from a group of generated test problems show that the performance of constraint programming and mathematical modeling might be mutable. The authors also conclude that the two methods are not necessarily applicable for the same problem type and depend on the type of resources and their influence on the proposed algorithm. Zeballos et al. (2010) propose a constraint programming model for the scheduling of FMCs with machines and tool limitations. As noted in that paper, the formulation presented uses the OPL constraint programming language. This work has the ability to integrate many variables in one model using this programming language. The scheduling model has integrated tool allocation, machine loading, and part routing. The authors conclude that the proposed constraint programming method is able to reduce the dimensionality of the model's variables while other methods such as mathematical modeling with mixed integer programming significantly increase the number of variables. This increase results in added computational complexity. For example, when generating different size problems with different objective functions, the authors compare the computational performance of the objective functions. Their conclusion is that the constraint programming does maintain an excellent computational performance; however, the computational effort varies between the different objective functions. Yet, the results from the test problems show that initial feasible solutions are found in much less times and attained solutions require very minimal computational efforts. Although the attained solutions with the proposed method are not optimal all the time, the authors note that the overall quality of the solution is satisfying. Zeballos (2010) use constraint programming to model tool allocation and production scheduling in FMCs. The formulation uses the same constraint programming language proposed by Zeballos et al. (2010). Constraint programming approach is chosen in that study to overcome the difficulties surrounding modeling FMC features as the modeling task becomes very difficult with other scheduling approaches, for example the mathematical modeling approach. The model takes into account the number of limitations imposed by the tools in the system, tools' lifetime, number of tool copies, tool planning and allocation, machine assignment, part routing, and task timing decisions. The scheduling problem is done with different objective functions and solved using different search strategies. When generating the different size problems, the authors consider different variables related to number of jobs, operations, and machines. The conclusion is that even with this complicated problem and larger dimensionality of the model variables, the constraint programming method demonstrates efficient results.

2.5. Conclusions

Scheduling with deadlock resolution is an extensive area of research. The literature survey can be summarized as follows. The scheduling with deadlock resolution in FMCs is a two-component problem that consists of generating deadlock-handling policies and optimization of an objective function. The scheduling with deadlock resolution is computationally challenging since it isan NP-hard problem, Ponnambalamam et al. (2009). Deaddlock-freeness can be tackled with many scheduling approaches including mathematical, heuristic, and meta-heuristic approaches. The mathematical approach gives an exact and an optimal solution; however, it is limited by the size of the problem. The heuristic and meta-heuristic approaches solve larger size problems and are mainly combined with mathematical tools such as Petri Nets. The power of constraint programming has never been used in FMCs when deadlocks are considered. Constraint programming has the power and the capabilities for modeling, analyzing, and synthesizing control laws with regard to the deadlock-freeness in FMCs when compared to traditional mathematical modeling, Hu et al. (2009). This lack of consideration of constraint programming in the literature is because constraint programming is relatively new for applications in FMCs' scheduling. The literature survey illustrates how the constraint programming methodology is tested and has proven its performance for modeling and computing scheduling problems in manufacturing cells. Figure 3 summarizes the reviewed papers for deadlock-free scheduling.

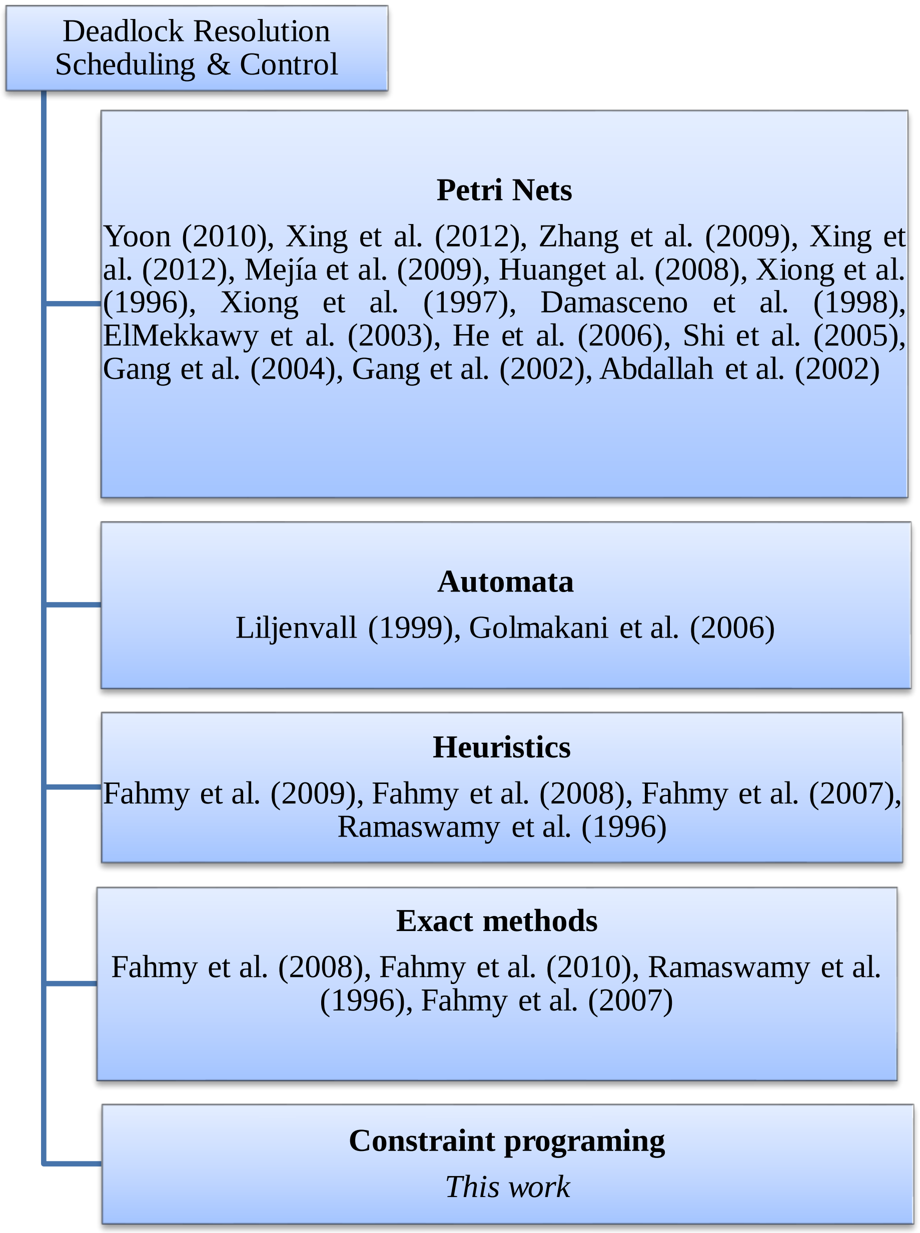

Deadlock Resolution
Scheduling & Control
Petri Nets
Yoon (2010), Xing et al. (2012), Zhang et al. (2009), Xing et al. (2012), Mejía et al. (2009), Huanget al. (2008), Xiong et al. (1996), Xiong et al. (1997), Damasceno et al. (1998), ElMekkawy et al. (2003), He et al. (2006), Shi et al. (2005), Gang et al. (2004), Gang et al. (2002), Abdallah et al. (2002)
Automata
Liljenvall (1999), Golmakani et al. (2006)
Heuristics
Fahmy et al. (2009), Fahmy et al. (2008), Fahmy et al. (2007), Ramaswamy et al. (1996)
Exact methods
Fahmy et al. (2008), Fahmy et al. (2010), Ramaswamy et al. (1996), Fahmy et al. (2007)
Constraint programing
This work

Figure 3. Deadlock-free Scheduling Approaches in Literature

Many configurations of buffers are proposed in the previous literature to ensure the deadlock safe state in FMCs. The studies examine the following: the existence of one central buffer serving the whole system, an input buffer associated with each machine, intermediate buffers between the machines where jobs can reside temporarily, and intermediate buffers between the machines for swapping jobs between the machines. Clearly, the existence of the buffers in the manufacturing cell presents many complexities. First, the modeling task of the buffers is not easy. Second, the existence of the buffers increases the computational complexity of the scheduling problem. In this sense, each buffer configuration performs differently in terms of the solution quality and computational times. The computational effort depends on the number of constraints and variables and the search space. Third, the buffer sizes and buffer configurations have a great impact on the solution quality. Fourth, the existence of limited buffer capacities requires complex strategies for the deadlock control especially since deadlocks can still occur when the buffers reach their capacity.

Interestingly, all available literature that deals with the different configurations of the buffers uses single-gripper robots for material handling. Therefore, the dual-gripper robots have never been utilized for deadlock resolution in FMCs. Many of the reviewed studies investigate productivity improvement when using dual-gripper robots in flow shops. Figure 4 summarises a review of studies for deadlock-free scheduling considering limited buffer capacities and positions the approach followed in this thesis in considering new resource types (material handling devices) besides machines and buffers.

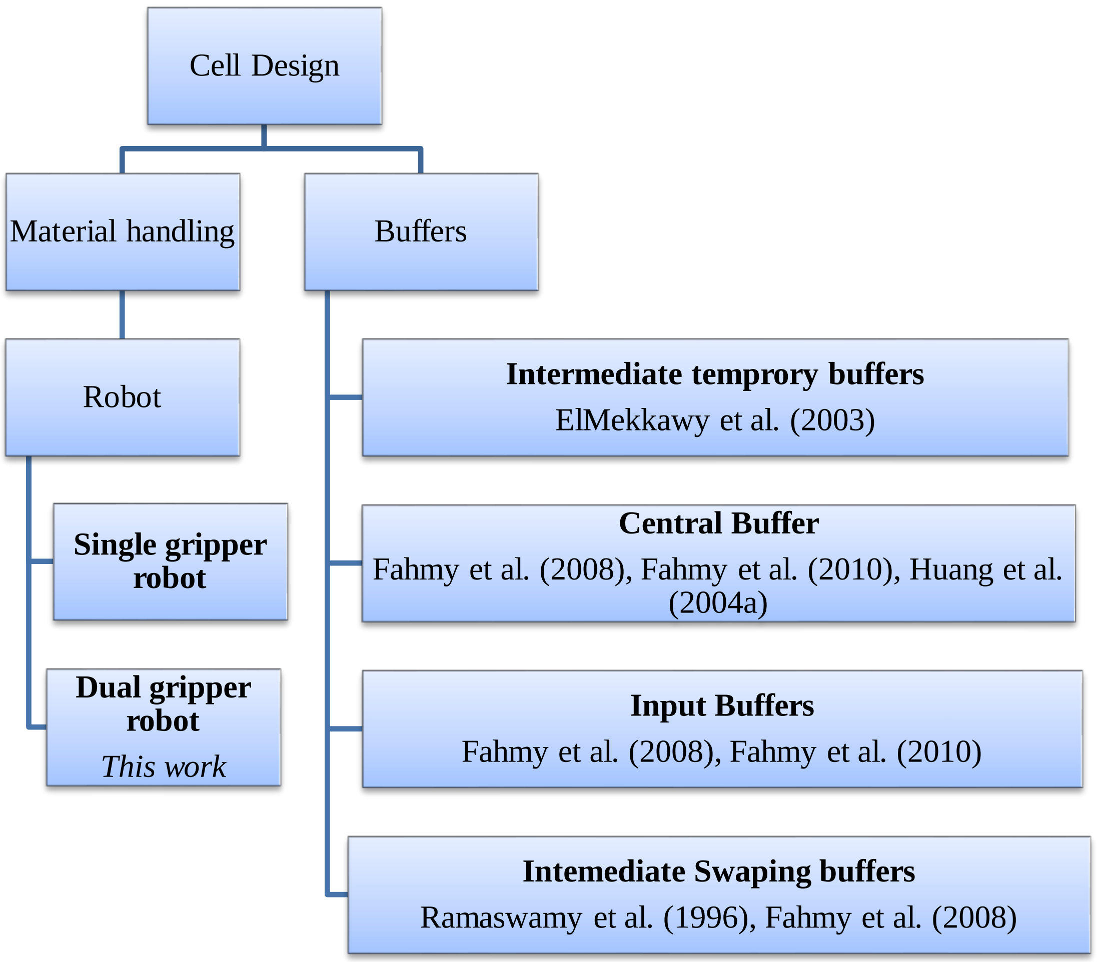

Cell Design
Material handling
Buffers
Robot
Single gripper robot
Dual gripper robot
This work
Intermediate temprory buffers
ElMekkawy et al. (2003)
Central Buffer
Fahmy et al. (2008), Fahmy et al. (2010), Huang et al. (2004a)
Input Buffers
Fahmy et al. (2008), Fahmy et al. (2010)
Intemediate Swaping buffers
Ramaswamy et al. (1996), Fahmy et al. (2008)

Figure 4. Used Resources in Literature

Clearly, the literature considers many variables related to FMCs such as different objective functions to measure the performance. However, behaviour of the FMCs with consideration of the deadlock situation has never been formally evaluated when dual-gripper robots are considered. Furthermore, the routing flexibility has been considered in only a few of the studies. This conclusion from the review gives researchers incentives to try new approaches to solve the deadlock scheduling problem. The next Chapter present the deadlock-free scheduling approach developed in this research.

Chapter 3 DEADLOCK-FREE SCHEDULING OF THE PROPSOED MANUFACTURING CELLS

3.1. Introduction

Scheduling of manufacturing cells refers to sequencing activities and tasks in the manufacturing cell. Scheduling in FMC is not an easy task. This is mainly due to the challenges surrounding the modeling and computation efforts. A variety of solution tools have been proposed in the literature for modeling the scheduling problems of manufacturing cells. To the best of the author'sknowledge, the constraint programming (CP) methodology has never been used for deadlock-free scheduling problems. CP is used in many scheduling applications of manufacturing cells and it provides interesting results. The reader can refer to Kim et al. (1998) for a comprehensive background of CP concepts.

IBM ILOG CP Optimizer (CPO) is a scheduling language that makes use of CP algorithms to solve scheduling problems. The CPO has an efficient formalism to model scheduling problems. The CPO has built-in scheduling functions that can compactly model difficult constraining expressions. A good introduction to scheduling functions with a CPO is given in IBM Corp (2011).

A CPO model consists of a preprocessing stage and a processing stage. The pre-processing stage represents the data construct and declarations of the variables. The processing stage is the core of the model where the objective function and the constraining expressions are presented. The preprocessing stage and processing stage are presented in the same program.

This chapter details the deadlock-free scheduling of manufacturing cells pursued in this thesis. The first section introduces the reader to the basic language construct and functions of the CPO. Then, a basic scheduling example is discussed to familiarize the reader with the CPO language construct. Next, details about the structure of the complete models for the scheduling problems are given. The last section presents the complete models.

3.2. Construct of the CPO

The CPO is based on the concept of intervals of time. This is quite different from scheduling with mathematical modeling where the scheduling is based on integer or continuous values to represent time. Consequently, the decision variables and solutions for the scheduling problem with a CPO are not constructed the same way as in mathematical modeling.

The CPO models the decision variable as **interval variables**. Each interval has a start and an end time. In case of scheduling in manufacturing cells, decision variables are the different activities performed in the cells such as buffering and processing. The solution for scheduling manufacturing cells problems are sequence of activities. The CPO denotes these sequences of activities in the manufacturing cells as **sequence variables**. A sequence variable is a series of sequenced interval variables.

The CPO language has different types of built-in scheduling functions that are used to build the model formalism by manipulating the decision variables and the sequences variables. The reader is referred to work by Laborie (2009) for a detailed recap with illustrative examples of the CPO scheduling concept.

The formalism used in this thesis uses the following: precedence constraints, cumulative constraints, no-overlap constraints, and functions that operate intervals. Precedence constraints are scheduling constraints that are generated between the decision intervals. Cumulative constraints are scheduling constraints that can model varying quantities over time. No-overlap constraints are scheduling constraints that allow distance between the decision intervals in sequence variables. Functions that operate intervals allow accessing end or start times of the decision intervals. Table 2 below recaps the functions used in the thesis as described in IBM Corp (2011). The next section presents a basic example to demonstrate the CPO construct in scheduling problems in manufacturing cells.

Table 2. Functions Used in the Thesis

Function	Purpose

endBeforeStart
endAtStart
startAtEnd
	Precedence constraint to restrict the relative positions of intervals.

noOverlap	Special constraint used to prevent intervals in a sequence from overlapping.
endOf	Function that accesses the end time of an interval.
endOfPrev	Function that returns the end of the previous interval in a given sequence.
cumulFunction	Function that creates a cumulative function to model a quantity that varies over time and whose value depends on other decision variables of the problem.

3.3. Basic example

The following example is a basic structure of a job shop scheduling problem. A manufacturing cell has three machines (Face, Drill, and Pocket). Each machine can perform a distinct operation. “Face” performs facing operations. “Drill” performs drilling operations. “Pocket” performs pocketing operations. The manufacturing cell produces three different jobs (Job1, Job2, and Job3). The jobs are mixed and require all the three processes, namely, facing, drilling, and pocketing. Each process takes a considerable interval of time. The processing time and jobs routes of the above manufacturing scenario are given as follows. Numbers shown in brackets are the processing times on each machine.

Table 3. Processing Routes and Times of the Basic Problem 3J x 3M

Jobs	Operations		
	Machine (Processing Time)		
	1	2	3
Job1	Drilling (12)	Facing(10)	Pocketing (13)
Job2	Facing (6)	Drilling (12)	Pocketing (13)
Job3	Pocketing (5)	Drilling (73)	Facing (11)

Table 3 is the processing time and jobs routes of the three jobs and three machines (3J x 3M) problem used in this example. The processing times of all the operations are different. The jobs have different routings in the system.

The following code is a simple model of the above example using the CPO language. The model declares the sets of machines (line 2), the sets of jobs (line 3), and the data concerning the duration of each operation of a job in a machine (line 4). The interval variables (dvar interval) in line 5, is the series of operations of the jobs in the machines that have fixed time duration (size) read from the previous line. The sequence variables (dvar sequence) are the series of the machines where the interval variables are sequenced (line 6). The objective in the model is to schedule the jobs in the manufacturing cell by assigning the operations to the machines and minimizing the end time of the last job in the schedule (line 7). The sequence variables have a no-overlap constraint which means that the interval variables, in line 6, have to be sequenced one after the other without overlapping (line 10). Finally, the operations of the interval variables have precedence constraints as these operations have to follow a predetermined machining order (lines 12-17). For example, line 12 forces the interval variable representing the drilling operation on Job1 to end before the start of the interval variable presenting the facing operation of Job1.

The formalism of the model uses the following:

- The *endOf* function in the objective function to access the end of the series of the variable intervals.
- The *noOverlap* function to constrain the presence of the interval variables on the sequence variable.
- The *endBeforeStart* function to set precedence constraints among the interval variables.

```
1.   usingCP;
2.   {string}Parts= {"Job1","Job2","Job3"};
3.   {string}Machines= {"Face","Drill","Pocket"};
4.   intduration[Parts][Machines]=[[10,12,13],[6,12,13],[11,10,5]];
5.   dvarintervalOperation[jinParts][minMachines]sizeduration[j][m];
6.   dvarsequenceMachine[minMachines]inall(jinParts,oinMachines:m==o)Operation[j][m];
7.   minimizemax(jinParts,minMachines)endOf(Operation[j][m]);
8.   subjectto{
9.     forall(minMachines)
10. noOverlap(Machine[m]);
11. forall(jinParts,minMachines)
12. endBeforeStart(Operation["Job1"]["Drill"],Operation["Job1"]["Face"]);
13. endBeforeStart(Operation["Job1"]["Face"],Operation["Job1"]["Pocket"]);
14. endBeforeStart(Operation["Job2"]["Face"],Operation["Job2"]["Drill"]);
15. endBeforeStart(Operation["Job2"]["Drill"],Operation["Job2"]["Pocket"]);
16. endBeforeStart(Operation["Job3"]["Pocket"],Operation["Job3"]["Drill"]);
17. endBeforeStart(Operation["Job3"]["Drill"],Operation["Job3"]["Face"]);
18. }
```

The decision variables in the above model are the interval variables in line 5 representing the processing operations of the jobs in the machines and the solution is the sequence of these interval variables in the machines as in line 6. The model uses line 10 to force that the machines are used exclusively. The model uses line 12-17 to force the operation routing of the jobs. For example, line 12 forces the drilling operation on Job1 before the facing operation. The solution for this example is as follows.

