**DOKUZ EYLÜL UNIVERSITY**
**GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES**

# THROUGHPUT OPTIMIZATION IN ROBOTIC CELLS

**by**
**Atabak ELMI**

**October, 2014**
**İZMİR**

# THROUGHPUT OPTIMIZATION IN ROBOTIC CELLS

**A Thesis Submitted to the**
**Graduate School of Natural and Applied Sciences of Dokuz Eylül University**
**In Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy in Industrial Engineering, Industrial Engineering Program**

**by**
**Atabak ELMI**

**October, 2014**
**İZMİR**

**Ph.D. THESIS EXAMINATION RESULT FORM**

We have read the thesis entitled **"THROUGHPUT OPTIMIZATION IN ROBOTIC CELLS"** completed by **ATABAK ELMI** under supervision of **ASSOC. PROF. DR. ŞEYDA TOPALOĞLU** and we certify that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Doctor of Philosophy.

Assoc. Prof. Dr. Şeyda TOPALOĞLU

Supervisor

Assoc. Prof. Dr. Zeki KIRAL

Thesis Committee Member

Asst. Prof. Dr. Gökalp YILDIZ

Thesis Committee Member

Prof. Dr. Levent KANDILLER

Examining Committee Member

Prof. Dr. Adil Baykasoğlu

Examining Committee Member

Prof. Dr. Ayşe OKUR

Director

Graduate School of Natural and Applied Sciences

## ACKNOWLEDGMENTS

First, I would like to point out my gratitude to my advisor, Assoc. Prof. Dr. Şeyda Topaloğlu for her guidance, continuing support, encouragement and invaluable advice throughout the progress of this dissertation. This research could not have reached its present form without their keen interest, direction and help. Their constant encouragement, friendly interaction, and constructive support has had a permanent effect on my way of thinking.

I also thank all the respected faculty members in the Department of Industrial Engineering, Dokuz Eylul University, who have directly or indirectly helped me during this research.

The most, my gratefulness and hearty thanks are due to my beloved parents, Mrs. Zarrin-Taj Nazari and Prof. Magsoud Elmi and my brother Dr. Afshin Elmi. They provided me with patience and courage and words are insufficient to express my deepest feelings of debts to them.

I am very much thankful to my dear friends Assis. Prof. Dr. Şener Akpinar, Dr. Hanifi Okan İşgüder, Dr. Alper Hamzadayi, Dr. Neslihan Avcu, and Dr. Abdurrahman Tosun . Their supportive relationship and cordial interaction alleviated many potential difficulties and provided a very delightful environment.

My special thanks are extended to Mrs. Nilgün Güleryüz and Mrs. Zühal Temiz, Student Affairs office, The Graduate School of Natural and Applied Science, Dokuz Eylul University, for their great cooperation and valuable helps during this research.

Atabak ELMI

## THROUGHPUT OPTIMIZATION IN ROBOTIC CELLS

## ABSTRACT

Scheduling problems can appear in almost any field of application in the real world. One specific class within this domain is the robotic cell scheduling problem. This thesis deals with throughput optimization in robotic cells. It arises in modern manufacturing companies, where the products move automatically between the different workstations. In this thesis, hybrid flow shop, cyclic flow shop, and cyclic job shop applications of robotic cells are studied. The encountered scheduling problems in mentioned systems are not a new research field, but hardly any work has been done in solving them considering multiple transportation robots. Within this thesis, this gap is closed and some first solution approaches are presented, discussing the structure of the problems. As a result, a mixed integer linear programming model and a simulated annealing algorithm are proposed for the hybrid flow shop robotic cell scheduling problem. Furthermore, a novel mixed integer linear programming model is proposed for the multi degree cyclic flow shop robotic cell scheduling problem with time window processing times. The optimal *k*-degree and cycle time can be found in reasonable computation times for realistic problem sizes with this model. An ant colony optimization approach is proposed to solve the multi degree cyclic flow shop robotic cell scheduling problem with free pickup criterion to find the appropriate *k*-degree and cycle time. The proposed ant colony optimization approach has been extended to solve the cyclic job shop robotic cell scheduling problem with multiple robots. The impact of transportation robots on the throughput rate and schedules are analyzed for each considered problem.