Table 4. Solution of the Basic Problem 3J x 3M

Machines (size 3)	Decision variable	Position	Start	End	Size
Face	Operation["Job2"]["Face"]	0	0	6	6
	Operation["Job1"]["Face"]	1	12	22	10
	Operation["Job3"]["Face"]	2	34	45	11
Drill	Operation["Job1"]["Drill"]	0	0	12	12
	Operation["Job2"]["Drill"]	1	12	24	12
	Operation["Job3"]["Drill"]	2	24	34	10
Pocket	Operation["Job3"]["Pocket"]	0	0	5	5
	Operation["Job1"]["Pocket"]	1	22	35	13
	Operation["Job2"]["Pocket"]	2	35	48	13

The equivalent Gantt chart of the solution is shown below:

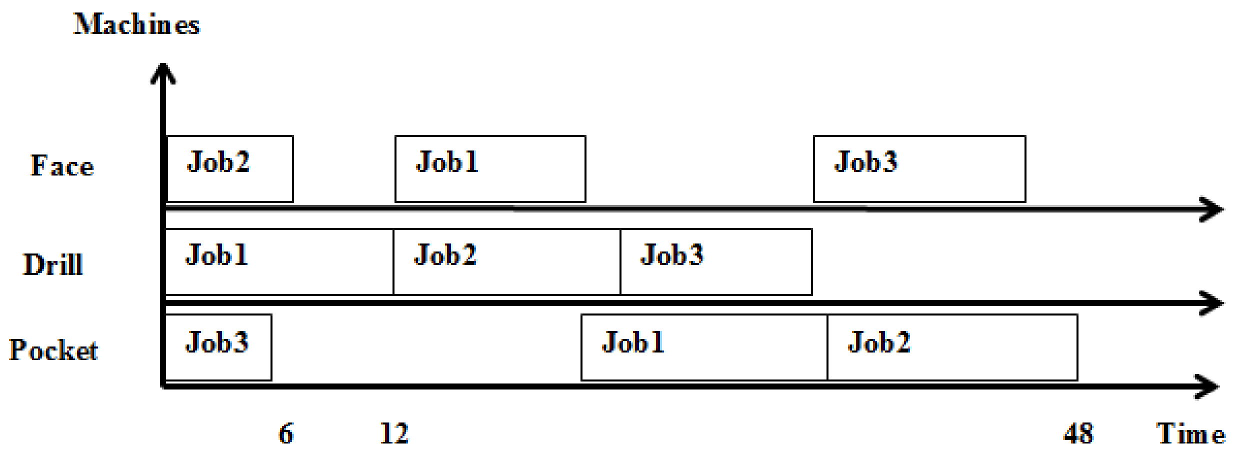


Figure 5. Gantt Chart for the Basic Problem 3J x 3M

After running the equivalent model for this 3J x 3M problem, the obtained solution is given in Table 4. The equivalent Gantt chart of the solution is given in Figure 5. The optimal solution obtained for this example is found to be 48 sec. In the chart, three sequence variables are presented: the sequence variable that represents the “Face” machine, the sequence variable that represents the “Drill” machine, and the sequence variable that represents the “Pocket” machine. Similarly, twelve decision variables are present. This number of decision variables represents the total number of the operations executed in the manufacturing cell. The processing intervals are ordered in the machines' sequences. Each machine sequence has three operations. The number of the operations in each machine is equivalent to the number of the jobs.

The processing intervals Operation["Job2"]["Face"], Operation["Job1"]["Face"], and Operation["Job3"]["Face"] are present respectively in the sequence of the "Face" machine. This sequence of intervals means that Job2, Job1, and Job3 are executed in the "Face" machine in this order. The processing intervals Operation["Job1"]["Drill"], Operation["Job2"]["Drill"], and Operation["Job3"]["Drill"] are present respectively in the sequence of the "Drill" machine. This sequence of intervals means that Job1, Job2, and Job3 are executed in the "Drill" machine in this order. The processing intervals Operation["Job3"]["Pocket"], Operation["Job1"]["Pocket"], and Operation["Job2"]["Pocket"] are present respectively in the sequence of the "Pocket" machine. This sequence of intervals means that Job3, Job1, and Job2 are executed in the "Pocket" machine in this order.

In each sequence, the processing intervals are not overlapping which shows the effect of the *noOverlap* function. For example, in the sequence variable representing the "Face" machine where Job2, Job1, and Job3 are sequenced respectively, Operation["Job2"]["Face"] starts at time 0 sec and ends at time 6 sec, Operation["Job1"]["Face"] starts at time 12 sec and ends at time 22 sec, and Operation["Job3"]["Face"] starts at time 34 sec and ends at time 45 sec. These starts and end points of the intervals prove that only one processing interval is present at the time.

The operations of the same job follow the precedence constraining expression in the model. The *endBeforeStart* precedence constraining function in the constraining expressions indicates that for two manipulated decision intervals, the end of the first interval is less than or equal to the start of the second interval. For example, the *endBeforeStart* in line 12 manipulates the intervals Operation["Job1"]["Drill"] and Operation["Job1"]["Face"] . Operation["Job1"]["Drill"] ends at 12 sec and Operation["Job1"]["Face"] starts at 12 sec as seen in Figure 5. The end of Operation["Job1"]["Drill"] is less than or equal to the start of Operation["Job1"]["Face"]. Therefore, the *endBeforeStart* has its effect in the chart. Similarly, the *endBeforeStart* in line 14 manipulates the intervals Operation["Job2"]["Face"], and Operation["Job2"]["Drill"]. In the chart, Operation["Job2"]["Face"] ends at 6 sec and Operation["Job2"]["Drill"]starts at 12 sec.The end of Operation["Job2"]["Face"] is less than or equal to the start of Operation["Job2"]["Drill"]. Therefore, the effect of the *endBeforeStart* function is proven in the schedule for this problem (Figure 5).

There are other CPO precedence functions that can be used in the constraining expression given in lines 12-17. For example, *endAtStart* is a precedence constraint that makes relative representation between the interval variables. However, its effect is different from the *endBeforeStart* function. For example, for two intervals manipulated by the *endBeforeStart*, the first interval ends before the second interval starts. The end of the first interval is less than or equal to the start of the second interval. In contrast, for two intervals manipulated by the *endAtStart*, the first interval ends exactly at the start of the second interval. The end of the first interval is equal to the start of the second interval. To illustrate the effect of the *endAtStart*, one needs to reconsider the code presented previously and change the *endBeforeStart* function in lines 12-17 to an *endAtStart* as presented below:

```
11. forall(jinParts,minMachines)
12. endAtStart (Operation["Job1"]["Drill"],Operation["Job1"]["Face"]);
13. endAtStart (Operation["Job1"]["Face"],Operation["Job1"]["Pocket"]);
14. endAtStart (Operation["Job2"]["Face"],Operation["Job2"]["Drill"]);
15. endAtStart (Operation["Job2"]["Drill"],Operation["Job2"]["Pocket"]);
16. endAtStart (Operation["Job3"]["Pocket"],Operation["Job3"]["Drill"]);
17. endAtStart (Operation["Job3"]["Drill"],Operation["Job3"]["Face"]);
```

The solution of this basic example with the *endAtStart* function follows:

Table 5. Solution of the Basic Problem 3J x 3M using the *endAtStart* Function

Machines (size 3)	Decision variable	Position	Start	End	Size
Face	Operation["Job1"]["Face"]	0	12	22	10
	Operation["Job2"]["Face"]	1	22	28	6
	Operation["Job3"]["Face"]	2	28	39	11
Drill	Operation["Job1"]["Drill"]	0	0	12	12
	Operation["Job3"]["Drill"]	1	18	28	10
	Operation["Job2"]["Drill"]	2	28	40	12
Pocket	Operation["Job3"]["Pocket"]	0	13	18	5
	Operation["Job1"]["Pocket"]	1	22	35	13
	Operation["Job2"]["Pocket"]	2	40	53	13

The equivalent Gantt chart of the solution is as follows:

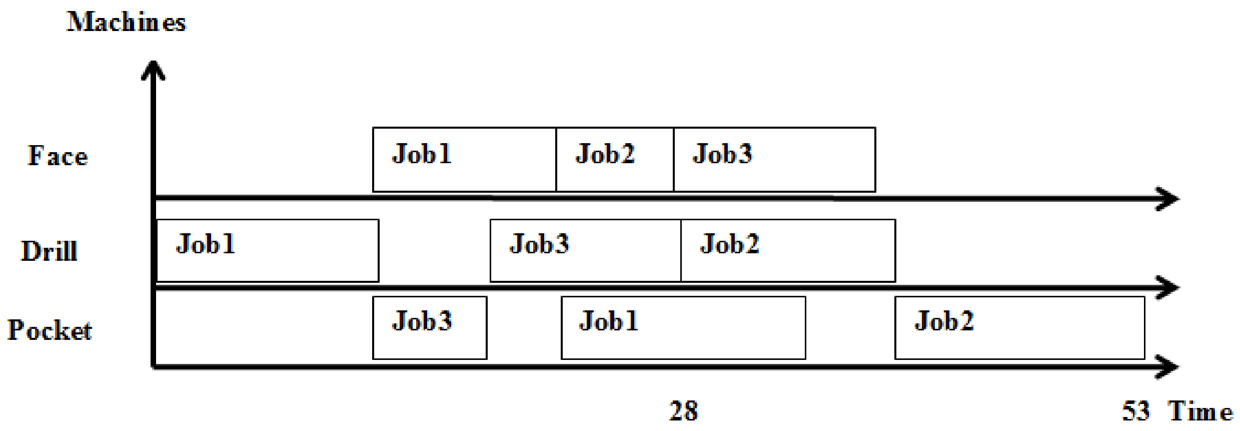


Figure 6. Gantt Chart of the Basic Problem 3J x 3M using the *endAtStart* Function

Table 5 is the solution of the 3J x 3M problem using the *endAtStart* Function. The equivalent Gantt chart of this solution is given in Figure 6. The *endAtStart* function has its effect on the objective value. The optimal solution obtained is now 53 sec. This change in the objective value is due to the change of sequence of the jobs in the machines, and the end times of the operations in the machines. The objective function was sensitive to the sequence of jobs in the machines. To clearly illustrate the effect of the *endAtStart* function, the following example is presented.The *endAtStart* in line 14 manipulate the intervals Operation["Job2"]["Face"], and Operation["Job2"]["Drill"]. In Figure 6, Operation["Job2"]["Face"] ends at 28 sec, and Operation["Job2"]["Drill"] starts at 28 sec. Therefore, the *endAtStart* has its effect. The end of Operation["Job2"]["Face"] is equal to the start of Operation["Job2"]["Drill"].

Scheduling constraints in CPO are not limited to precedence constraints such as the *endBeforeStart* and the *endAtStart* functions. Some scheduling problems may require special CPO functions to model the features of the system. Cumulative functions are the CPO scheduling function that can be used to model cumulative resource. A cumulative resource is a resource that is consumed over a time. For example, machines in manufacturing cells are cumulative resources. When a job is present in a machine, the capacity of the machine is consumed since a machine can process only one job at a time.

In the basic example, the machine exclusivity is modeled with the *noOverlap*. However, the machine exclusivity can be modeled by the CPO cumulative functions. The following code is a model of the basic example using cumulative functions:

```
1.  usingCP;
2.  {string}Parts= {"Job1","Job2","Job3"};
3.  {string}Machines= {"Face","Drill","Pocket"};
4.  intduration[PartsNames][MachinesNames]=[[10,12,13],[6,12,13],[11,10,5]];
5.  dvarintervalOperation[jinParts][minMachines]sizeduration[j][m];
6.  cumulFunctionMachine[minMachines] =sum(jinParts)pulse(Operation[j][m],1);
7.  minimizemax(jinParts,minMachines)endOf(Operation[j][m]);
8.  subjectto{
9.  forall(minMachines)
10. Machine [m] <=1;
11. forall(jinParts,minMachines)
12. endBeforeStart(Operation["Job1"]["Drill"],Operation["Job1"]["Face"]);
13. endBeforeStart(Operation["Job1"]["Face"],Operation["Job1"]["Pocket"]);
14. endBeforeStart(Operation["Job2"]["Face"],Operation["Job2"]["Drill"]);
15. endBeforeStart(Operation["Job2"]["Drill"],Operation["Job2"]["Pocket"]);
16. endBeforeStart(Operation["Job3"]["Pocket"],Operation["Job3"]["Drill"]);
17. endBeforeStart(Operation["Job3"]["Drill"],Operation["Job3"]["Face"]);
18.   }
```

In the above model, Line 6 and line 10 have changed from the initial model. The sequence variables and the no-overlap constraint are no longer modeled. Line 6 is now a declaration for a cumulative function. This declaration identifies the contributing interval to the cumulative function value. The contributing interval is the processing interval. Every time a job is present in a machine, a processing interval is present, and then a cumulative function is incremented by one. When the job leaves the machine, the processing interval is no longer present, and then the cumulative function decreases its value by one. The cumulative function changes its value depending on the presence of jobs on the machines. The cumulative function is indexed by the machine’s name. Each machine has a corresponding cumulative function. Line 10 sets a limit to the cumulative function value. This limit represents the maximum number of the jobs present in a machine at the time. Here, the limit is one job.

When solving the model, the solution given in Table 4 and the equivalent Gantt chart (Figure 5) of the solution remain the same. The flowing figure illustrates the profile of the cumulative functions of the machines.

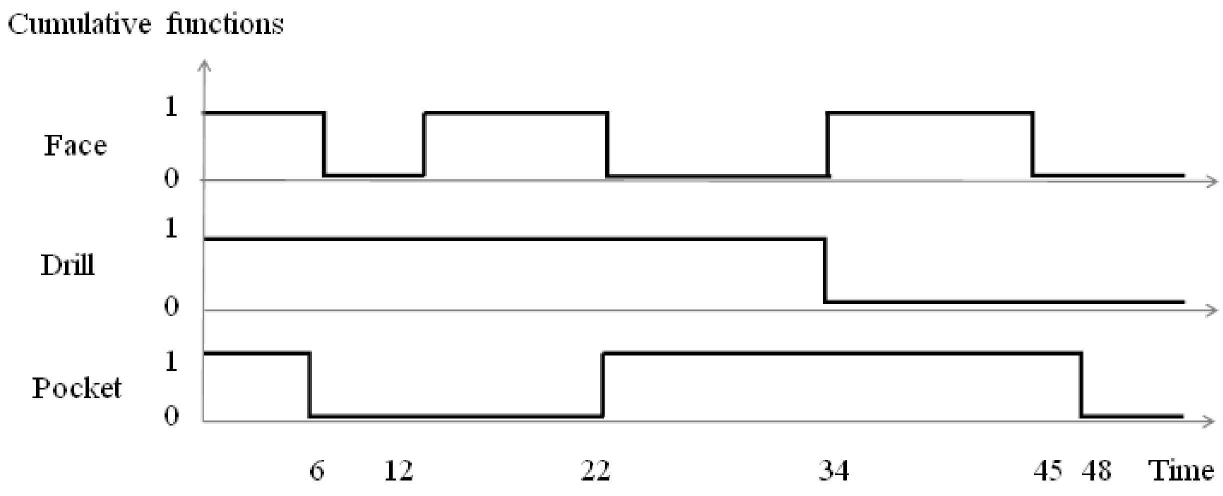


Figure 7. Profile of the Cumulative Functions of the Machines

In the above figure, the corresponding cumulative functions of the machines are presented. The value of each cumulative function reflects the presence of a job in a machine. For example, in Figure 5 Job2 is processed in the "Face" machine between 0 sec and 6 sec, Job1 between 12 sec and 22 sec, and Job3 between 34 sec and 45 sec. In Figure 7, the first interval is present on the "Face" machine between 0 sec and 6 sec. The second interval is present on the "Face" machine between 12 sec and 22 sec. The third interval is present on the "Face" machine between 34 sec and 45 sec. In the corresponding cumulative function for the "Face" machine, the value of the cumulative function is equal to1 between: 0 sec and 6 sec, 12 sec and 22 sec, and 34 sec and 45 sec. The maximum value taken by the cumulative function is one. Therefore, the limit set for the cumulative function value of the machine "Face" has it effect on the number of jobs present in the machine at that time. The effect of the cumulative function is verified for the rest of the machines (Drill and Pocket).

To summarize, this basic example has built the basis for further discussion in this thesis. Only a few scheduling functions are presented. However, there are many other CPO functions that can be used in the scheduling problem. The same CPO scheduling concepts described in this example can be applied to any other scheduling problem for a manufacturing cell. Decision variables are intervals of times. Sequence variable are the solution of the problem. The constraint expressions manipulate the end point and start point of the decision variables.

3.4. Description of the Proposed Models

The models are structured to represent:

- Declarations,
- Objective function,
- Jobs' routing,
- Machines' disjunction,
- Buffers' capacity,
- Deadlock-freeness.

3.4.1. Declarations

The data for the models are the number of jobs, the number of machines, processing time, and job routing. These data are stored in a separate data file. However, the data are read and manipulated in the core of the CPO models. Processing and buffering are the two activities present in the modeled manufacturing cells. Therefore, the decision variables are the intervals representing processing and the intervals representing buffering. The CPO syntax for interval of times is used to model the two decision intervals. Similarly, the solution of the problem is the sequence of the jobs in the machines. Machines are the sequence variables. The CPO syntax for sequence variables is used to model the sequence of the jobs in the machines. Cumulative function is declared to model the varying quantity of the jobs residing in the buffers. Finally, the declaration stage includes a special construct for guiding the solution search engine. The CPO has many parameters that can guide the search engine to find the solution such as the *cp.param.TimeLimit*. This parameter sets limits to the search time forcing the search to stop at a certain time as scheduling problems might otherwise take a considerable amount of time to find the best solution.

3.4.2. Objective Functions

The objective function in all models is to minimize the mean flow time (MFT). The MFT is the average of the completion times for the last operation of the jobs. The objective function expression is modeled using the *endOf* function to access the end times of the operations intervals.

3.4.3. Jobs' Routing

The jobs' routing is the jobs' operations' order in the cell. The jobs routing in the models is done by the CPO functions that can force relativity between intervals of time representing the processing activity. This includes the functions *endBeforeStart* and *endAtStart.* The example in Section 3.3 illustrated the effect of the precedence constraints *endBeforeStart*and *endAtStart* in scheduling problems.

In the formulations with limited buffer capacities, the jobs routing is expressed using the *endBeforeStart* function. For two consecutive processing intervals of the same job, the first interval ends any time before the start of the second interval. Further, the *endBeforeStart* is chosen among other CPO functions that model precedence constraints because it allows the buffer utilization. If the first machine where the first operation is executed is needed by another job, the job uses the assumed buffer space between the two operations.

In the bufferless formulations, the jobs routing is expressed using the *endAtStart* function.When no buffers exist in the system, a job keeps holding a machine until its next processing starts or move immediately to the next downstream machine. Therefore, the expression for the jobs routing in bufferless formulations forces the condition that for two consecutive operation intervals of the same job, the first interval ends exactly at the start of the second interval.

3.4.4. Machines’ Disjunction

The machines’ disjunction (also called machines’ exclusivity) is the machines’ ability to process one job at a time. The proposed models used the *noOverlap* function to force that in each sequence of operation intervals for a given machine only one processing interval is present at a given time. The effect of the *noOverlap* constraint was illustrated in Section 3.3.

3.4.5. Buffers’ capacity

The modeling of the buffers in the manufacturing cells is done with CPO cumulative functions since buffers are cumulative resources. The quantity of jobs residing in the buffers varies over time. Modeling of the buffers with the cumulative function is done as follows:

Declaring the cumulative function, constraining the contributing interval to the cumulative function, and setting limit to the cumulative function value.

- The declaration of the cumulative function names the functions and identifies the contributing interval. Simply put, the contributing interval is the buffering interval. Every time a job is present in a buffer, a storage interval is present, and then a cumulative function is incremented by one. When the job leaves the buffer, the buffering interval elapses, and then the cumulative function decreases its value by one. The cumulative function changes its value depending on the jobs leaving or entering the buffer.

- The constraints of the contributing interval to the cumulative function forces the start and the end of the buffering interval. For two consecutive operation intervals of the same job, the buffering interval starts at the end of the first interval and ends at the start of the second interval. The difference between the end of the first interval and the start of the second interval is the size of the buffering interval. When the next processing of this job starts immediately, the size of the buffering interval is zero. When the size of the buffering interval is zero, it means that the buffer will not be used and the job starts its next process after finishing the previous one. When the size of the buffering interval is greater than zero, it means that the buffer will be used by the job for a period of time equal to the size of the buffering interval.

- Setting limits to the cumulative function value is a constraining expression that bonds the varying value of the cumulative function. This constraint limits the quantity of the jobs residing in the buffers. It forces that the number of jobs residing in the buffer should be less than or equal to the given capacity all the time.

3.4.6. Deadlock-freeness

The circular waits happen between the existence of resources including the buffers and the machines. Deadlock-freeness constraint ensures that circular waits never happen. The deadlock-free constraint ensures that a resource will not release a job until the resource in its next route is available.

The approach used to express the deadlock-freeness in bufferless formulations is similar to the approach used in Fahmy et al. (2009). The model makes use of a variable E shown in Figure 8. The operation completion time of a job on a machine must be greater than at least by "E" when compared to the completion time of the job that precedes the first job on the next machine in its route. E is a small unit of time. Using the concept of CPO intervals, the deadlock-freeness CPO constraint can be formulated. For two consecutive processing intervals of the same job, the end time of the first processing interval must be greater than the end of the previous processing interval of another job proceeding to the next resource by a small unit of time. The result of the deadlock constraints in this bufferless scenario is shown in Figure 8.

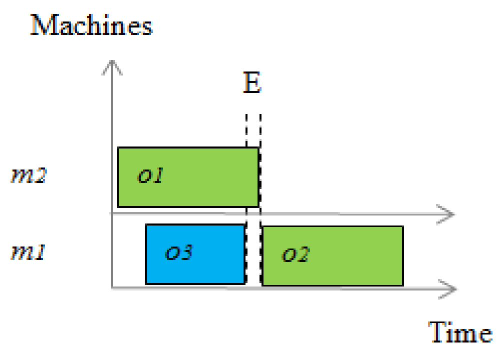
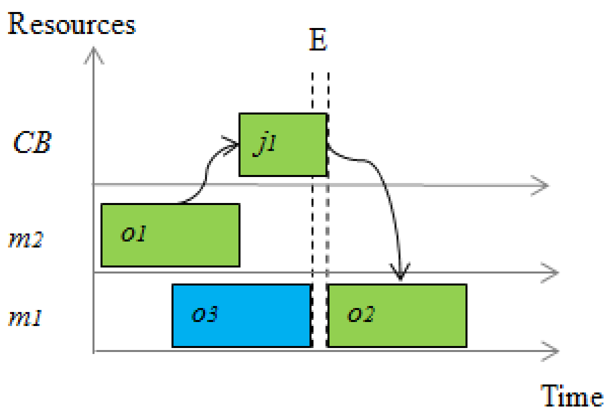


Figure 8. Effect of the Deadlock-free Constraint in Bufferless Scenario

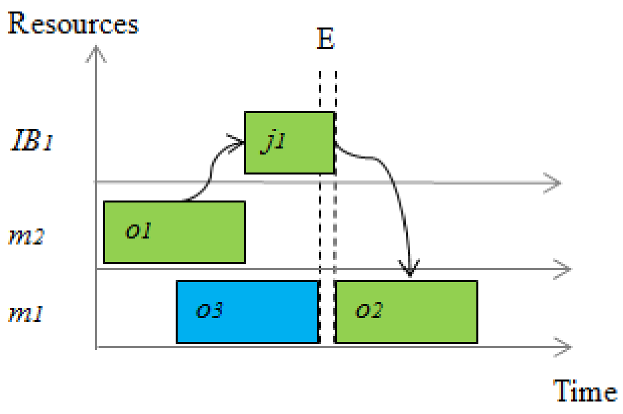
In the above figure, operations o*1* and o*2* are two consecutive processing intervals of the same job*j1*. *o1* is processed in machine *m2*. *o2* is processed in machine *m1*.Operation *o3* is the operation previous to *o2* in the sequence of *m1*. E is a small unit of time. The deadlock expression forces *o3* to end before *o1* by E.

To express the deadlock-freeness in formulations with limited capacity buffers, note that buffering is constructed as interval of times in the CPO models. Each buffering interval precedes a processing interval even if the size of the buffering interval is zero. Consequently, the deadlock-freeness constraint is expressed. For two consecutive processing intervals of the same job, the end time of the buffering interval must be greater than the end of the previous processing interval of another job proceeding onto the next resource by a small unit of time.

Illustration of the effect of the deadlock constraints in manufacturing cells with buffers is shown in Figure 9.



(a)



(b)

Figure 9. Effect of the Deadlock-free Constraint : (a) Manufacturing Cells with Central Buffer (b) Manufacturing Cell with Input Buffers

In the above figure, operations o*1* and o*2* are two consecutive processing intervals of the same job*j1*. *o1* is processed in machine *m2*. *o2* is processed in machine *m1*.Operation *o3* is the previous operation to *o2* in the sequence of *m1*. E is a small unit of time. Figure 9 (a) represents the central buffer case. *j1* uses the central buffer *CB* between *o1 and o2*. The deadlock expression forces *o3* to end before the buffering interval of *j1* in *CB* by E. Figure 9 (b) represents the input buffer case. *j1* uses the input buffer *IB1* associated to *m1* before starting *o2*. The deadlock expression forces *o3* to end before the buffering interval of *j1* in *IB1*by E.

The deadlock-freeness in bufferless formulation considers machines as the only constraining resource. The deadlock-freeness in formulations with limited capacity buffers considers machines and buffers as constraining resources. The approach used to express the deadlock-freeness in formulations with limited capacity buffers is different from the deadlock-freeness expression in the bufferless case. However, in the two deadlock-freeness expressions, the concept of end time of an interval, and end time of a previous interval in a sequence are utilized.

Therefore, the *endOf* and *endOfPrev* functions are combined into one inequality in the proposed deadlock-free expressions.

To conclude, the models described in this section need to consider a few more issues in order to represent a more complete model.

- The unlimited buffer case does not require any deadlock-free constraint since the buffer availability causes the deadlock situation not to occur in this case. Similarly, the unlimited buffer case does not require any modeling for the buffers since the buffer availability is unlimited. Simply put, there is no limit that has to be set for the buffer capacities. Therefore, jobs' routing and machines' disjunction are sufficient to constrain the unlimited buffer model.
- The formulation with dual-gripper robot does not require any deadlock-free constraint since the implementation of dual-gripper robot ensures the deadlock-freeness. In case of a circular wait, the dual-gripper robot intervenes to solve it, and the manufacturing cell proceeds to its processing activities normally. The dual-gripper robot has two grippers where two parts can be placed simultaneously. This feature can be used for deadlock resolution. The methodology is to make the deadlock situation permissive at the processing stage and use the dual-gripper robot to swap jobs between the resources to ensure the deadlock resolution. Figure 10 illustrates the effect of the robot gripper type on the deadlock resolution. *r1* and *r2* are two resources where activities are sequenced. Activities *(o1, o2)* of a job *j1* are staged to happen consecutively in *r2* and *r1*. Another pair of activities *(o3, o4)* of another job *j2* are staged to happen consecutively in *r1* and *r2*. Job *j1* is moving from *r2*to*r1*and job *j2* is moving from *r1* to *r2*. Further, the activity *o4* follows activity *o1* on *r2* and activity *o2* follows activity *o3* on

r1. Clearly, Figure 10 is an unfeasible schedule when a single-gripper robot is used. At time *t*, a circular wait is created between *r1* and *r2*. The interval of time of *o1* and *o3*had ended. The following activities *o2* and *o4*cannot start. In contrast, with a dual-gripper robot, Figure 10 is a feasible schedule. At time *t*, the robot can load job *j2* from *r1*, move to *r2*, switch the position of its grippers and load job *j1* from *r2*, switch the position of its grippers and unload part *j2* at *r2*, and then move part *j1* to *r1*. Therefore, the robot takes advantage of having two grippers to swap jobs *j1* and *j2* between *r1* and *r2*. This swapping will allow the complete execution of the manufacturing schedule.

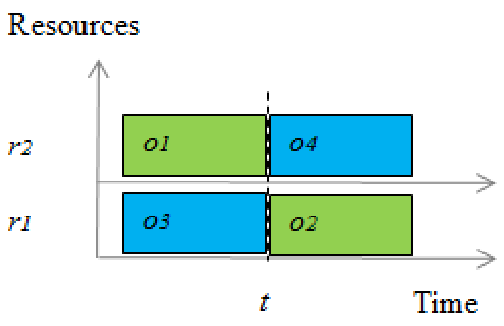


Figure 10. Deadlock State in Manufacturing Schedule

- In the formulation that had considered the limited buffer capacities, the central buffer is associated with the whole system while the input buffers are associated with individual machines. The central buffer and input buffers are represented differently. The central buffer is modeled by one cumulative function. The cumulative function representing an input buffer is indexed by the machine number to which the input buffer is associated.

- In the declarations, the size of the processing interval of a given operation is at least the processing time on a machine. This means that a job can remain in a machine even when the processing is completed. The processing interval ends when the job moves to the next resource.

- The declaration of jobs' ordering and processing times used two tuples. The first tuple declares the order of the operations of the jobs. The tuple uses tree identifiers; {operation, job, order}. The second tuple declares the processing time of jobs' operations. The tuple uses three identifiers; {operation, machine, processing time}. The first identifier is common between the two tuples to identify the same job. This data construct makes the data manipulation easier in the models.

3.5. Complete Models

3.5.1. Nomenclature

The following notations define the sets, parameters, and variables considered in the developed models. The following notations are similar to the notations used previously in the example of the job shop problem presented in IBM Corp (2011).

Sets:

Jobs	set of all Jobs {1, 2... n},
Mchs	set of all machines {1, 2… m},
Ops	set of fixed intervals of all ops that represent processing activities, and
Modes	set of fixed interval alternatives to *Ops.*

Parameters:

nbJobs	number of jobs,

nbMchs	number of machines,
Pt	processing time of an operation,
Pos	position of operations of a job,
b	capacity of a buffer, and
E	very small unit of time.

Variables:

ops	fixed interval representing processing activities with size of at least *Pt*,
modes	fixed interval representing processing alternative *ops*,
Buffer	fixed interval representing storage activities,
mchs	sequence variable of *ops*, and
bufferUsage	cumulative function of the storage activities in the buffer.

Model 1: Manufacturing cell unlimited capacity buffers (UB)

```
usingCP;
intnbJobs= ...;
intnbMchs= ...;
rangeJobs=1..nbJobs;
rangeMchs=1..nbMchs;
tupleOperation{intid;intjobId;intpos;};
tupleMode{intopId;intmch;intpt;};
{Operation}Ops= ...;
{Mode}Modes= ...;
dvarintervalops[Ops];
dvarintervalmodes[mdinModes]optionalsizemd.pt..1000;
dvarsequencemchs[minMchs]inall(oinOps,mdinModes:md.mch==m&&md.opId==o.id)modes[md];
execute{cp.param.TimeLimit=2000;}
minimizesum(oinOps:o.pos==nbMchs)endOf(ops[o])/(nbJobs);
subjectto{
forall(jinJobs,o1inOps,o2inOps:o1.jobId==j&&o2.jobId==j&&o2.pos==1+o1.pos)
endBeforeStart(ops[o1],ops[o2]);
forall(minMchs)
noOverlap(mchs[m]);
}
```

The instruction at line 1 specifies that the code is CP model. Line 2 reads the number of jobs *nbJobs* from the problem data file. Line 3 reads the number of machines *nbMchs* from the data file. Line 4 sets the range of the jobs from one to the number of jobs *nbJobs* read in line 2. Line 5 sets the range of the machine from one to the number of machines *nbMchs* read in line 3. Line 6 is a tuple declaration of the processing operations of the jobs. Line 7 is a tuple declaring the processing time of the jobs' operations. Line 8 is a triplet that reads the data for the tuple line 6. Line 9 is a triplet that reads the data for the tuple line 7. Line 10 declares an array of interval variables representing the operations. The interval variable is indexed by the set of operation. Line 11 declares another array of interval variable representing the operations. The interval variable is indexed by the set of operations. This interval is given an optional size to be at least the processing time.Line 12 declares a sequence variable on the set of all machine. Each sequence variable contains the interval variables representing the operations that are processed in the same machine. Line 13 limits the search time to 2000 sec. Line 14 is the objective function for minimizing the MFT. Line 16-17 is the jobs' routing. Line 19 is the machine disjunction.

Model 2: Manufacturing cell with no buffer and single-gripper robot (SGNB)

```
1.  usingCP;
2.  intnbJobs= ...;
3.  intnbMchs= ...;
4.  rangeJobs=1..nbJobs;
5.  rangeMchs=1..nbMchs;
6.  tupleOperation{intid;intjobId;intpos;};
7.  tupleMode{intopId;intmch;intpt;};
8.  {Operation}Ops= ...;
9.  {Mode}Modes= ...;
10. dvarintervalops[Ops];
11. dvarintervalmodes[mdinModes]optionalsizemd.pt..1000;
12. dvarsequencemchs[minMchs]inall(oinOps,mdinModes:md.mch==m&&md.opId==o.id)mod
    es[md];
13. execute{cp.param.TimeLimit=2000;}
14. minimizesum(oinOps:o.pos==nbMchs)endOf(ops[o])/(nbJobs);
15. subjectto{
```

```
16. forall(jinJobs,o1inOps,o2inOps:o1.jobId==j&&o2.jobId==j&&o2.pos==1+o1.pos)
17. endAtStart(ops[o1],ops[o2]);
18. forall(minMchs)
19. noOverlap(mchs[m]);
20. forall(jinJobs,o1inOps,o2inOps,md2inModes,m1inMchs,md1inModes:o1.jobId==j&&o2.job
    Id==j&&o2.pos==1+o1.pos&&md2.mch==m1&&md2.opId==o2.id&&md1.opId==o1.id)
21. endOfPrev(mchs[m1],modes[md2],0)+1<=endOf(ops[o1]);
22. }
```

Lines 1-19 are similar to the previous model except that the jobs routing in line 17 uses the *endAtStart* function. Line 20-21 is the deadlock-free constraint.

Model 3: Manufacturing cell with central buffer and single-gripper robot (SGCB)

```
1.  usingCP;
2.  intnbJobs= ...;
3.  intnbMchs= ...;
4.  rangeJobs=1..nbJobs;
5.  rangeMchs=1..nbMchs;
6.  tupleOperation{intid;intjobId;intpos;};
7.  tupleMode{intopId;intmch;intpt;};
8.  {Operation}Ops= ...;
9.  {Mode}Modes= ...;
10. dvarintervalops[Ops];
11. dvarintervalmodes[mdinModes]optionalsizemd.pt..1000;
12. dvarsequencemchs[minMchs]inall(oinOps,mdinModes:md.mch==m&&md.opId==o.id)mod
    es[md];
13. dvarintervalbuffer[oinOps] ;
14. cumulFunctionbufferUsage=sum(oinOps)pulse(buffer[o],1);
15. execute{cp.param.TimeLimit=2000;}
16. minimizesum(oinOps:o.pos==nbMchs)endOf(ops[o])/(nbJobs);
17. subjectto{
18. forall(jinJobs,o1inOps,o2inOps:o1.jobId==j&&o2.jobId==j&&o2.pos==1+o1.pos)
19. endBeforeStart (ops[o1],ops[o2]);
20. forall(minMchs)
21. noOverlap(mchs[m]);
22. forall(jinJobs,o1inOps,o2inOps:o1.jobId==j&&o2.jobId==j&&o2.pos==1+o1.pos) {
23. startAtEnd(buffer[o2],ops[o1]);
24. endAtStart(buffer[o2],ops[o2]);};
25. forall(jinJobs,o1inOps,o2inOps,md2inModes,m1inMchs,md1inModes:o1.jobId==j&&o2.job
    Id==j&&o2.pos==1+o1.pos&&md2.mch==m1&&md2.opId==o2.id&&md1.opId==o1.id)
26. endOfPrev(mchs[m1],modes[md2],0)+1<=endOf(buffer[o2]);
27. bufferUsage<=b;
28. }
```

Lines 1-12 are similar to the previous model. Line 13 is the interval variable of the buffering activities. Line 14 declares the cumulative function for the central buffer. Line 15 limits the search time to 2000 sec. Line 16 is the objective function for minimizing the MFT. Lines 18-19 represent the jobs' routing. Lines 20-21 are the machine disjunction. Lines 22-24 forces the end and the start of the buffering interval. Line 25-26 is the deadlock-free constraint. Line 27 sets the limit for the cumulative function value to model the central buffer capacity.

Model 4: Manufacturing cell with input buffer and single-gripper robot (SGIB)

```
1.  usingCP;
2.  intnbJobs= ...;
3.  intnbMchs= ...;
4.  rangeJobs=1..nbJobs;
5.  rangeMchs=1..nbMchs;
6.  tupleOperation{intid;intjobId;intpos;};
7.  tupleMode{intopId;intmch;intpt;};
8.  {Operation}Ops= ...;
9.  {Mode}Modes= ...;
10. dvarintervalops[Ops];
11. dvarintervalmodes[mdinModes]optionalsizemd.pt..1000;
12. dvarsequencemchs[minMchs]inall(oinOps,mdinModes:md.mch==m&&md.opId==o.id)mod
    es[md];
13. dvarintervalbuffer[oinOps] ;
14. cumulFunctionbufferUsage[minMchs]
    =sum(oinOps,mdinModes:md.mch==m&&md.opId==o.id)pulse(buffer[o],1);
15. execute{cp.param.TimeLimit=2000;}
16. minimizesum(oinOps:o.pos==nbMchs)endOf(ops[o])/(nbJobs);
17. subjectto{
18. forall(jinJobs,o1inOps,o2inOps:o1.jobId==j&&o2.jobId==j&&o2.pos==1+o1.pos)
19. endBeforeStart (ops[o1],ops[o2]);
20. forall(minMchs)
21. noOverlap(mchs[m]);
22. forall(jinJobs,o1inOps,o2inOps:o1.jobId==j&&o2.jobId==j&&o2.pos==1+o1.pos) {
23. startAtEnd(buffer[o2],ops[o1]);
24. endAtStart(buffer[o2],ops[o2]);};
25. forall(jinJobs,o1inOps,o2inOps,md2inModes,m1inMchs,md1inModes:o1.jobId==j&&o2.job
    Id==j&&o2.pos==1+o1.pos&&md2.mch==m1&&md2.opId==o2.id&&md1.opId==o1.id)
26. endOfPrev(mchs[m1],modes[md2],0)+1<=endOf(buffer[o2]);
27. forall(minMchs)
28. bufferUsage[m] <=1;
29. }
```

All codes in this model are similar to the previous model except that the cumulative functions in line 14 and line 28 are indexed by the machine number to model the input buffers.

Model 5: Manufacturing cell with no buffer and dual-gripper robot (DGNB)

```
1.  usingCP;
2.  intnbJobs= ...;
3.  intnbMchs= ...;
4.  rangeJobs=1..nbJobs;
5.  rangeMchs=1..nbMchs;
6.  tupleOperation{intid;intjobId;intpos;};
7.  tupleMode{intopId;intmch;intpt;};
8.  {Operation}Ops= ...;
9.  {Mode}Modes= ...;
10. dvarintervalops[Ops];
11. dvarintervalmodes[mdinModes]optionalsizemd.pt..1000;
12. dvarsequencemchs[minMchs]inall(oinOps,mdinModes:md.mch==m&&md.opId==o.id)modes[md];
13. execute{cp.param.TimeLimit=2000;}
14. minimizesum(oinOps:o.pos==nbMchs)endOf(ops[o])/(nbJobs);
15. subjectto{
16. forall(jinJobs,o1inOps,o2inOps:o1.jobId==j&&o2.jobId==j&&o2.pos==1+o1.pos)
17. endAtStart(ops[o1],ops[o2]);
18. forall(minMchs)
19. noOverlap(mchs[m]);
20. }
```

This model is similar to the Model 2 except that the deadlock-free constraint is omitted.

Model 6: Manufacturing cell with central buffer and dual-gripper robot (DGCB)

```
30. usingCP;
31. intnbJobs= ...;
32. intnbMchs= ...;
33. rangeJobs=1..nbJobs;
34. rangeMchs=1..nbMchs;
35. tupleOperation{intid;intjobId;intpos;};
36. tupleMode{intopId;intmch;intpt;};
```

```
{Operation}Ops= ...;
{Mode}Modes= ...;
dvarintervalops[Ops];
dvarintervalmodes[mdinModes]optionalsizemd.pt..1000;
dvarsequencemchs[minMchs]inall(oinOps,mdinModes:md.mch==m&&md.opId==o.id)modes[md];
dvarintervalbuffer[oinOps] ;
cumulFunctionbufferUsage=sum(oinOps)pulse(buffer[o],1);
execute{cp.param.TimeLimit=2000;}
minimizesum(oinOps:o.pos==nbMchs)endOf(ops[o])/(nbJobs);
subjectto{
forall(jinJobs,o1inOps,o2inOps:o1.jobId==j&&o2.jobId==j&&o2.pos==1+o1.pos)
endBeforeStart (ops[o1],ops[o2]);
forall(minMchs)
noOverlap(mchs[m]);
forall(jinJobs,o1inOps,o2inOps:o1.jobId==j&&o2.jobId==j&&o2.pos==1+o1.pos) {
startAtEnd(buffer[o2],ops[o1]);
endAtStart(buffer[o2],ops[o2]);};
bufferUsage<=b;
}
```

This model is similar to the Model 3 except that the deadlock-free constraint is omitted.

Model 7: Manufacturing cell with input buffer and dual-gripper robot (DGIB)

```
usingCP;
intnbJobs= ...;
intnbMchs= ...;
rangeJobs=1..nbJobs;
rangeMchs=1..nbMchs;
tupleOperation{intid;intjobId;intpos;};
tupleMode{intopId;intmch;intpt;};
{Operation}Ops= ...;
{Mode}Modes= ...;
dvarintervalops[Ops];
dvarintervalmodes[mdinModes]optionalsizemd.pt..1000;
dvarsequencemchs[minMchs]inall(oinOps,mdinModes:md.mch==m&&md.opId==o.id)modes[md];
dvarintervalbuffer[oinOps] ;
cumulFunctionbufferUsage[minMchs]
=sum(oinOps,mdinModes:md.mch==m&&md.opId==o.id)pulse(buffer[o],1);
execute{cp.param.TimeLimit=2000;}
minimizesum(oinOps:o.pos==nbMchs)endOf(ops[o])/(nbJobs);
subjectto{
```

```
forall(jinJobs,o1inOps,o2inOps:o1.jobId==j&&o2.jobId==j&&o2.pos==1+o1.pos)
endBeforeStart (ops[o1],ops[o2]);
forall(minMchs)
noOverlap(mchs[m]);
forall(jinJobs,o1inOps,o2inOps:o1.jobId==j&&o2.jobId==j&&o2.pos==1+o1.pos) {
startAtEnd(buffer[o2],ops[o1]);
endAtStart(buffer[o2],ops[o2]);};
forall(jinJobs,o1inOps,o2inOps,md2inModes,m1inMchs,md1inModes:o1.jobId==j&&o2.jobId==j&&o2.pos==1+o1.pos&&md2.mch==m1&&md2.opId==o2.id&&md1.opId==o1.id)
endOfPrev(mchs[m1],modes[md2],0)+1<=endOf(buffer[o2]);
forall(minMchs)
bufferUsage[m] <=1;
}
```

This model is similar to the Model 4 except that the deadlock-free constraint is omitted.