**Keywords:** Robotic cell, scheduling problems, flow shop, job shop, multiple robots.

## ROBOTİK HÜCRELERDE ÇIKTI OPTİMİZASYONU

## ÖZ

Çizelgeleme problemleri gerçek dünya uygulamalarının birçok alanında ortaya çıkabilir. Robotik hücre çizelgeleme problemi bu alanın özel bir dalıdır. Bu tez kapsamında robotik hücrelerde çıktı optimizasyonu konusu ele alınmaktadır. Bu konu, ürünlerin farklı iş istasyonları arasında otomatik olarak taşındığı modern üretim işletmelerinde ortaya çıkmaktadır. Bu tezde robotik hücrelerin melez akış tipi üretim, döngüsel akış tipi üretim ve döngüsel atölye tipi üretim uygulamaları çalışılmıştır. Bahsedilen sistemlerde karşılaşılan bu çizelgeleme problemleri yeni bir araştırma konusu olmamakla birlikte birden fazla taşıma robotunu dikkate alan herhangi bir çalışma yoktur. Bu tez kapsamında problemlerin yapısı tartışılarak, bu boşluk kapatılmış ve bazı yeni çözüm yaklaşımları geliştirilmiştir. Sonuç olarak, melez akış tipi robotik hücre çizelgeleme problemi için bir karışık tamsayılı doğrusal programlama modeli ve bir tavlama benzetimi algoritması önerilmiştir. Ayrıca, zaman pencereli işlem zamanları olan çok dereceli döngüsel akış tipi robotik hücre çizelgeleme problemi için yeni bir karışık tamsayılı doğrusal programlama modeli önerilmiştir. Bu model ile optimum *k*-derecesi ve çevrim zamanı gerçekçi problem boyutları için makul hesaplama zamanlarında bulunabilir. Serbest zamanlı boşaltım kriteri ile çok dereceli döngüsel akış tipi robotik hücre çizelgeleme problemini çözmek için uygun *k*-derecesini ve çevrim zamanını bulacak şekilde bir karınca kolonisi optimizasyonu yaklaşımı önerilmiştir. Önerilen karınca kolonisi optimizasyonu yaklaşımı döngüsel atöyle tipi robotik hücre çizelgeleme problemini birden fazla robotla çözecek şekilde genişletilmiştir. Taşıma robotları sayısının çıktı oranı ve çizelgeler üzerindeki etkisi düşünülen her bir problem için analiz edilmiştir.

**Anahtar kelimeler**: Robotik hücreler, çizelgeleme problemleri, akış tipi üretim, atölye tipi üretim, çoklu robotlar.

**CONTENTS**

**Page**

THESIS EXAMINATION RESULT FORM ..... ii
ACKNOWLEDGEMENTS ..... iii
ABSTRACT ..... iv
ÖZ ..... v
LIST OF FIGURES ..... x
LIST OF TABLES ..... xiv

**CHAPTER ONE - INTRODUCTION ..... 1**

1.2 A Classification Scheme for Robotic Cells ..... 5
1.2.1 Machine Environment ..... 5
1.2.1.1 Number of Machines ..... 5
1.2.1.2 Number of Robots ..... 5
1.2.1.3 Types of Robots ..... 5
1.2.2 Processing Characteristics ..... 6
1.2.2.1 Pickup Criterion ..... 6
1.2.2.2 Travel-Time Metric ..... 7
1.2.2.3 Number of Part-Types ..... 7
1.2.3 Objective Function ..... 7
1.2.4 An $\alpha|\beta|\gamma$ Classification for Robotic Cells ..... 7
1.2.4.1 Notation $\alpha = RS_{s,r}^{g}\ (m_1, \ldots, m_s)$ ..... 7
1.2.4.2 Notation $\beta = (pickup, travel - metric, part\ type - production\ type)$ 8
1.2.4.3 Notation $\gamma = \{\mu, Cmax\}$ ..... 8
1.3 Aims and Scope ..... 9
1.4 Outline of the Thesis ..... 11
1.5 Publications Produced Based on the Work in this Thesis ..... 13