3.6. Summary

The IBM ILOG CP Optimizer (CPO) language is successfully used to the model the different scenarios studied in the thesis. The built-in scheduling functions along with the concept of intervals of CPO make the modeling more flexible. The modeled scenarios include the following: manufacturing cell with unlimited buffer; with no buffer and a single-gripper robot; with no buffer and a dual-gripper robot; with central buffer and a single-gripper robot; with central buffer and dual-gripper robot; with input buffers and dual-gripper robot; and with input buffers and a dual-gripper robot. The level of complexity of the modeling differs among the different scenarios. The unlimited buffer case was the simplest model; only jobs' routing and machine disjunction constraints are needed. The models with the single-gripper robot and buffer are complex to model and result in larger size models. The models with dual-gripper robot had less number of constraints than the models with single-gripper robots.

Chapter 4 VALIDATION OF THE DEVELOPED MODELS

4.1. Introduction

This section validates the solutions of the scheduling problem (JSSP and F-JSSP) for the studied scenarios of the manufacturing cells (UB, SGNB, SGCB, SGIB, DGNB, DGCB, and DGIB). The validation means examining the accuracy and feasibility of the solutions. The solutions are presented in Gantt charts where the different activities are presented in time intervals. The Gantt chart is commonly used as a validation tool for scheduling problems in manufacturing cells. The jobs' routing, machines' disjunction, buffers' capacity, and deadlock-freeness constraints are the constraining expressions in the scheduling problem. The effects of these expressions are demonstrated through the interpretation of the presence of the different intervals of activities in the schedules. The presence of the intervals of activities needs to follow the logics of the constraining expression. A small size problem is generated in order to facilitate the validation. The solutions of the JSSP with the different scenarios are examined first. Discussions for the F-JSSP with the same scenarios are presented next. A summary is given at the end of this Chapter.

4.2. Validation Test Problem

A manufacturing cell that needs to produce eight different jobs *Jj* (*J1*, *J2*.., *J8*) has three CNC machines (*M1*, *M2*, and *M3*). The jobs are mixed and require three processes each. Each process has different processing times. The processing time and jobs routes for the above manufacturing scenario are given in Table 6. Table 6 shows the processing time and jobs routes for the JSSP problem. The processing times of all the operations are different. The jobs have different routings in the system. Each machine can perform a distinct operation. Table 7 gives the processing time and jobs routes when F-JSSP is considered. Table 7 features the routing flexibility where the same operation of a job can be allocated to different machines.

Table 6. Data of the Initial Problem in JSSP

Job	Operation	Machine	Processing Time	Job	Operation	Machine	Processing Time
1	1	1	20	5	1	1	23
	2	2	30		2	2	25
	3	3	40		3	3	39
2	1	3	20	6	1	2	20
	2	2	30		2	1	35
	3	1	35		3	3	30
3	1	3	25	7	1	2	26
	2	2	32		2	3	20
	3	1	40		3	1	30
4	1	3	40	8	1	1	32
	2	1	26		2	2	26
	3	2	38		3	3	30

Table 7. Data of the Initial Problem in F-JSSP

Job	Operation	Machine	Processing Time	Job	Operation	Machine	Processing Time
1	1	1	20	5	1	1	23
	2	2	30		2	2	25
	3	3	40		3	3	39
	3	1	20		3	2	20
2	1	3	20	6	1	2	20
	2	2	30		2	1	35
	3	1	35		3	3	30
3	1	3	25		3	2	20
	2	2	32	7	1	2	26
	3	1	40		2	3	20
	3	3	20		3	1	30
4	1	3	40		3	3	40
	2	1	26	8	1	1	32
	3	2	38		2	2	26
					3	3	30

The above example is not only applicable for a manufacturing cell with machines only, but also for a manufacturing cell with an unlimited buffer, a manufacturing cell with a central buffer, and a manufacturing cell with input buffers. Figure 11 shows a basic layout of the manufacturing cell. It presents examples of different buffer configurations. The manufacturing cells can use a robot with either dual grippers or a single gripper (both gripper types are shown below in each figure). For all configurations, the raw material input device and the finished job's output device would be placed at the same location. In these scenarios, the robot picks a job from the raw material input device and then moves the job to each machine in a prescribed order. The processing of a job in the machine starts right after the robot loads it into a machine. The robot unloads the job from a machine when its processing is done and moves the jobs among the machines according to the process plan. The robot moves the jobs into buffers if available and needed. After the last operation of a job is completed in a machine, the robot unloads the part and takes it to the finished job's output device.

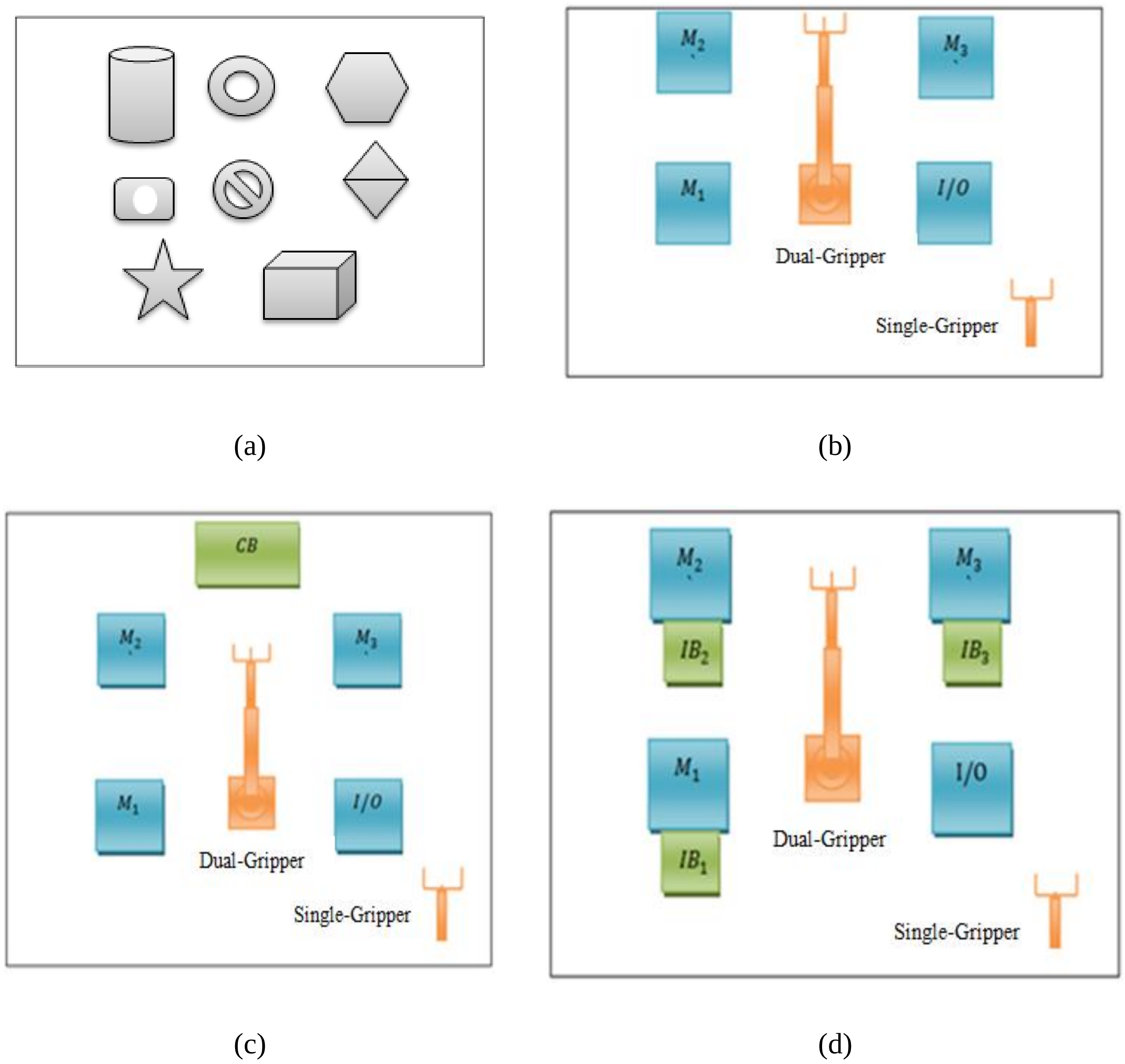


Figure 11. Illustration of the Test Problem: (a) Set of jobs of the case study, (b) A three-machine FMC with no buffers, (c) A three -machine FMC with a central buffer, (d) A three-machine FMC with input machine buffers

4.3. Solving the Test Problem with CPO

To solve the scheduling problems, the formulations of the models are programmed in the IBM ILOG CPLEX CP Optimizer with CPLEX CP Optimizer version 12.3 and tested by means of PC Intel(R) Core(TM) i7-2600 CPU @ 3.40GHz with 8 GB RAM under Windows 7 operating system. The CPO *cp.param.TimeLimit* parameter is set at 2000 sec which is consistent with those assumed in previous literature such as Zeballos (2010).

4.4. Examining the Solution of the Test Problem

When solving the models, best objectives are found when the search is terminated upon reaching a time limit. The derived Gantt charts of the solution are given in Figures 12-25. The sequence of the jobs in the machines is aligned with the constraining resources and assumptions of the models. In general, the objective values found for the models vary as each configuration performs differently. The performance of the configuration depends on the availability of the resources. The performance of the configuration will be presented in the next chapter. Each chart of the JSSP and F-JSSP is tracked to consolidate the validity of the models (UB, SGNB, SGCB, SGIB, DGNB, DGCB, and DGIB) and analyzed below.

4.4.1. Validation of the JSSP

The solution obtained of the UB model is shown in Figure 12. The best objective is found to be 169.4 sec. Deadlocks are not an issue for this model because of the infinite capacity assumed for buffers.

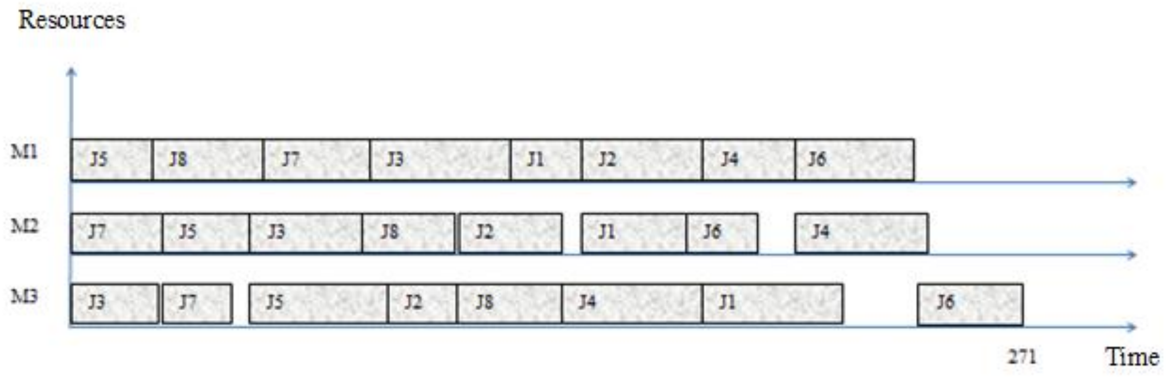


Figure 12. Solution Obtained for JSSP||UB

The solution obtained of the SGNB model is shown in Figure 13. The best objective is found to be 213 sec. The resulting SGNB chart is a deadlock-free schedule even though there are no additional resources other than the machines that could be used for the deadlock resolution. Therefore, the only solution for deadlock resolution is shifting the jobs' order in the machines. Clearly, each job keeps to its defined routing in the system with respect to the deadlock-free constraint and no job leaves a machine until the following machine is available. When a job is completed on a machine, the next machine in its route is already free. For example, J7 visits M2, M3, and M1 respectively. When J7 finishes it processing in M2, M3 is available to start the second processing of J7, and then when J7 finishes it processing in M3, M1 is available to start the last processing of J7. Thus, J7 always finds the following machine to be available. All the jobs, in this SGNB chart, are tracked to validate the deadlock-freeness.

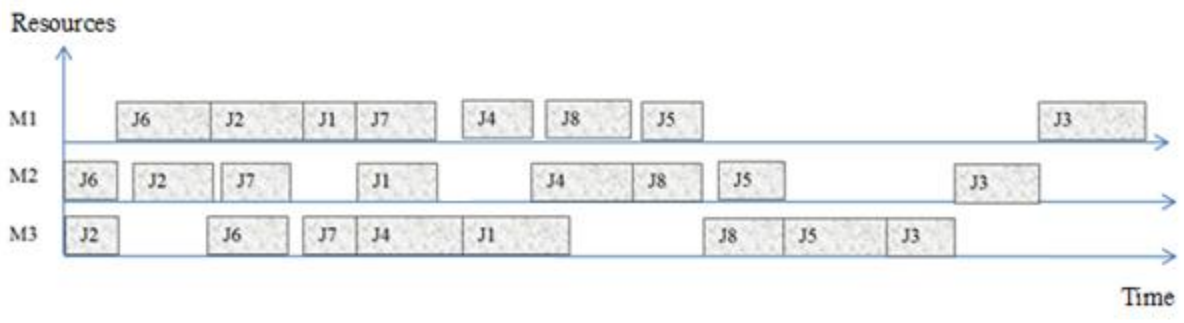


Figure 13. Solution Obtained for JSSP||SGNB

The solution obtained for the SGCB model is shown in Figure 14. The best objective is found to be 179.88 sec. The schedule in the SGCB chart is deadlock-free. When a job is completed on machine, the next machine in its route is already free or the central buffer is available. Similarly, a job is not released from the central buffer until the next machine in its route is already free. For example, from the SGCB chart, J3 visits M3, M2, and M1 respectively. J3 uses the central buffer between the processing on M2 and M1. J3 finds the next available resource. All the end times and start times of the jobs, in this SGCB chart, are tracked to validate the deadlock-freeness. Furthermore, with one unit capacity, the central buffer is used six times by J2, J1, J4, J6, J3, and J5 at times 20, 96, 128, 148, 193 and 216 respectively. Clearly, the use of the central buffer several times as seen in the chart shows the benefits derived from the presence of a central buffer in manufacturing cells as an additional resource. The central buffer had significant effect on the deadlock resolution. At time 148, for example, the central buffer resolves the circular wait between M1 and M2 caused by J4 and J6. This deadlock could not have been solved without the availability of the central buffer. The single gripper robot in this case would not have been able to resolve the deadlock. Without the use of the central buffer, this schedule would be identical to the schedule depicted in Figure 16 which shows the presence of deadlocks.

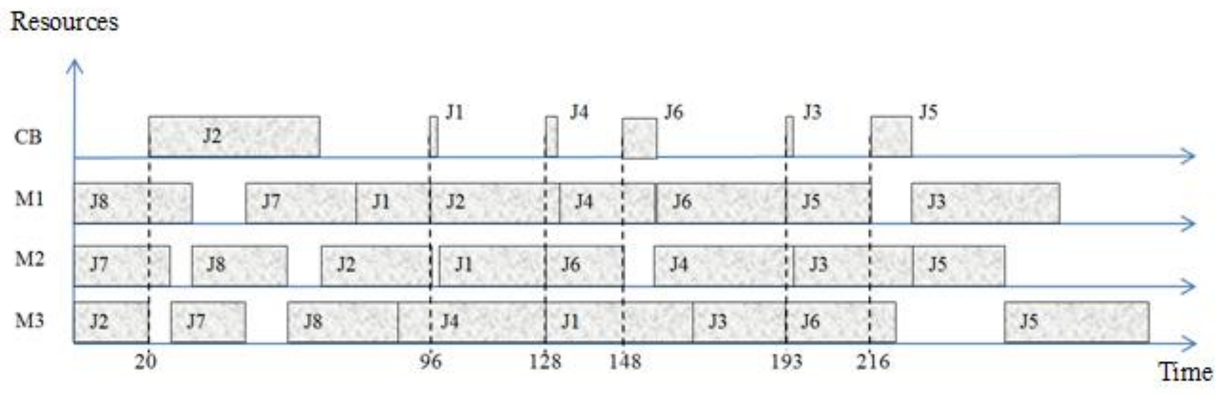


Figure 14. Solution Obtained for JSSP||SGCB

The solution obtained for the SGIB model is shown in Figure 15. The best objective is found to be 169.37 sec. The schedule is deadlock-free. The deadlock expression has its effect on the chart. When a job is completed on resource (machine or input buffer), the next resource in its route is available. Clearly, the use of the central buffer several times shows the benefits from the presence of input buffers attached to each machine as an additional resource. The input buffer IB1 of M1 is used four times at 51, 83, 140, 180, and 195 for jobs J7, J3, J2, J4, and J6 respectively. The input buffer IB2 of M2 is used three times at 23, 26, and 55 for jobs J5, J3, and J8 respectively. The input buffer IB3 of M3 is used three times at 26, 109, and 175 for jobs J7, J8, and J1 respectively. Deadlocks would have resulted if the input buffers were not present. The single gripper robot in this case would not have been able to resolve the deadlock. Without the use of the input buffers, this schedule would be identical to the schedule depicted in Figure 16 which shows the presence of deadlocks.

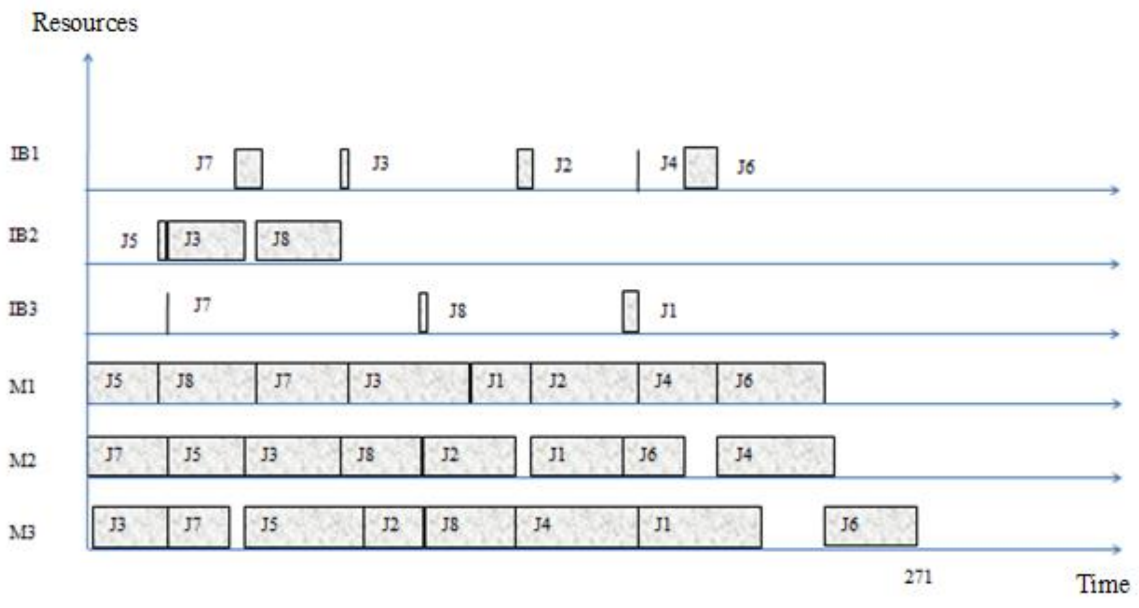


Figure 15. Solution Obtained for JSSP||SGIB

The solution obtained for the DGNB model is shown in Figure 16. The best objective is found to be 183.62 sec. In the resulting chart, the completion time of a job is not constrained by the completion time of any other job. Without buffers in the manufacturing cell, the two grippers of the robot play a crucial role when circular waits occur. The dual-gripper robot swaps the jobs between the machines to resolve deadlocks. In the chart, the two grippers of the robot are used five times to avoid the deadlock situation (66, 99, 124, 156, and 229). All the circular waits are avoided between the two machines. For example, J8 and J2 complete their processing in M2 and M3 at the same time (66 sec) even though J2 is moving to M2 and J8 is moving to M3. This could cause a circular wait between machines M2 and M3 if the dual-gripper robot is not used. However, the dual-gripper robot swaps J8 and J2 between M1 and M2, and resolves the circular wait.

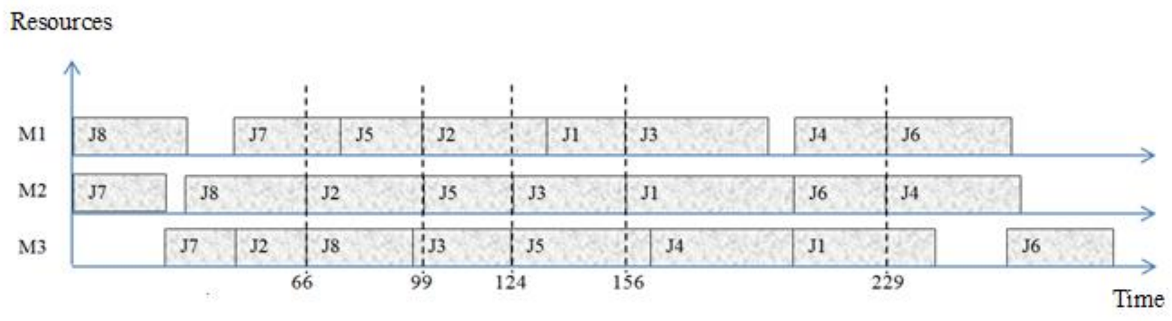


Figure 16. Solution Obtained for JSSP||DGNB

The solution obtained of the DGCB model is shown in Figure 17. The best objective is found to be 169.37 sec. With a capacity of one unit, the central buffer is used nine times for this test problem. Jobs J5, J3, J7, J8, J3, J8, J2, J1, and J6 use the central buffer at times 23, 26, 51, 55, 83, 109, 140, 175, and 195 respectively. The deadlock occurs when the central buffer reaches its maximum capacity. For example, at time 83, the two grippers of the robot are used once to avoid a deadlock situation between the central buffer and machine M2. J8 is leaving the central buffer to move to M2. J3 is leaving M2 and entering the central buffer. The dual-gripper robot swaps J3 and J8 between M2 and the central buffer and resolves the deadlock.