**CHAPTER TWO - SCHEDULING MULTIPLE PARTS IN HYBRID FLOW SHOP ROBOTIC CELLS ..... 15**

2.1 Introduction ..... 15
2.2 Literature Review ..... 17
2.3 Single Robot ..... 21
2.3.1 Problem Statement and Mixed Integer Programming Model ..... 22
2.3.2 The Proposed Simulated Annealing Algorithm ..... 26
2.3.2.1 Encoding the Initial Solution ..... 27
2.3.2.2 Decoding Algorithm ..... 28
2.3.2.3 The Neighborhood Structure ..... 35
2.3.3 Computational Results ..... 39
2.3.4 Conclusion ..... 43
2.4 Multiple Robots ..... 43
2.4.1 Problem Statement and Mixed Integer Programming Model ..... 44
2.4.2 The Proposed Simulated Annealing Algorithm ..... 49
2.4.2.1 Encoding the Initial Solution ..... 50
2.4.2.2 Rectifying Blocking States ..... 51
2.4.2.3 Decoding Algorithm ..... 51
2.4.2.4 The Neighborhood Structure ..... 57
2.4.2.5 Cooling Schedule ..... 60
2.4.3 Computational Results ..... 63
2.4.3.1 Parameter Settings of $SA^{BP}$ ..... 64
2.4.3.2 Comparison Between $SA^{BP}$ and $SA^{IN}$ ..... 67
2.4.3.3 Comparing $SA^{BP}$ with Optimum Methods ..... 72
2.4.4 Conclusion ..... 73

**CHAPTER THREE - MULTI-DEGREE CYCLIC FLOW SHOP ROBOTIC CELL SCHEDULING PROBLEM WITH MULTIPLE ROBOTS: A MIXED-INTEGER LINEAR PROGRAMMING MODEL ..... 74**

3.1 Introduction ..... 74
3.2 Literature Review ..... 75
3.3 Problem Definition ..... 78
3.4 Mathematical Programming Model ..... 80

3.4.1 Mixed-Integer Non-Linear Programming Model ..... 81
3.4.1.1 Objective Function ..... 81
3.4.1.2 Basic Constraints ..... 82
3.4.1.3 Processing Time Constraints ..... 82
3.4.1.5 Operational Sequence Constraints for the Machines ..... 85
3.4.1.6 Operational Sequence Constraints for the Robots ..... 87
3.4.1.7 Cyclic Machine Availability Constraints ..... 88
3.4.2 Linearization of the Proposed Model ..... 93
3.5 Instance Problem ..... 96
3.6 Computational Results ..... 99
3.7 Conclusion ..... 105

**CHAPTER FOUR - MULTI-DEGREE CYCLIC FLOW SHOP ROBOTIC CELL SCHEDULING PROBLEM: ANT COLONY OPTIMIZATION ..... 106**

4.1 Introduction ..... 106
4.2 Literature Review ..... 107
4.3 Problem Definition ..... 110
4.4 Ant Colony Optimization (ACO) ..... 112
4.4.1 Initialization of the Pheromone Trails ..... 113
4.4.2 Transition Rule ..... 113
4.4.3 Updating of the Pheromone Trails ..... 116
4.4.3.1 Local Updating ..... 116
4.4.3.2 Global Updating ..... 117
4.5 The Multi Bridge Ant Colony Optimization (MB-ACO) ..... 117
4.5.1 Determining the Optimum k-Degree of the Cycle ..... 118
4.5.2 Determining the Occupied Machines at the Start of the Cycle ..... 119
4.5.3 Sequencing the Transportation Operations ..... 123
4.5.4 Determining the Robot Assignments for the Transportation Operations 124
4.5.5 Encoding Algorithm ..... 128
4.5.6 Decoding Algorithm ..... 130

4.6 Computational Analysis ..... 132
4.7 Conclusion ..... 136

**CHAPTER FIVE - SOLVING THE CYCLIC JOB SHOP ROBOTIC CELL SCHEDULING PROBLEM USING ANT COLONY OPTIMIZATION ..... 138**

5.1 Introduction ..... 138
5.2 Literature Review ..... 139
5.3 Problem Definition ..... 140
5.4 The Multi Bridge Ant Colony Optimization (MB-ACO) ..... 141
5.4.1 Determining the Occupied Machines at the Start of the Cycle ..... 142
5.4.2 Sequencing the Transportation Operations ..... 147
5.4.3 Robot Assignments for the Transportation Operations ..... 148
5.4.4 Encoding Feasible Solution ..... 152
5.4.5 Decoding Algorithm ..... 154
5.5 Computational Analysis ..... 156
5.6 Conclusion ..... 164