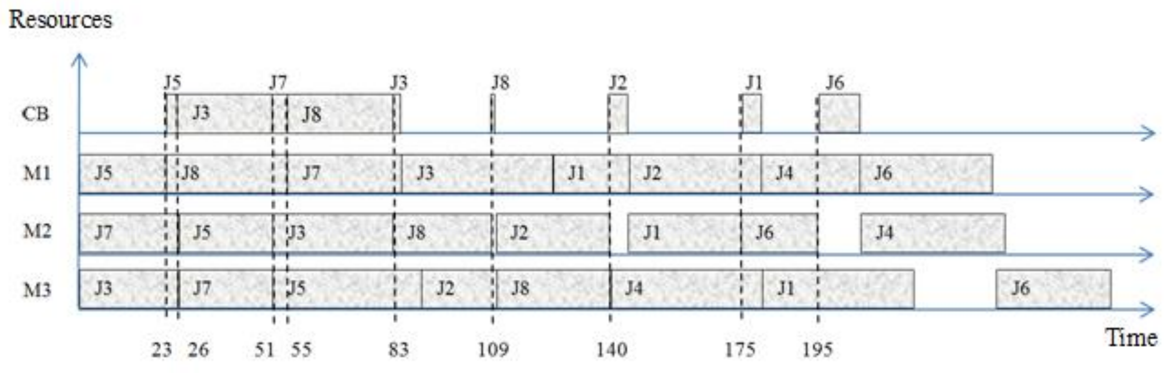


Figure 17. Solution Obtained for JSSP||DGCB

The solution obtained for the DGIB model is shown in Figure 18. The best objective is found to be 169.37 sec. The input buffer IB1 of M1 is used four times at 51, 83, 140, and 195 for jobs J7, J3, J2, and J6 respectively. The input buffer IB2 of M2 is used at times 23, 26, and 55 for jobs J5, J3, and J8 respectively. The input buffer IB3 of M3 is used at times109, and 175 for jobs J8 and J1 respectively. The schedule is deadlock-free. This is confirmed by tracking the start times and ends time of the processing and buffering activities in the chart. This problem uses input buffers and a dual-gripper robot which are both effective in resolving deadlocks. However, the potential of the dual-gripper robot is not utilized in this schedule since the buffer capacities are large enough to deal with deadlocks. The potential of the dual-gripper robot may be seen more clearly with large size problems.

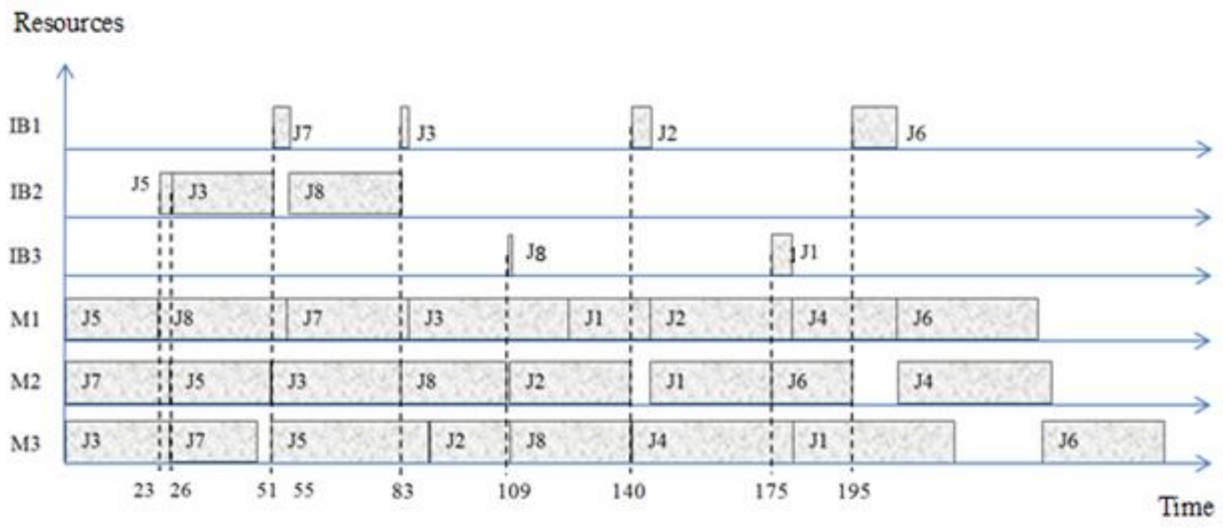


Figure 18. Solution Obtained for JSSP||DGIB

4.4.2. Validation of the F-JSSP

Following the same approach used for the JSSP validation, all the charts UB, SGNB, SGCB, SGIB, DGNB, DGCB, and DGIB are validated to be deadlock-free for the F-JSSP. Interestingly, the routing flexibility has a special effect on the deadlock resolution in the manufacturing cells. The jobs use the alternative routes to avoid the deadlocks in the manufacturing cells. The results of the F-JSSP are as follows:

The solution obtained of the UB model is shown in Figure 19. The best objective is found to be 158.12 sec. Deadlocks are not an issue for this model because of the infinite capacity assumed for buffers.

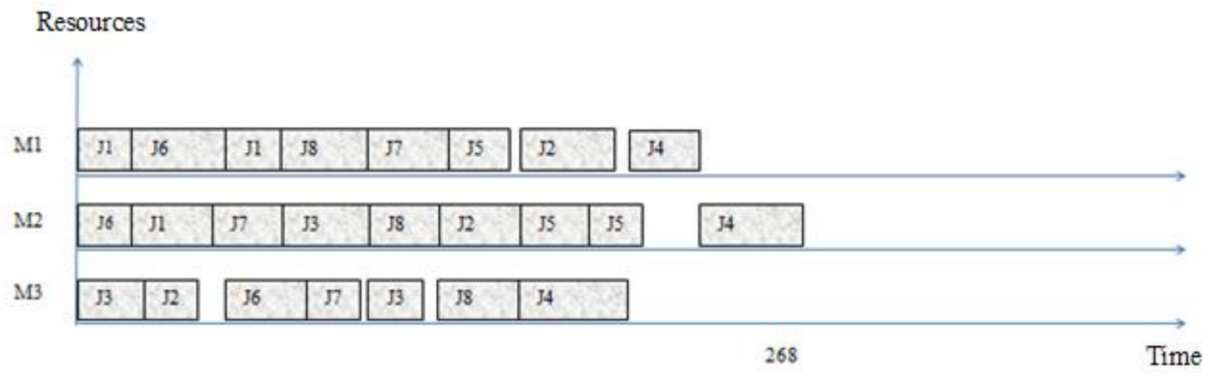


Figure 19. Solution Obtained for F-JSSP||UB

The solution obtained of the SGNB model is shown in Figure 20. The best objective is found to be 193 sec. The resulting SGNB chart is a deadlock-free schedule. Clearly, each job keeps to its defined routing in the system with respect to the deadlock-free constraint and no job leaves a machine until the following machine is available. When a job is completed on a machine, the next machine in its route is ready for new processing operation. For example, J7 visits M2, M3, and M1 respectively in the chart. When J7 finishes it processing in M2, M3 is available to start the second processing of J7, and then when J7 finishes its processing in M3, M1 is available to start the last processing of J7. Thus, J7 always finds the following machine to be available. All the jobs, in this SGNB chart, are tracked to validate the deadlock-freeness.

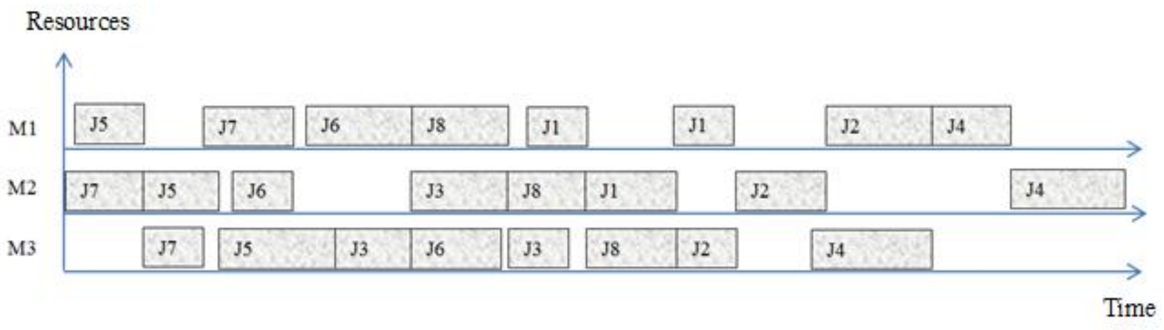


Figure 20. Solution Obtained for the F-JSSP||SGNB

The solution obtained of the SGCB model is shown in Figure 21. The best objective is found to be 164.13 sec. The schedule in the SGCB chart is deadlock-free. When a job is completed on a machine, the next machine in its route is already free or the central buffer is available. Similarly, a job is not released from the central buffer until the next machine in its route is already free. For example, in the SGCB chart, J2 visits M3, M2, and M1 respectively. J2 uses the central buffer between the processing on M2 and M1. J2 finds the next resource (machines or central buffer) needed to be available when required. All the end times and start times of the jobs, in this SGCB chart, are tracked to validate the deadlock-freeness. Furthermore, the central buffer had its effect on the deadlock resolution. Clearly, the several times that the central buffers are used in the chart shows the benefits derived from central buffer implementation in manufacturing cells as an additional resource. With one unit capacity, the central buffer is used three times. It is used at times 50, 82, and 133 for jobs J2, J3, and J5 respectively. At time 50, for example, the central buffer is used to resolve the deadlock between M1, M2, and M3 caused by J6, J2, and J3. Similarly, at time 133, CB is used to avoid a deadlock between M1, M2 and M3 caused by J1 and J5. This deadlock could not have been resolved without the availability of the central buffer. The single gripper robot in this case would not have been able to resolve the deadlock. Without the use of the central buffer, this schedule would be identical to the schedule depicted in Figure 23 which shows the presence of deadlocks.

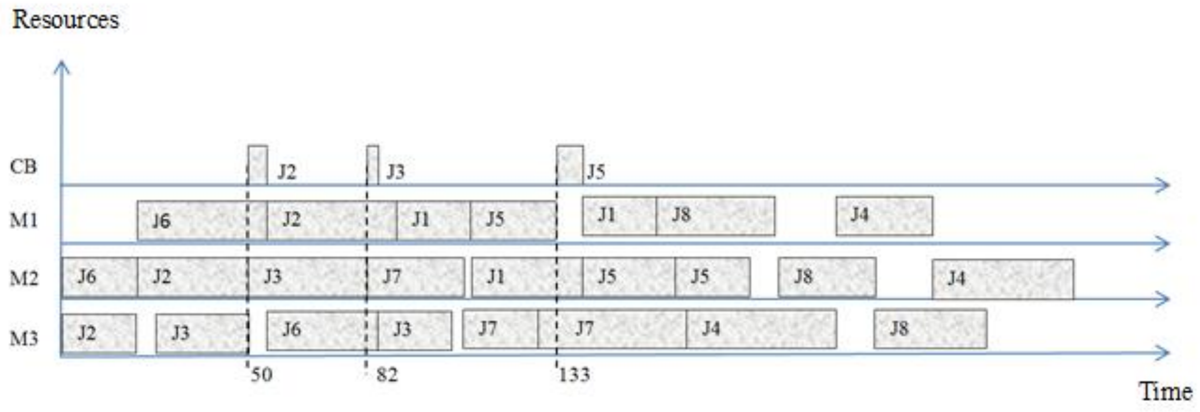


Figure 21. Solution Obtained for the F-JSSP||SGCB

The solution obtained for the SGIB model is shown in Figure 22. The best objective is found to be 158.63 sec. The schedule is deadlock-free. The deadlock expression has its effect on the chart. When a job is completed on resource (machine or input buffer), the next resource in its route is available. Clearly, the use of input buffers several times shows the benefits from input buffers attached to machines. The input buffer IB1 of M1 is used for J6, J1, J7 and J2 respectively. The input buffer IB2 of M2 is used for jobs J1, J3, J2, and J5 respectively. The input buffer IB3 of M3 is used one time for job J3. Deadlocks would have occurred if the input buffers were not present. The single gripper robot in this case would not have been able to resolve the deadlock. Without the use of the input buffers, this schedule would be identical to the schedule depicted in Figure 23 which shows the presence of deadlocks.

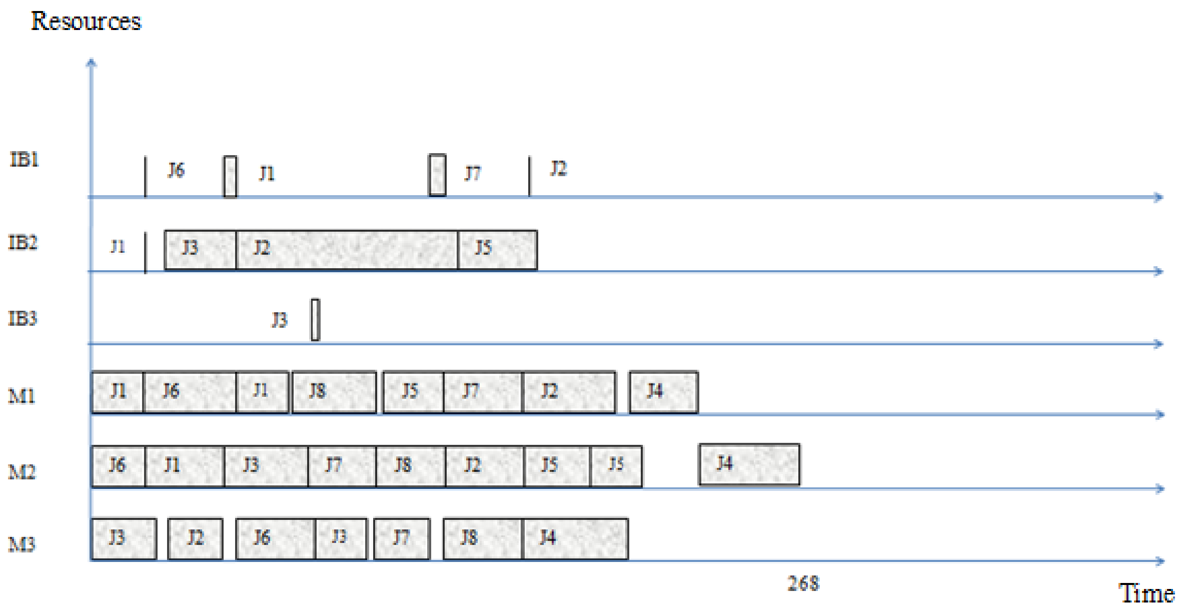


Figure 22. Solution Obtained for the F-JSSP||SGIB

The solution obtained for the DGNB model is shown in Figure 23. The best objective is found to be 166.75 sec. In the resulting chart, the completion time of a job is not constrained by the completion time of any other job even though these jobs are being interchanged in the machines. For example, J7 and J3 complete their processing in M2 and M3 at the same time (111 sec) even though J3 is moving to M2 and J7 is moving to M3. This could cause a circular wait between machines M2 and M3 if the dual-gripper robot is not used. However, the dual-gripper robot swaps J7 and J3 between M1 and M2 and resolves the circular wait. In total, the two grippers of the robot are used four times to avoid the deadlock situation. This swapping of the jobs happens at 20 sec, 55 sec,111 sec, and 143 sec. The first and third circular waits (20 sec and111sec) in the chart are avoided between two machines. The second and last circular waits (55 sec and 143 sec) in chart are avoided between three machines.

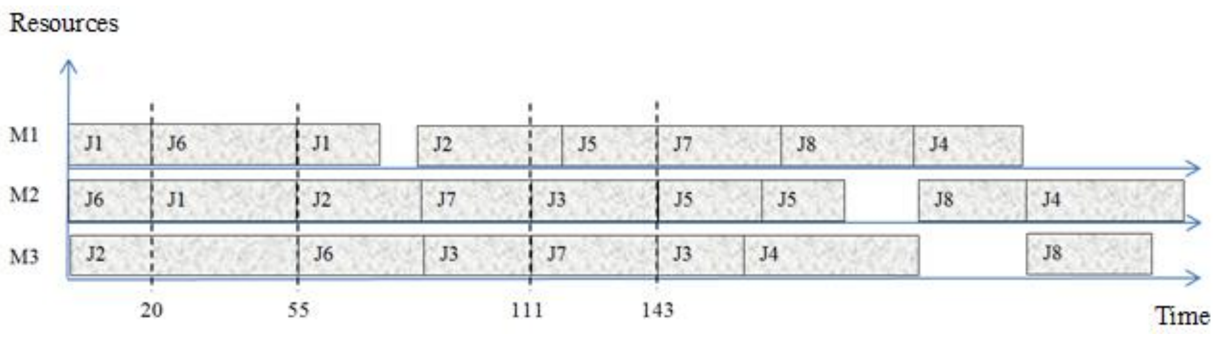


Figure 23. Solution Obtained for the F-JSSP||DGNB

The solution obtained of the DGCB model is shown in Figure 24. The best objective is found to be 160 sec. With a capacity of one, the central buffer is used ten times for this test problem. Each of the Jobs J3, J1, J2, J3, J8, J8, J4, J7, J7 and J5 use the central buffer in this order. The deadlock occurs when the central buffer reaches its maximum capacity. The two grippers of the robot are used two times to avoid the deadlock situation between the central buffer and the different machines (80 sec and 196 sec). For example, at time 80, J2 is leaving the central buffer to move to M2. J3 is leaving M2 and entering the central buffer. The dual-gripper robot swaps J3 and J2 between M2 and the central buffer. Remarkably, at time 20, a circular wait between M1 and M2 caused by J1 and J6 occurs. Even though the central buffer was available, the two grippers of the robot are used to resolve the deadlock between M1 and M2 and resolve the deadlock.

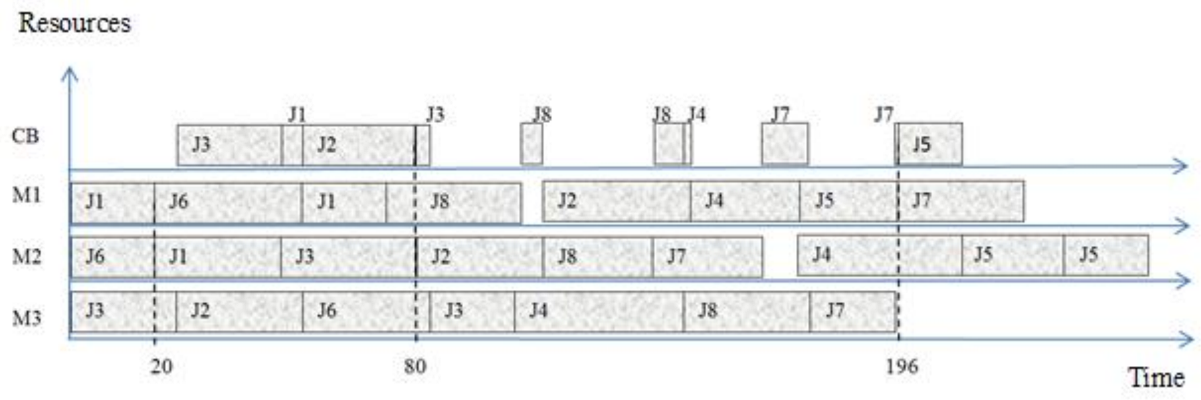


Figure 24. Solution Obtained for the F-JSSP||DGCB

The solution obtained of the DGIB model is shown in Figure 25. The best objective is found to be 158.62 sec. IB1 is used twice; at times 50 and 128 for storing jobs J1 and J7 consecutively. IB2 is used twice; at times 20 and 134 for jobs J2 and J5 consecutively. The input buffer for M3 is used for J3 at 82. The schedule is deadlock-free. This is confirmed by tracking the start times and ends time of the processing and buffering activities in the chart. This schedule uses input buffers and a dual-gripper robot which are both effective in resolving deadlocks. However, the potential of the dual-gripper robot is not utilized in this problem since the buffer capacities are large enough to deal with deadlocks. The potential of the dual-gripper robot may be more beneficial with large size problems.

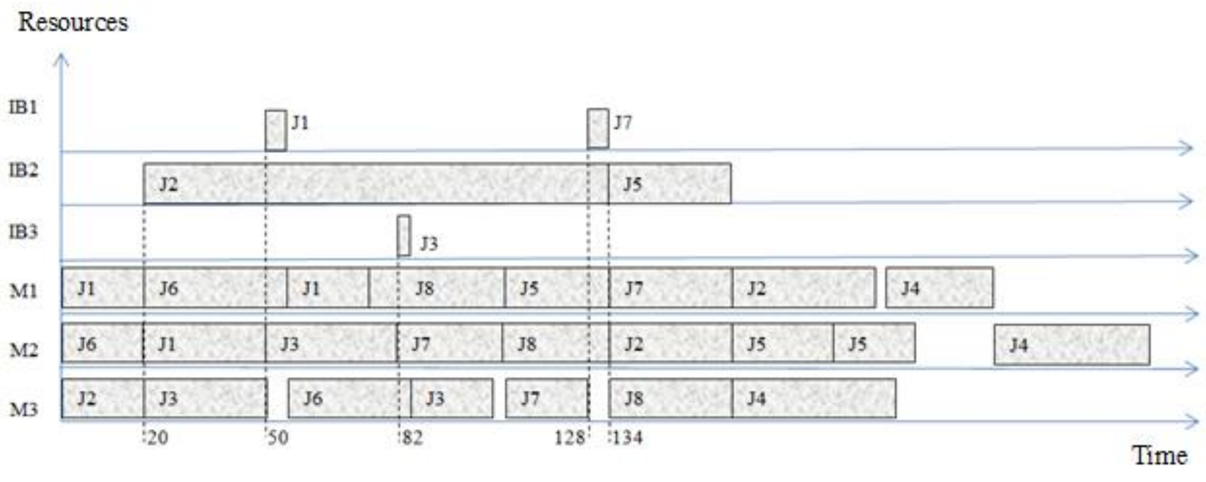


Figure 25. Solution Obtained for the F-JSSP||DGIB

The flexible routing in the F-JSSP can be noticed in the charts. Some jobs benefit from the routing flexibility in the initial data and visit a machine more than once. For example, J5 has an alternative route for the third operation to be processed in M2 or M3 (Table 7). In the DGNB (Figure 23), J5 is processed two times consecutively in M2. However, some jobs have routing flexibility in the initial data but each processing operation is performed in different machines in the resulting schedule. For example, J7 has an alternative route for the third operation to be processed in M2 or M3 (Table 7). However, in the DGIB chart (Figure 25), J7 visited all the machines (M1, M2, and M3). J7 does not use the alternative route given in the initial data for its last operation which may appear to be a better decision. Other jobs follow a unique route in the initial data. Their processing operations are performed by different machines in the manufacturing cell. This unique route is applied in the resulting schedule. For example, J8 has a unique route of M1, M2, and M3 respectively according to Table 7. This unique route is verified in the all the charts (Figures 19-25). J8 starts with M1, then goes to M2, and finishes the processing on M3.

More importantly, the routing flexibility has an effect on deadlock resolution. For example, in the SGNB, there are no additional resources other than the machines that could be used for the deadlock resolution. However, some jobs use the alternative route to avoid deadlocks in the manufacturing cell. Specifically, the route of J1 in the initial data is {M1, M2, (M3 or M1)}. The route of J2 in the initial data is {M3, M2, and M1}. In Figures 20, J1 is scheduled to go to {M1, M2, and M1}. J1 avoids visiting M3 and selects M1. J2 follows a unique route and then visits {M3, M2, M1} in the chart. The following observation can be made. J1 visits M1 twice to avoid the circular wait that would have occurred between M2 and M3 caused by J1 and J2.

4.5. Summary

This chapter presented a validation of the seven developed models for different configurations namely: *UB*, *SGNB*, *SGIB*, *SGCB*, *DGNB*, *DGIB*, and *DBCB*. The validation is done considering the JSSP and F-JSSP. The models are validated using resource allocation and Gantt charts. The processing times, jobs' routing, machines' disjunction, buffers' capacities, and deadlock-free constraints are verified by tracking the start and end times of the different processing intervals in all the charts. The deadlock-freeness received special attention. First, in the UB chart, deadlocks are not an issue for this model because of the infinite capacity assumed for buffers. Similarly, deadlocks are not an issue for charts that represent models with a dual-gripper robot (DGNB, DGCB, and DGIB) as the dual-gripper robot acted as a flexible swapping device. All the remaining models (SGNB, SGCB, and SGIB) are carefully verified to be deadlock-free. The effect of deadlock expressions on the resulting schedules is demonstrated. The flexible routing in the F-JSSP has a noticeable effect in the charts for resolving deadlocks.

The goal outlined at the beginning of this chapter has been met by ensuring that the developed model provides accurate decisions. However, a performance evaluation of the proposed manufacturing cells requires examination of different groups of problems. In Chapter 5, a comparative study between the performances of the proposed configurations will be presented.

Chapter 5 PERFORMANCE EVALUATION OF THE DUAL-GRIPPER ROBOT IN THE MANUFACTURING CELLS

5.1. Introduction

Manufacturing industries are constantly looking at ways to improve production methods in order to be competitive. They look at means to find the best fit for the resources in the manufacturing cells. Hence, many questions need to be addressed concerning which type of robots to implement and which configurations of buffers to use. The dual-gripper robot is found to result in higher productivity improvement than the single-gripper robot. However, this might not be true for all manufacturing cell configurations. In this chapter, a quantitative performance evaluation of the dual-gripper robot in the manufacturing cells is established. In Section 5.2, a comparison is made between the results of the performance of the single-gripper robot published in the literature and the performance of dual-gripper robot for the same proposed problems with the models developed in this thesis. Section 5.3 presents new groups of problems for the performance evaluation of the dual-gripper robot in the manufacturing cells. The generation of new test problem is needed to rigorously observe and measure the effect of the dual-gripper robot on the productivity improvement for different classes of problems. Section 5.4 summarizes the findings.

5.2. Comparative Study

This section compares the performance of the dual-gripper robot versus the single-gripper robot based on the different problems available in previous literature. All the available literature considers a single-gripper robot. Ramaswamy et al., (1996) propose bufferless formulations and solve a given scheduling problems of manufacturing cells with four jobs and three machines (*4J*3M)*. Fahmy et al. (2008) considered input buffers and central buffers with one unit capacity. They solve the *4J*3M*, *3J*3M*, and *5J*4M* problems. Fahmy (2009) consider intermediate buffers and a central buffer with arbitrary capacity. They solve the *6J*3M* problem. All these studies use a mathematical formulation to minimize the MFT. Fahmy et al. (2008) and Fahmy (2009) do not consider the bufferless formulation. Therefore, there are no results available for the *3J*3M*, *5J*4M*, and *6J*3M* problems in the no-buffer case. The type of cell structure considered by Ramaswamy et al. (1996) with no buffers is used to solve the *3J*3M*, *5J*4M*, and *6J*3M* problems in order to compare with the results in this study. The problems are solved using the mathematical programming component of CPLEX.

The solutions obtained for the *4J*3M*, *3J*3M*, *5J*4M*, and *6J*3M* with the no-buffer case (NB), central buffer case (CB), and input buffer case (IB) are shown in Table 8. It can be noticed that for the *4J*3M* problem, the dual-gripper robot performance is better or at least equal to the single-gripper robot performance. For the bufferless formulation, the MFT is 301.5 when using single-gripper robot and 295.25 when using a dual-gripper robot. Therefore, the dual-gripper robot improves the solution objective. Similarly, the dual-gripper robot improves the MFT in the input buffer case. The MFT is 275.25 when using a single-gripper robot and 272.75 when using a dual-gripper robot. Finally, the MFT value of 272.75 is reached when either type of robot arms are used in the central buffer case. This is because the unit capacity of the central buffer is large enough to ensure deadlock freeness. For the *3J*3M* problem, the dual-gripper robot performs better or at least equal to the single-gripper scenario. For the bufferless formulation, the MFT is 21 when using single-gripper robot and 19.67 when using a dual-gripper robot. Therefore, the dual-gripper robot improves the solution objective. Furthermore, the MFT of 18.33 is similar for all types of robot grippers when buffers are considered. The central buffers and the input buffer capacities are large enough to ensure the best solution for this problem. For the *5J*4M* problem, the dual-gripper robot performs better than the single-gripper robot in all formulations. The MFT with no-buffer is 290.4 for the single-gripper robot and 275 for the dual-gripper robot. The MFT with an input buffer is 265.40 for the single-gripper robot and 260.2 for the dual-gripper robot. The MFT with a central buffer is 262.8 for the single-gripper robot and 260.2 for the dual-gripper robot. Thus, the dual-gripper robot is very useful for this problem. Further, the central buffer and the input buffer reach the best solution with the dual-gripper robot. The *6J*3M* problem is similar to the *3J*3M* in terms of the robot performance; the dual-gripper robot improves the solution objective in the bufferless case and performs at the same level as the single-gripper robot when the capacity of the buffers is considered. However, the central buffer case requires a capacity of two until the best solution is reached. To summarize, the dual-gripper robot uses the potential of having two grippers to improve the solution objective values presented in previous literature. The best improvement is reached in the bufferless cases. When considering the capacity of the buffers, the dual-gripper robot is at least as effective as the single-gripper robot. This effectiveness is due to the availability of resources that avoid deadlock for these small problems.

Table 8. Comparison with Available Literature

Reference	Problem	Buffer	Single Gripper Robot		Dual Gripper Robot	
			Buffers' Capacity Required	Objective value	Capacity Required	Objective value
Ramaswamy et al. (1996)	4J*3M	No	0	301.5	0	295.25
Fahmy et al. (2008)	4J*3M	CB	1	272.75	1	272.75
		IB	1	275.25	1	272.75
	3J*3M	No	0	21	0	19.67
		CB	1	18.33	1	18.33
		IB	1	18.33	1	18.33
	5J*4M	No	0	290.4	0	275
		CB	1	265.40	1	260.2
		IB	1	262.8	1	260.2
Fahmy (2009)	6J*3M	No	0	163.67	1	148.67
		CB	2	140.17	2	140.17
		IB	1	140.17	1	140.17

Up to now, the comparative study is done with the existing problems in the literature. It is important to observe the effect of the dual-gripper robot with different group of problems. In the next section, more problems are generated to study the effect of dual-gripper robot.

5.3. Performance Analysis

In this section, further performance analysis of the dual-gripper robot is established to understandand the effect of the dual-gripper robot when operating in the manufacturing cells is considered. Therefore, various classes of test problems are generated. Table 9 represents the test problems considered in the performance analysis. The problems are grouped into two classes.

The first group {P1, P2, P3, P4, P5} considers 5 machines that are processing 3, 4, 5, 6, and 7 jobs respectively. The second group {P6, P7, P8, P9, P10} considers 3 machines that are processing 4, 5, 6, 7 and 8 jobs respectively.

Table 9. Test Problems

PROBLEMS	NUMBER OF PARTS	NUMBER OF MACHINES
P1	3	5
P2	4	5
P3	5	5
P4	6	5
P5	7	5
P6	4	3
P7	5	3
P8	6	3
P9	7	3
P10	8	3

All models (*SGNB*, *SGIB*, *SGCB*, *DGNB*, *DGIB*, and *DGCB)* are solved considering problems {P1…P10}. The capacity of all buffers is limited to one. This limitation is a good way to study the effect of the robot types when serving in the same manufacturing system. This is true especially when the size of the buffersand their capacity is critical in industrial applications.

Tables 10-11 present the results of the test problems when minimizing the MFT. The dual-gripper robot and single-gripper formulations are solved using the default search mechanism in the CPO. The default search uses default combinations that are proven to be the best choice on an average IBM Corp (2011). For these small to fairly medium problems, the time limit is set to 2000 sec. This limit is found to be large enough to provide realistic and accurate results for the test problems. In Tables 10-11, the first column represents the generated problem. The second column represents the considered formulation. This formulation includes the *DGNB*, *DGIB*, *DGCB*, *SGNB*, *SGIB*, and *SGCB*. Column 3 represents the number of constraints. Column 4 represents the number of variables. Columns 5 and 6 represent the objective value and the computational time in seconds for the first obtained solution. Columns 7 and 8 represent the objective value and the computational time in seconds for the optimal (or near optimal) solution obtained at the end of the search process. Column 9 represents the buffer capacity set up in this experiment.

5.3.1. Performance of the JSSP

Table 10 presents the results found using different problem sizes with different scenarios of the JSSP.

Table 10. Results of the JSSP

Problem	Model	Constraints	Variables	First solution		Best solution		Capacity
				Objective value	CPU time (seconds)	MFT time (sec	CPU time (seconds)	
P1 3X5	UB	32	35	188.33	0.04	175*	0.04	M
	DGNB	32	35	200	0.01	182.67	0.34	0
	DGIB	61	50	188.33	0.01	175	0.9	1
	DGCB	57	50	188.33	0.01	175	0.06	1
	SGNB	44	35	200.33	511.71	200.33	511.71	0
	SGIB	73	50	175	0.14	175	0.14	1
	SGCB	69	50	175.00	0.14	175	0.14	1
P2 4x5	UB	41	45	183.5	0.01	169.25*	0.06	M
	DGNB	41	45	186.25	0.01	179.5	0.01	0
	DGIB	78	65	183.5	0.03	169.25*	0.07	1
	DGCB	74	65	183.5	0.01	169.25	0.04	1
	SGNB	57	45	191.50	0.17	191.5	0.17	0
	SGIB	94	65	183.50	0.07	169.25	0.07	1
	SGCB	90	65	179.50	0.29	179.50	0.29	1
P3 5x5	UB	50	55	196.8	0.01	183.8	0.14	M
	DGNB	50	55	207.8	0.01	193.6	0.26	0
	DGIB	95	80	196.8	0.07	183.80	0.39	1
	DGCB	91	80	196.8	0.06	184.20	0.31	1

	SGNB	70	55	214.40	0.06	203	0.21	0
	SGIB	115	80	196.80	0.03	183.80*	0.70	1
	SGCB	111	80	205.40	2.93	190.40	11.93	1
P4	UB	59	65	219.67	0.03	193.67	0.82	M
6x5	DGNB	59	65	244.17	0.09	205.17	0.43	0
	DGIB	112	95	219.67	0.07	193.67	5.70	1
	DGCB	108	95	232.33	0.18	193.67	1.24	1
	SGNB	83	65	259.00	72.99	253.00	190.72	0
	SGIB	136	95	208	0.10	193.67	3.79	1
	SGCB	132	95	231.83	2.79	200.67	38.23	1
P5	UB	68	75	291.28	0.06	230.71	5.67	M
7x5	DGNB	68	75	260.28	0.04	240.43	0.32	0
	DGIB	129	110	257.28	0.10	230.71	1.71	1
	DGCB	125	110	268	0.10	230.86	0.98	1
	SGNB	96	75	275.43	8.82	264.145	1416.98	0
	SGIB	157	110	282.86	0.31	230.71	3.69	1
	SGCB	153	110	247.43	1.52	236.29	10.93	1
P6	UB	23	27	147	0.00	121.75*	0.17	M
4x3	DGNB	23	27	146.75	0.00	129.25	0.00	0
	DGIB	42	39	123.75	0.06	121.75	0.21	1
	DGCB	40	39	124	0.03	121.75	0.17	1
	SGNB	31	27	145.25	0.28	145.25	0.28	0
	SGIB	50	39	131	0.01	121.75	0.15	1
	SGCB	48	39	139.50	0.17	121.75	0.23	1
P7	UB	28	33	186	0.01	142.8*	0.29	M
5x3	DGNB	28	33	166.2	0.01	151.60	0.17	0
	DGIB	51	48	160	0.01	142.80	0.10	1
	DGCB	49	48	160	0.03	142.8	0.04	1
	SGNB	38	33	169.20	1.24	169.20	1.24	0
	SGIB	61	48	162.80	0.03	142.80	0.37	1
	SGCB	59	48	151.00	0.12	143	18.14	1
P8	UB	33	39	154.33	0.04	132.33	1.38	M
6x3	DGNB	33	39	146.5	0.01	139.33	0.06	0
	DGIB	60	57	141	0.01	132.33	9.96	1
	DGCB	58	57	141	0.03	132.33	0.20	1
	SGNB	45	39	161.00	4.53	152.50	4.71	0
	SGIB	72	57	141	0.20	132.33	0.63	1
	SGCB	70	57	146.17	1.29	140.33	30.09	1
P9	UB	38	45	195.71	0.07	157.14	1.59	M
7x3	DGNB	38	45	187.57	0.01	164.86	1.48	0
	DGIB	69	66	161.85	0.12	157.14	273.67	1
	DGCB	67	66	161.85	0.07	157.14	0.20	1
	SGNB	52	45	192.43	1.71	179.43	82.10	0
	SGIB	83	66	160.43	0.24	158.29	1.41	1
	SGCB	81	66	177.57	0.20	159.72	452.37	1

P10 8x3	UB	43	51	217.37	0.01	172	2.01	M
	DGNB	43	51	209.12	0.07	182.87	5.47	0
	DGIB	78	75	181.25	0.07	172	9.65	1
	DGCB	76	75	181.25	0.03	172	0.29	1
	SGNB	59	51	209.12	8.45	195.13	968.26	0
	SGIB	94	75	177	0.14	172	2.77	1
	SGCB	92	75	202.38	0.21	176.37	16.52	1

M: infinite capacity
* Optimal solution

Based on the results shown in Table 10, the following observations can be made:

- As can be noticed, the unlimited buffer case achieves the best performance in minimizing the MFT among all models (*SGNB*, *SGIB*, *SGCB*, *DGNB*, *DGIB*, and *DGCB)* for all problems (P1, P2, P3, P4, P5, P6, P7, P8, P9 and P10). This performance is as expected because of the availability of unlimited buffers. In addition, the unlimited buffer case model has the least constraint among all developed models. The type of robot gripper does not affect the results. This, in fact, is due again to the assumed unlimited buffers. In case of a circular wait, buffers are always available to resolve the problem.

- The dual-gripper robot excels the single-gripper robot in the bufferless formulation for all problems. Figure 26 illustrates the improvement made by the dual-gripper robot in the bufferless formulation based on the test results.

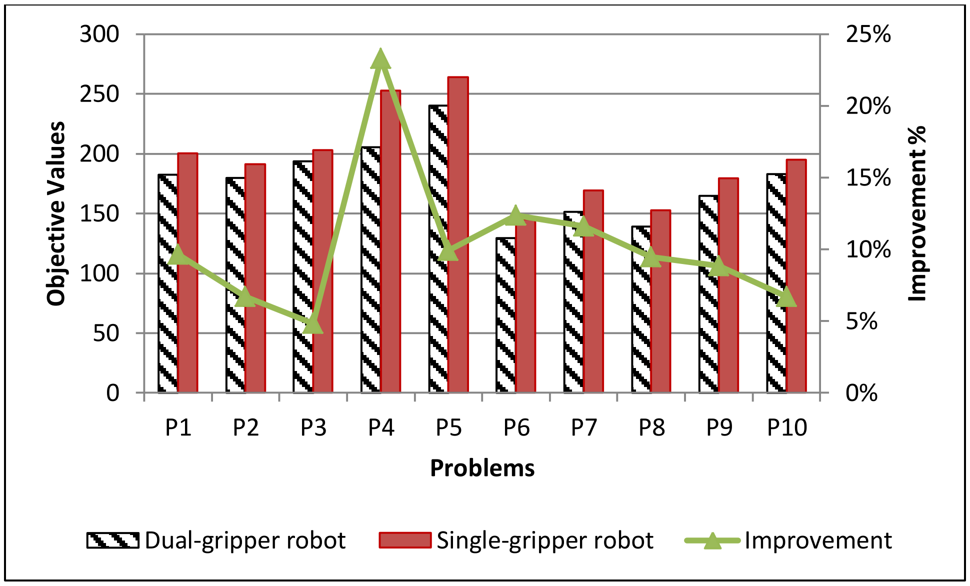


Figure 26. Improvement by the Dual-Gripper Robot in Bufferless Formulation (JSSP)

In the above figure, the observation made is that the dual-gripper robot has made an improvement in the objective values of 17.66 sec, 12 sec, 9.4 sec, 47.83 sec, 23.86 sec, 16 sec, 17.6 sec, 13.17 sec, 14.57 sec, and 12.26 sec for the P1, P2, P3, P4, P5, P6, P7, P8, P9 and P10 respectively. This improvement is quite significant. In fact, in the bufferless formulation with a single-gripper robot, no additional resources are available to deal with the deadlocks. The only solution is to constrain the machines to guarantee deadlock-free schedules. In turn, this constraining increases the objective values of the solutions. In contrast, in the bufferless formulation with dual-gripper robot, the robot acts as a flexible device that swaps jobs between machines when deadlocks occur. The model of the bufferless manufacturing cell with this type of robot does not need any deadlock-freeness constraint which, in turn, reduces the restrictions on the machines and then improves the objective values.

➢ The dual-gripper robot has practically made no improvement compared to the single-gripper robot in the input buffer formulation for the generated problems. Figure 27 illustrates the improvement made by the dual-gripper robot in the input buffers based on the test results.

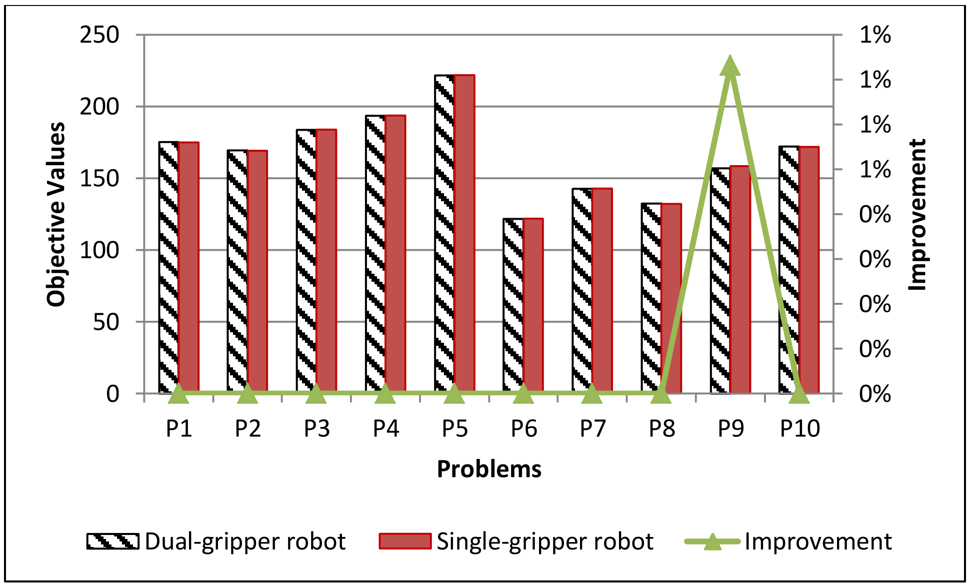


Figure 27. Improvement by the Dual-Gripper Robot in Input Buffers Formulation (JSSP)

In the above figure, the observation that can be made is that P9 is the only model where a slight improvement of 1% is made which is equivalent to 1 sec. For the remaining problems (P1, P2, P3, P4, P5, P6, P7, P8, and P10), the performance of the dual-gripper robot and the single-gripper robot are the same. The input buffer formulation with the dual-gripper robot reaches the same objective value with the single-gripper robot. For example, the objective value reached in P1 is 175 with either type of robot. These results show that with enough capacity in input buffers, no advantage can be realized by using the dual-gripper robot.

- The dual-gripper robot has made an improvement compared to the single-gripper robot in the central buffer formulation for most of the problems. Figure 28 illustrates the improvement made by the dual-gripper robot in the central buffer formulation based on the test results.

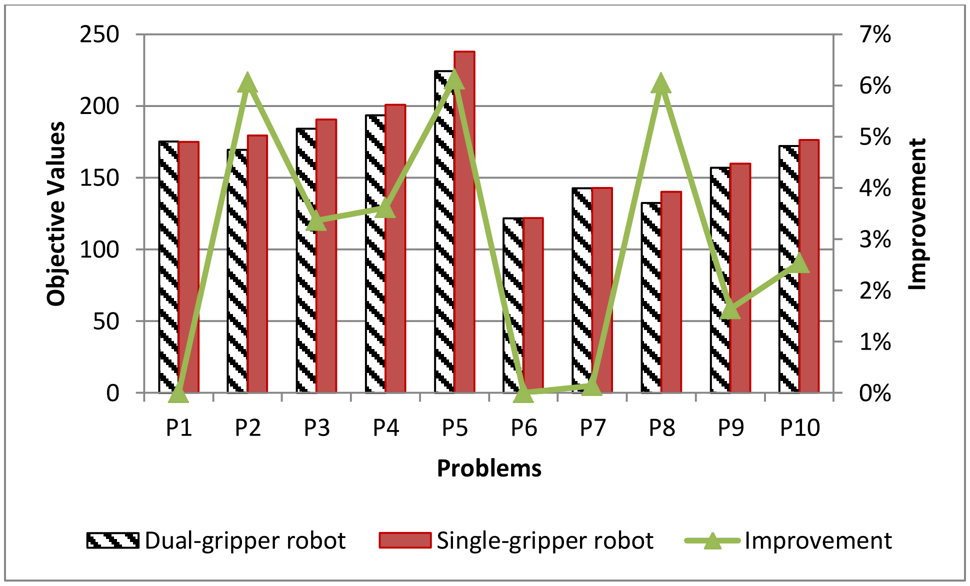


Figure 28. Improvement by the Dual-Gripper Robot in Central Buffer Formulation (JSSP)

The dual-gripper robot has made some improvement compared to the single-gripper robot in the central buffer formulation for many of the problems; P2, P3, P4, P5, P8, P9 and P10. The percentages of improvement for these problems are 6%, 3%, 4%, 6%, 6%, 2%, and 3% respectively. For example, the percentage of improvement in P5 is 6%. The objective value reached with the dual-gripper robot is 230.86 sec compared to 236.29 sec with a single-gripper robot. For the remaining problems (P1, P6, and P7), the dual-gripper robot has not made any improvement compared to the single-gripper robot. For example, the objective value reached in P1 is 175 with either of the robot type. These results show that even with the existence of a central buffer in manufacturing cells, the dual-gripper robot could be an asset for improving the objective values.

5.3.2. Performance of the F-JSSP

The results found using different problem sizes in different scenarios in F-JSSP are presented in Table 11.

Table 11. Results of the F-JSSP

Problem	Model	Constraints	Variables	First solution		Best solution		Capacity
				Objective value (sec	CPU time (seconds)	MS time (sec	CPU time (seconds)	
P1 3X5	UB	32	37	184.67	0.01	168.67*	0.03	M
	DGNB	32	37	170	0.00	169.33*	0.01	0
	DGIB	61	52	170	0.01	168.67*	0.04	1
	DGCB	57	52	170	0.00	168.67*	0.09	1
	SGNB	46	37	205.67	0.01	169.33	0.35	0
	SGIB	75	52	184.67	0.00	168.67	0.09	1
	SGCB	71	52	186	0.01	168.67	0.42	1
P2 4x5	UB	41	47	173.25	0.01	160.5*	0.10	M
	DGNB	41	47	179.5	0.01	160.5*	0.10	0
	DGIB	78	67	183.5	0.01	160.5*	0.07	1
	DGCB	74	67	183.5	0.03	160.5*	0.18	1
	SGNB	59	47	199.5	16.72	186	23.94	0
	SGIB	96	67	173.25	0.03	160.5	0.29	1
	SGCB	92	67	177	0.10	160.5	4.83	1
P3 5x5	UB	50	57	189.4	0.01	180.6*	0.20	M
	DGNB	50	57	211.8	0.00	187.4	0.31	0
	DGIB	95	82	189.4	0.07	180.6	0.07	1
	DGCB	91	82	189.4	0.01	180.6	0.15	1
	SGNB	72	57	213.20	0.09	201	0.54	0
	SGIB	117	82	189.40	0.01	180.60	0.74	1
	SGCB	113	82	204.20	1.57	180.60	2.09	1
P4 6x5	UB	59	68	219.33	0.07	190.17	6.33	M
	DGNB	59	68	239	0.01	201.83	0.40	0

	DGIB	112	98	214.33	0.03	190.17	1.48	1
	DGCB	108	98	218.17	0.03	190.17	2.04	1
	SGNB	86	68	255.67	68.17	221	1471.68s	0
	SGIB	139	98	223.17	0.09	190.17	6.17	1
	SGCB	135	98	228.50	6.47	197.33	35.36	1
P5 7x5	UB	68	78	258.28	0.03	209.29	73.69	M
	DGNB	68	78	230.28	0.10	216.86	2.94	0
	DGIB	129	113	219.85	0.03	209.29	239.28	1
	DGCB	125	113	225.14	0.06	210.57	19.06	1
	SGNB	101	78	305.85	1.95	249.43	800.48	0
	SGIB	162	113	236	5.69	209.29	103.70	1
	SGCB	158	113	232.71	3.33	221.86	37.33	1
P6 4x3	UB	23	29	129	0.00	108.5*	0.12	M
	DGNB	23	29	146.75	0.03	112.75	0.23	0
	DGIB	42	41	117	0.01	108.5	0.10	1
	DGCB	40	41	124	0.06	108.5	0.10	1
	SGNB	33	29	114.75	0.12	114.75	0.12	0
	SGIB	52	41	117	0.10	108.5	0.10	1
	SGCB	50	41	127.75	0.03	108.5	0.28	1
P7 5x3	UB	28	35	178.8	0.01	140.4	0.26	M
	DGNB	28	35	156.8	0.01	141.6	0.15	0
	DGIB	51	50	157.6	0.04	140.4	2.01	1
	DGCB	49	50	157.6	0.04	140.4	1.49	1
	SGNB	40	35	170.20	3.58	169.20	8.14	0
	SGIB	63	50	179.20	0.03	140.4	0.34	1
	SGCB	61	50	150.60	0.35	140.4	10.85	1
P8 6x3	UB	33	41	154.33	0.00	129.33	1.41	M
	DGNB	33	41	146.5	0.00	133	0.20	0
	DGIB	60	59	139.33	0.01	129.33	1.73	1
	DGCB	58	59	139.33	0.01	129.33	0.73	1
	SGNB	47	41	156.83	7.17	149.67	10.26	0
	SGIB	74	59	139.33	0.24	129.33	3.55	1
	SGCB	72	59	145.33	1.59	133	47.90	1
P9 7x3	UB	38	48	185.71	0.07	152.14	1.54	M
	DGNB	38	48	197.28	0.01	161.14	0.96	0
	DGIB	69	69	168.85	0.17	152.14	1.62	1
	DGCB	67	69	175.14	0.03	152.14	1.32	1
	SGNB	55	48	206.14	2.90	179.43	258.02	0
	SGIB	86	69	168.43	0.23	152.15	18.17	1
	SGCB	84	69	163.14	0.17	153.57	435.92	1
P10 8x3	UB	43	54	216.62	0.01	165.62	64.74	M
	DGNB	43	54	206.25	0.01	173.62	1.45	0
	DGIB	78	78	184.62	0.03	165.62	6.83	1
	DGCB	76	78	184.62	0.04	165.62	6.39	1
	SGNB	62	54	204	5.99	203.13	572.49	0

	SGIB	97	78	184.75	0.17	165.62	75.13	1
	SGCB	95	78	194.87	1.17	168.75	183.02	1

M: infinite capacity
* Optimal solution

Based on the results in Table 11, the following observations are made:

- The unlimited buffer case achieves the best performance in minimizing the MFT among all models (*SGNB*, *SGIB*, *SGCB*, *DGNB*, *DGIB*, and *DGCB)* for all problems (P1, P2, P3, P4, P5, P6, P7, P8, P9 and P10). This performance is as expected because of the assumed availability of unlimited buffers which makes the unlimited buffer case model have the least constraint among all developed models. Even the robot type does not affect the results. These results of the UB are obtained with the dual-gripper robot as with the single gripper robot. This, in fact,is due again to the assumed unlimited buffers. In case of a circular wait, buffers are always available to resolve the problem.

- The dual-gripper robot excels the single-gripper robot in the bufferless formulation for all problems. Figure 29 illustrates the improvement made by the dual-gripper robot in the bufferless formulation based on the test results.

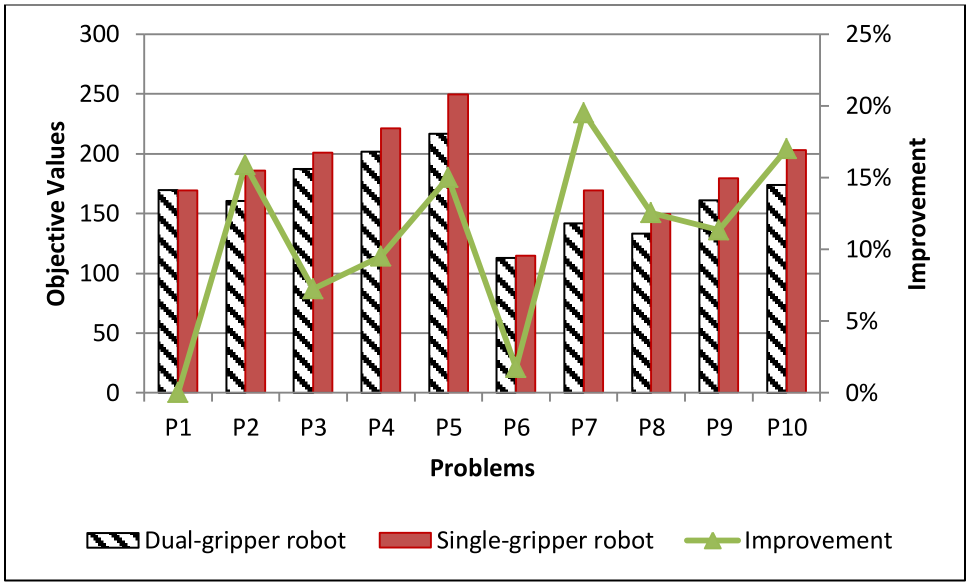


Figure 29. Improvement by the Dual-Gripper Robot in Bufferless Formulation (F-JSSP)

Most of the problems are influenced by the robot type. P2, P3, P4, P5, P6, P7, P8, P9, and P10 had the dual-gripper robot performing better than the single-gripper robot. The gain using the dual gripper robot in these problems is 25.5 sec, 13.6 sec, 19.17 sec, 32.57 sec, 2 sec, 27.6 sec, 16.67 sec, 18.29 sec, and 29.51 sec respectively. In turn, the percentages of improvement is 16%, 7%, 9%, 15%, 2%, 19%, 13%, 11%, and 17% respectively. Noticeably, P1 reaches the same result of 169.33 sec for this bufferless formulation with the dual-gripper robot or the single gripper robot. From this, it can be concluded that the robot type has less effect in P1 because the routing flexibility in P1 is enough to guarantee a deadlock-free schedule without using the dual-gripper robot.

- The dual-gripper robot has not made any improvement compared to the single-gripper robot in the input buffer formulation for all problems; P1, P2, P3, P4, P5, P6, P7, P8, and P10. Figure 30 illustrates the improvement made by the dual-gripper robot in the input buffers based on the test results.

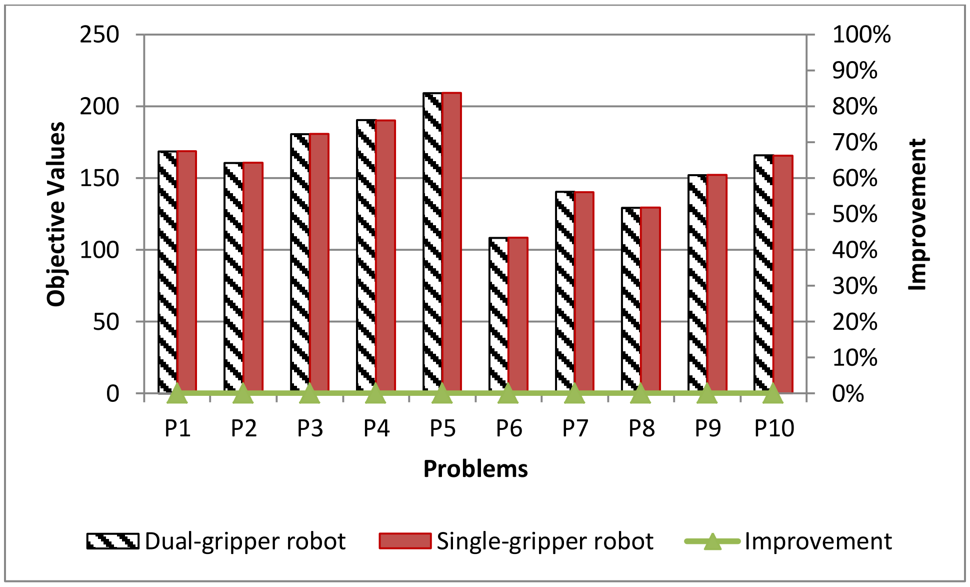


Figure 30. Improvement by the Dual-Gripper Robot in Input Buffers Formulation (F-JSSP)

As can be noticed, the routing flexibility and the existence of the buffers guarantee deadlock –free schedules with either of the dual-gripper or single-gripper robot. A justification for this result could be that whenever a deadlock happens, either the jobs use alternative machines, or available input buffers of the machines involved in the circular waits. Thus, the dual-gripper robot does not need to do any swapping activities.

- The dual-gripper robot has made an improvement compared to the single-gripper robot in the central buffer formulation for most of the problems. Figure 31 illustrates the improvement made by the dual-gripper robot in the central buffer formulation based on the test results.

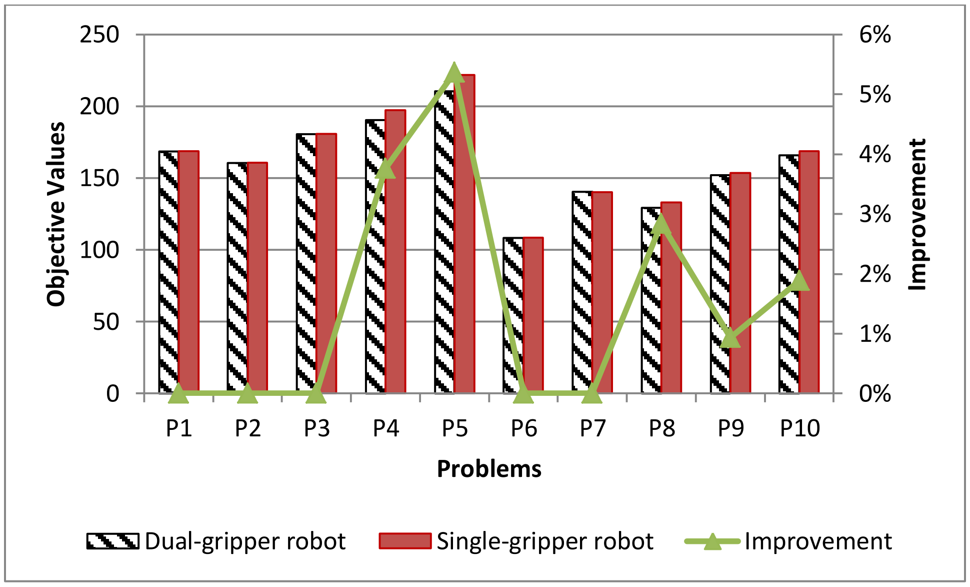


Figure 31. Improvement by the Dual-Gripper Robot in Central Buffer Formulation (F-JSSP)

In the above figure, the dual-gripper robot has made some improvement compared to the single-gripper robot in the central buffer formulation for P4, P5, P8, P9 and P10. The percentages of improvement for these problems are 4%, 5%, 3%, 1%, and 2% respectively. For example, the objective value reached in P4 with the dual-gripper robot is 190.17 sec and the objective value reached with a single-gripper robot is 197.33 sec. This variation in the objective values is equivalent to an improvement of 4%. For the remaining problems (P1, P2, P3, P6 and P7), the dual-gripper robot has not made any improvement compared to the single-gripper robot. For example, the objective value reached in P2 is 160.5 sec with either type of robot. These results show that even with the existence of the central buffer in manufacturing cells or routing flexibility, the dual-gripper robot could be an asset for improving the objective values.

The JSSP and the F-JSSP reacts the same way with the dual-gripper robot in the manufacturing cells even though the routing flexibility gives advantage to the F-JSSP. Some interesting findings that can be summarized are as follows:

- The gain of having a dual-gripper robot depends on the configuration of the manufacturing cell. Each FMC (bufferless manufacturing cells, manufacturing cells with input buffers, and manufacturing cells with central buffer) performs differently as seen in the performance evaluation results for both JSSP (Table 10) and F-JSSP (Table 11). This is due to the fact to each manufacturing cell has different resources type.

- Identifying the best performing cell configuration is based on objective values of the solutions. The UB scenario is the best among all. Clearly, the availability of buffers has its effects in this case. Buffer capacities are assumed large enough to solve any circular waits between the machines. However, assuming unlimited buffer capacities is not realistic in an industrial case. Therefore, this case is not likely to happen in an industrial scale where the batch size of the parts is considerable.

- The dual-gripper robot has tremendous effect on the productivity improvement in the bufferless manufacturing cells. The percentage of improvement in this type of cell is high. The main reason for this gain is due to the swapping activity of the jobs between the resources presented to the dual-gripper robot. However, the bufferless formulation is the least performing cell configuration in terms of the objective values.

- The dual gripper robot in the input buffer formulation is similar to the single gripper robot in all problems. The results reveal the fact that the perceived gain of the dual-gripper robot is substantially reduced with the existence of the input buffers in the manufacturing cells. Having input buffers in the cells could be more economical with a single gripper robot.

- The benefit of the dual gripper robot with central buffers in the manufacturing cell is not consistent. As can be noticed in the results, some problems obtain the same objective value with the dual-gripper robot as with the single gripper robot and other problems obtain better objective values with the dual-gripper robot than with the single gripper robot. This variation in the results is due to the fact that the limited capacity of the central buffer is enough for some problems and not for others. This is an interesting observation and proves that the central buffer alone with a limited capacity does not guarantee the best performance all the time.

- From the analysis of the results in Table 10 and Table 11, the effect of the dual gripper robot is inconsistent between the problems. This inconsistency is seen in the central buffer formulation (Figure 28 and Figure 31) and the bufferless formulations (Figure 26. Figure 29). The percentages of improvement are not based on the number of machines or the number of jobs. In this regard, there are not any firm conclusions made in regards to the relationship between the robot type and the problem size.

- The results concerning the productivity improvement using dual-gripper robot in JSSP and F-JSSP are similar. This might be due to the similarity in arrangement of resources in the manufacturing cell. Interestingly enough, the flexible routing has its role on reducing the effect of the dual-gripper robot. For example, in the JSSP, the improvement made by this robot type is up to 23% (Figure 26). However, the maximum improvement made by the F-JSSP is 19 % (Figure 29). The reason behind this is that the jobs take advantage of the existing alternative routes before taking advantage of the additional resources in the manufacturing cell whose capacity might be needed by another job that has less routing flexibility. That way, F-JSSP benefits more from the routing flexibility and reduces the effect of the limited resources.

5.4. Conclusions

In this chapter, an extensive performance evaluation has been presented for the developed models. This performance evaluation includes a comparative study between the developed models with the dual-gripper robot and the existing results from the previous literature for a single-gripper robot. The results of the comparative study prove that the cells with the dual-gripper robot outperform the reported results in the literature for corresponding cells serviced by a single-gripper robot. A variety of test problems are generated and solved using the proposed models. The results show how robot type can be critical for some manufacturing cells. Certain scenarios of the manufacturing cell can have a higher productivity improvement compared to others. The advantage of having a dual-gripper robot is very clear in bufferless formulation. The central buffer formulation gain some benefit for this type of robot. Only input buffer formulation is unaffected by the robot type. Finally, the results for the F-JSSP show that the routing flexibility reduces the need for buffers or a dual-gripper robot in the manufacturing cells since deadlocks are resolved using alternative resources.

Chapter 6 CONCLUSIONS AND RECOMMENDATIONS

6.1. Introduction

The motivation for this study is rooted in today's industrial need. New technologies play a crucial role for solving industrial problems. The problem of deadlock-free scheduling in F-JSSP and JSSP is investigated in this research. The deadlock situation has been solved in the past literature considering different resources. Unlike any studies found in the previous literature, this thesis proposes, for the first time, that dual-gripper robots can be very effectively used for resolution of deadlocksin FMCs. The performance analysis of various configurations of manufacturing cells shows that the dual-gripper robot outperforms the single-gripper robot and thus increases the productivity in many settings. Similarly, the deadlock situation has been solved in the past literature using different modeling approaches. Unlike these studies, this thesis proposes for the first time the scheduling with deadlock resolution using the constraint programming methodology.

6.2. Thesis Results

The modeling task of the deadlock-free scheduling problem in the manufacturing cells scenarios with different configurations is done with IBM ILOG CP Optimizer. These configurations include a manufacturing cell with unlimited buffers capacity, and single-gripper robot or dual-gripper robot with different buffer settings including: no buffer, central buffer, and input buffer. The models are built based on the available CPO scheduling language construct. The modeling task using CPO is relatively easier than other methods. The challenges surrounding the buffer modeling and the deadlock-free constraining are properly managed using CPO functions.

The developed models were solved and validated through an initial test problem. Gantt charts of the solutions were presented. The validation is done by tracking the different activities in the schedule to verify the effect of the constraining expressions (jobs' routing, machines' disjunction, deadlock-freeness, and buffers' capacities). Clearly, the Gantt charts proved that accuracy of the programming practice of the developed models as the constraining expressions are very effective. Similarly, the modeling capabilities of the IBM ILOG CP Optimizer have proven to be a very good solution methodology.

The dual gripper robot was compared with the single-gripper robot using test problems from previous literature. The results of dual-gripper robot are obtained using the developed models. The results of single-gripper robot are obtained from the literature. It is found that the dual-gripper robot outperforms the single-gripper robot in most cases. However, the number of test problems generated in the literature is limited and not enough to develop a rigorous synthesis of the effect of the dual-gripper robot. Consequently, more test problems are generated to understand the effect of the robot type in different class of problems. The productivity improvement is based on the objective value of the obtained solutions. The results show that benefits arising from the use of a dual-gripper robot are not always substantial in all cell configurations. The dual-gripper robot is highly recommended in bufferless manufacturing cells even when routing flexibility exists. However, the need for the dual-gripper robot in manufacturing cells with buffers is not as critical as is for the manufacturing cells with the bufferless formations. The dual-gripper robot madea very limited contribution for manufacturing cells with the central buffer because the productivity improvement in this model may not be practically significant. Further, the effect of the dual-gripper robot was not seen with input buffers. The strong impact of the input buffers makes the dual gripper robot less significant for productivity improvement.

To summarize, the current research shows that using dual-gripper robots can go beyond most existing deadlock resolution approaches. However, buffers' configurations and robot types must be strategically chosen at the cell design stage.

6.3. Recommendations

The first recommendation of this work is to develop an approach that can consider other activities in the manufacturing cell such as the transportation task. This function is hardly addressed with the use of the single-gripper robot in previous studies. Therefore, scheduling the dual-gripper robot with an appropriate deadlock-freeness is potentially quite interesting.

The second recommendation is regarding the development of the controller since most manufacturing systems havedynamic functionalities. The studied scenarios in this thesis require computerized systems that are designed to properly supervise and control the manufacturing activities' schedules.

Further, although the application of this work is not meant for a specific industrial case, this effort is considered to be a novel innovation and contribution in the field of FMCs. The difficulty of implementing this work in real manufacturing lines was confirmed by some local professionals in Winnipeg who believe that production lines of most Canadian manufacturing companies have not implemented the FMC philosophy. That being said, the previous work done by Fahmy (2009) for FMCs with limited capacity buffers and single-gripper robot has been implemented in the Robotics & Automation Laboratory at the University of Manitoba. The other alternative for this implementation is in virtual modeling of the proposed FMC. Virtual implementation is a highly interdisciplinary field since it is widely used in all fields of research from engineering and computer science to economics and social science, and at different levels, from academic research to manufacturing, Liu et al. (2010). The simulation will, of course, reinforce this thesis's results, ease the robotic cell design from an engineering perspective, and helps the decision-making side for a strategic manufacturing plan. Even more, the use of the virtual implementation can illustrate a new methodology for implementing scheduling framework theories into design and visualization. This realization can be done with the help of advanced programming skills in the existing virtual reality packages.

Finally, the research analytically proves that the dual-gripper robot have definite benefits when compared to a single-gripper robot in the bufferless formulation and the central buffer formulation. However, a practical analysis is needed to determine the related cost of each robot type in these manufacturing cells.

References

Abdallah, I. B., Elmaraghy, H. A., & Elmekkawy, T. (2002). Deadlock-free scheduling in flexible manufacturing systems using Petri nets. *International Journal of Production Research*, *40*(12), 2733-2756. doi:10.1080/00207540210136496

Baker, K. R. (1974). *Introduction to sequencing and scheduling*. New York: Wiley

Baptiste, P., Le Pape, C., & Nuijten, W. (2001). *Constraint-Based Scheduling: Applying Constraint Programming to Scheduling Problems*.

Ben Abdallah, I., El Maraghy, H., & El Mekkawy, T. (1998). An efficient search algorithm for deadlock-free scheduling in FMS using Petri nets. *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*. doi:10.1109/ROBOT.1998.677427

Chen, Y. L., Sun, T. H., & Fu, L. C. (1994). A Petri-net based hierarchical structure for dynamic scheduler of. *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on*. doi:10.1109/ROBOT.1994.351170

Damasceno, B. C., & Xie, X. (1998). Scheduling and deadlock avoidance of a flexible manufacturing system. *Systems, Man, and Cybernetics, 1998. 1998 IEEE International Conference on*. doi:10.1109/ICSMC.1998.725472

Dashora, Y., Kumar, S., Tiwari, M. K., & Newman, S. T. (2007). Deadlock-free scheduling of an automated manufacturing system using an enhanced colored time resource Petri-net model-based evolutionary endosymbiotic learning automata approach. *International Journal of Flexible Manufacturing Systems, 19*(4), 486. Boston: Springer Science & Business Media. doi:10.1007/s10696-008-9046-8

Dawande, M., Pinedo, M., & Sriskandarajah, C. (2009). Multiple Part-Type Production in Robotic Cells: Equivalence of Two Real-World Models. *Manufacturing & Service Operations Management*. INFORMS. doi:10.1287/msom.1070.0208

Dawande, M. W., & service), S. (Online. (2007). *Throughput optimization in robotic cells* (Vol. 101). New York: Springer.

Deering, P. E. (2012). Sufficient conditions for a flexible manufacturing system to be deadlocked. *International Journal of Industrial Engineering Computations*, *3*(1), 53-62. Growing Science. doi:10.5267/j.ijiec.2011.08.016

Drobouchevitch, I. G., Sethi, S. P., & Sriskandarajah, C. (2006). Scheduling dual gripper robotic cell: One-unit cycles. *European Journal of Operational Research*, *171*(2), 598-631. Elsevier B.V. doi:10.1016/j.ejor.2004.09.019

El-Tamimi, A. M., Abidi, M. H., Mian, S. H., & Aalam, J. (2012). Analysis of performance measures of flexible manufacturing system. *Journal of King Saud University - Engineering Sciences*, *24*(2), 115-129.

ElMekkawy, T. Y., & ElMaraghy, H. A. (2003). Real-time scheduling with deadlock avoidance in flexible manufacturing systems. *The International Journal of Advanced Manufacturing Technology*, *22*(3), 259-270. London: Springer-Verlag. doi:10.1007/s00170-002-1468-y

Fahmy, S A, ElMekkawy, T. Y., & Balakrishnan, S. (2007). Job shop deadlock-free scheduling using Mixed Integer Programming and rank matrices. *Systems, Man and Cybernetics, 2007. ISIC. IEEE International Conference on*. doi:10.1109/ICSMC.2007.4413614

Fahmy, S. (2009). *A hierarchical control system for scheduling and supervising flexible manufacturing cells*.

Fahmy, Sherif A, Balakrishnan, S., & ElMekkawy, T. Y. (2009). A generic deadlock-free reactive scheduling approach. *International Journal of Production Research*, *47*(20), 5657-5676. doi:10.1080/00207540802112652

Fahmy, Sherif A, ElMekkawy, T. Y., & Balakrishnan, S. (2008). Deadlock-free scheduling of flexible job shops with limited capacity buffers. *European Journal of Industrial Engineering*, *2*(3), 231-252. doi:10.1504/EJIE.2008.017685

Fahmy, Sherif A, ElMekkawy, T. Y., & Balakrishnan, S. (2010). Mathematical formulations for scheduling in manufacturing cells with limited capacity buffers. *International Journal of Operational Research*, *7*(4), 463-486. doi:10.1504/IJOR.2010.032422

Fanti, M. P., & Zhou, M. (2004). Deadlock control methods in automated manufacturing systems. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans* (Vol. 34, pp. 5-22). doi:10.1109/TSMCA.2003.820590

Gang, X., & Wu, Z. (2002). A kind of deadlock-free scheduling method based on Petri net. *High Assurance Systems Engineering, 2002. Proceedings. 7th IEEE International Symposium on*. doi:10.1109/HASE.2002.1173123

Gang, X., & Wu, Z. (2004). Deadlock-free scheduling strategy for automated production cell. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*. doi:10.1109/TSMCA.2003.820573

Geismar, H. N., Chan, L. M. A., Dawande, M., & Sriskandarajah, C. (2008). Approximations to Optimal k-Unit Cycles for Single-Gripper and Dual-Gripper Robotic Cells. *Production and Operations Management*, *17*(5), 551-563. Blackwell Publishing Ltd. doi:10.3401/poms.1080.0053

Geismar, H. N., Dawande, M., & Sriskandarajah, C. (2006). Throughput Optimization in Constant Travel-Time Dual Gripper Robotic Cells with Parallel Machines. *Production and*

Operations Management, 15(2), 311-328. Oxford, UK: Blackwell Publishing Ltd. doi:10.1111/j.1937-5956.2006.tb00247.x

Geismar, N., Dawande, M., & Sriskandarajah, C. (2011). Productivity Improvement From Using Machine Buffers in Dual-Gripper Cluster Tools. *IEEE Transactions on Automation Science and Engineering* (Vol. 8, pp. 29-41). New York: IEEE. doi:10.1109/TASE.2009.2039567

Golmakani, H. R., Mills, J. K., & Benhabib, B. (2006). Deadlock-free scheduling and control of flexible manufacturing systems using automata theory. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*. doi:10.1109/TSMCA.2005.851338

Hathout, L., & Balakrishnan, S. (2005). An Intelligent Movement Sequence in Production Process for Real Time Robot Control in a Multi-Machine Manufacturing Cell. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.133.9562

He, Y. L., & Wang, G. N. (2006). Petri Nets based Deadlock-free Scheduling for Flexible Manufacturing Systems. *Control, Automation, Robotics and Vision, 2006. ICARCV '06. 9th International Conference on*. doi:10.1109/ICARCV.2006.345377

Hu, H., & Li, Z. (2009). Local and global deadlock prevention policies for resource allocation systems using partially generated reachability graphs. *Comput. Ind. Eng., 57*(4), 1168-1181. Tarrytown, NY, USA: Pergamon Press, Inc. doi:10.1016/j.cie.2009.05.006

Huang, H., Li, J., & Zhou, W. (2008). Deadlock-Free Design of JSP with Multi-Resource Sharing. *Computer and Electrical Engineering, 2008. ICCEE 2008. International Conference on*. doi:10.1109/ICCEE.2008.184

Huang, Y.-S., Pan, Y.-L., & Zhou, M. (2012). Computationally Improved Optimal Deadlock Control Policy for Flexible Manufacturing Systems. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*. doi:10.1109/TSMCA.2011.2164241

Huang, Z., & Wu, Z. (2004a). Deadlock-free scheduling method for automated manufacturing systems with limited central buffers. *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*. doi:10.1109/ROBOT.2004.1307208

Huang, Z., & Wu, Z. (2004b). Deadlock-free scheduling method for automated manufacturing systems using genetic algorithm and Petri nets. *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*. doi:10.1109/ROBOT.2004.1307209

Huang, Z., & Wu, Z. (2004c). Deadlock-free scheduling in automated manufacturing systems with multiple resource requests. *Control Applications, 2004. Proceedings of the 2004 IEEE International Conference on*. doi:10.1109/CCA.2004.1387568

IBM Corp. (2011). *IBM ILOG CPLEX Optimization Studio V12.3 documentation*.

Jenkins, H. (2004). Design of Robotic End Effectors. *Robotics and Automation Handbook*. CRC Press. doi:doi:10.1201/9781420039733.ch11

Khayat, G. E., Langevin, A., & Riopel, D. (2006). Integrated production and material handling scheduling using mathematical programming and constraint programming. *European Journal of Operational Research, 175*(3), 1818-1832. Elsevier B.V. doi:10.1016/j.ejor.2005.02.077

Kim Marriott and Peter J. Stuckey. (1998). *Programming with Constraints: An Introduction*. The MIT Press.

Laborie, P., (2009), *IBM ILOG CP Optimizer for Detailed Scheduling Illustrated on Three Problems*, Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization ProblemsLecture Notes in Computer Science Volume 5547, pp 148-162

Lei, D. (2009). Genetic Algorithm for Job Shop Scheduling under Uncertainty. In U. Chakraborty (Ed.), (Vol. 230, pp. 191-228). Springer Berlin / Heidelberg. doi:10.1007/978-3-642-02836-6_7

Li, Zhiwu, Zhou, M., & service), S. (Online. (2009). *Deadlock resolution in automated manufacturing systems: a novel Petri Net approach*. London: Springer.

Li, Zhiwu, Wu, N., & Zhou, M. (2012). Deadlock Control of Automated Manufacturing Systems Based on Petri Nets-A Literature Review. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 42*(4), 437-462. doi:10.1109/TSMCC.2011.2160626

Liljenvall, T. (1999). Scheduling for production systems with limited buffers. *Systems, Man, and Cybernetics, 1999. IEEE SMC ’99 Conference Proceedings. 1999 IEEE International Conference on*. doi:10.1109/ICSMC.1999.816597

Liu, Z., Bu, W., & Tan, J. (2010). Motion navigation for arc welding robots based on feature mapping in a simulation environment. *Robotics and Computer Integrated Manufacturing, 26*(2), 137-144. Elsevier Ltd. doi:10.1016/j.rcim.2009.09.002

Lustig, I., Van Hentenryck, P., & NetLibrary, I. (1999). *The OPL optimization programming language*. Cambridge, Mass: MIT Press.

MONTAZERI, M., & VAN WASSENHOVE, L. N. (1990). Analysis of scheduling rules for an FMS. *International Journal of Production Research, 28*(4), 785-802. doi:10.1080/00207549008942754

Mati, Y., Rezg, N., & Xie, X. (2001). Geometric approach and taboo search for scheduling flexible manufacturing systems. *IEEE Transactions on Robotics and Automation* (Vol. 17, pp. 805-818). doi:10.1109/70.975998

Matta, A., Semeraro, Q., & Tolio, T. (2005). Configuration of AMSs. In Andrea Matta & Q. Semeraro (Eds.), (pp. 125-189). Springer Netherlands. doi:10.1007/1-4020-2931-4_4

Mejía, G., & Montoya, C. (2009). Scheduling manufacturing systems with blocking: a Petri net approach. *International Journal of Production Research*, *47*(22), 6261-6277. doi:10.1080/00207540802225983

Milano, M., & Wallace, M. (2010). Integrating Operations Research in Constraint Programming. *Annals of Operations Research*, *175*(1), 37-76. Boston: Springer US. doi:10.1007/s10479-009-0654-9

Murata, T. (1989). Petri nets: Properties, analysis and applications. *Proceedings of the IEEE, 77*(4), 541-580. doi:10.1109/5.24143

Ouelhadj, D., & Petrovic, S. (2009). A survey of dynamic scheduling in manufacturing systems. *Journal of Scheduling*, *12*(4), 417-431. Springer. doi:10.1007/s10951-008-0090-8

Pinedo, M. (2012). *Scheduling: theory, algorithms, and systems*. New York: Springer.

Piroddi, L., Cordone, R., & Fumagalli, I. (2008). Selective Siphon Control for Deadlock Prevention in Petri Nets. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans* (Vol. 38, pp. 1-1348). IEEE. doi:10.1109/TSMCA.2008.2003535

Ponnambalam, S., Jawahar, N., & Girish, B. (2009). Giffler and Thompson Procedure Based Genetic Algorithms for Scheduling Job Shops. In U. Chakraborty (Ed.), (Vol. 230, pp. 229-259). Springer Berlin / Heidelberg. doi:10.1007/978-3-642-02836-6_8

Ramaswamy, S. E., & Joshi, S. B. (1996). Deadlock-free schedules for automated manufacturing workstations. *IEEE Transactions on Robotics and Automation*, *12*(3), 391-400. doi:10.1109/70.499821

Sabuncuoglu, I., & Kizilisik, O. B. (2003). Reactive scheduling in a dynamic and stochastic FMS environment. *International Journal of Production Research, 41*(17), 4211-4231. Taylor & Francis. doi:10.1080/0020754031000149202

Sethi, S. P., Sidney, J. B., & Sriskandarajah, C. (2001). Scheduling in dual gripper robotic cells for productivity gains. *IEEE Transactions on Robotics and Automation* (Vol. 17, pp. 324-341). doi:10.1109/70.938389

Shi, X., & Wu, Z. (2005). Deadlock-Free Scheduling Method for FMSs Using Beam Search. *Systems, Man and Cybernetics, 2005 IEEE International Conference on*. doi:10.1109/ICSMC.2005.1571307

Singh, N. (1996). *Systems approach to computer-integrated design and manufacturing*. New York: Wiley.

Sriskandarajah, C., Drobouchevitch, I., Sethi, S. P., & Chandrasekaran, R. (2004). Scheduling Multiple Parts in a Robotic Cell Served by a Dual-Gripper Robot. *Operations Research, 52*(1), 65-82. INFORMS. doi:10.1287/opre.1030.0073

Su, Q., & Chen, F. F. (1996). Optimal sequencing of double-gripper gantry robot moves in tightly-coupled serial production systems. *Robotics and Automation, IEEE Transactions on*. doi:10.1109/70.481748

Um, I.-S., Lee, H.-C., & Cheon, H.-J. (2007). Determination of buffer sizes in flexible manufacturing system by using the aspect-oriented simulation. *2007 International Conference on Control, Automation and Systems*

Wysk, R. A., Yang, N. S., & Joshi, S. (1991). Detection of deadlocks in flexible manufacturing cells. *IEEE Transactions on Robotics and Automation*, *7*(6), 853-859. doi:10.1109/70.105378

Xing, K., Han, L., Zhou, M., & Wang, F. (2012). Deadlock-free genetic scheduling algorithm for automated manufacturing systems based on deadlock control policy. *IEEE transactions on systems, man, and cybernetics. Part B, Cybernetics : a publication of the IEEE Systems, Man, and Cybernetics Society*, *42*(3), 603-615. United States: Institute of Electrical and Electronics Engineers, Inc. doi:10.1109/TSMCB.2011.2170678

Xiong, H. H., & Zhou, M. (1997). Deadlock-free scheduling of an automated manufacturing system based on Petri nets. *Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on*. doi:10.1109/ROBOT.1997.614256

Xiong, H. H., Zhou, M., & Caudill, R. J. (1996). A hybrid heuristic search algorithm for scheduling flexible manufacturing systems. *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*. doi:10.1109/ROBOT.1996.506585

Yang, X.-S., & Koziel, S. (2011). *Computational optimization, methods and algorithms* (Vol. 356). New York: Springer.

Yoon, H. J. (2010). Scheduling for deadlock avoidance operation in robotic manufacturing cells. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, *224*(2), 329-340. doi:10.1243/09544054JEM1422

Zeballos, L. J. (2010). A constraint programming approach to tool allocation and production scheduling in flexible manufacturing systems. *Robotics and Computer-Integrated Manufacturing*, *26*(6), 725-743.

Zeballos, L. J., Quiroga, O. D., & Henning, G. P. (2010). A constraint programming model for the scheduling of flexible manufacturing systems with machine and tool limitations. *Engineering Applications of Artificial Intelligence*, *23*(2), 229-248.

Zhang, G., Gao, L., & Shi, Y. (2011). An effective genetic algorithm for the flexible job-shop scheduling problem. *Expert Systems with Applications*, *38*(4), 3563-3573.

Zhang, H., Gu, M., & Song, X. (2009). A deadlock-free scheduling with sequence-dependent setup times. *The International Journal of Advanced Manufacturing Technology*. doi:10.1007/s00170-009-1981-3

Zhou, M., McDermott, K., & Patel, P. A. (1993). Petri net synthesis and analysis of a flexible manufacturing system cell. *IEEE Transactions on Systems, Man, and Cybernetics*, *23*(2), 523-531. doi:10.1109/21.229464