**CHAPTER SIX - CONCLUSION ..... 165**

**REFERENCES ..... 171**

## LIST OF FIGURES

**Page**

Figure 1.1 A three-machine overhead and track-based robot cells. .....3
Figure 1.2 A robotic cell with parallel machines and multiple robots.....3
Figure 2.1 An example of a hybrid robotic cell with parallel machines considered in this research.....22
Figure 2.2 Solution representation of BHFS-RCS problem.....29
Figure 2.3 Flowchart of the proposed rectifying algorithm. .....31
Figure 2.4 General flowchart of the proposed decoding algorithm. .....33
Figure 2.5 Graph representation of BHFS-RCS problem with six parts and four stages containing two machines.....34
Figure 2.6 Generating neighborhood structure via insertion method. .....36
Figure 2.7 Generating neighborhood structure via block properties based insertion method (done on critical operations no.11-13 mentioned in table 2.3). .36
Figure 2.8 An example of BHFS-RCS problem considered in this research. .....44
Figure 2.9 Solution representation of BHFS-RCS problem.....52
Figure 2.10 Flowchart of the proposed rectifying algorithm .....55
Figure 2.11 General flowchart of the proposed decoding algorithm. .....57
Figure 2.12 Generating neighborhood structure via insertion method based on block properties.....58
Figure 2.13 Graph representation of BHFS-RCS problem with 6 parts and 2 stages containing 2 machines. .....59
Figure 2.14 Average makespan reduction using different numbers of robots for each problem.....71
Figure 2.15 Interaction plot for the makespan of each problem obtained by $SA^{BP}$ ...71
Figure 3.1 An example of the cyclic flow shop robotic cell with multiple single-gripper robots. .....79
Figure 3.2 The graph representation of processing time constraints related to successive processing operations of a part (constraint set (3.11)). .....83

Figure 3.3 (A) The example gantt chart for 3-degree cyclic solution. (b) the cyclic change of the sequence of classes on machines after any occupied machine. ..... 86
Figure 3.4 The graph representation of the operational sequence of transportation operations performed by the same robot. ..... 88
Figure 3.5 The cyclic machine availability of machines between two successive parts. ..... 90
Figure 3.6 The Gantt chart of the instance problem indicating the sequence of all transportation operations and their assigned robots. ..... 98
Figure 3.7 The Gantt chart of three consecutive cycles for the instance problem. ..... 98
Figure 3.8 The graph model representation of processing and transportation operations for the instance problem. ..... 99
Figure 4.1 An example of the cyclic flow shop robotic cell with multiple single-gripper robots. ..... 111
Figure 4.2 The first set of nodes that determines the degree of cycle. ..... 119
Figure 4.3 The Gantt chart of the instance problem indicating the sequence of all transportation operations. ..... 120
Figure 4.4 The second set of nodes that determines the number of occupied machines at the start time of the cycle. ..... 120
Figure 4.5 The third set of nodes that determines each one of the occupied machines at the start time of the cycle. ..... 121
Figure 4.6 Determining the sequence of $Q = k \times M + 1$ number of transportation operations for the $k$-degree cycle (fourth step). ..... 123
Figure 4.7 The grid graph representation for the precedence relations of transportation operations. ..... 124
Figure 4.8 Determining the assigned robot to perform the $x^{th}$ transportation operation. ..... 125
Figure 4.9 The Gantt chart of the instance problem indicating the sequence of all transportation operations and their assigned robots. ..... 126
Figure 4.10 The general structure of the proposed MB-ACO algorithm and the regions that the artificial ants have to visit. ..... 127

Figure 4.11 The flowchart of the proposed mb-aco algorithm for the considered $k$DCFS-RCS problem. .....128
Figure 4.12 Encoding the feasible solution; the sequence of transportation operations. .....129
Figure 4.13 Encoding the feasible solution; the assigned robot to perform each transportation operation $[j, m]$. .....129
Figure 4.14 The impact of the number of robots on the throughput rate for the $k$DCFS-RCS problem. .....135
Figure 4.15 The obtained throughput rate ($\mu$) and determined $k$-degree for the generated problems with different single gripper robot numbers. .....136
Figure 5.1 The Gantt chart of the feasible solution for the instance problem (4 jobs and 7 machines including input and output devices).....143
Figure 5.2 The Gantt chart of two consecutive cycles for the instance problem. ....143
Figure 5.3 The first set of nodes that determines the number of occupied machines at the start time of the cycle. .....144
Figure 5.4 Determining each one of the occupied machines at the start time of the cycle. .....145
Figure 5.5 Determining the jobs that occupy the machines at the start time of the cycle. .....145
Figure 5.6 Determining the sequence of $\sum_{j=1}^{N}(n_j - 1)$ number of transportation operations for the $n$-job cycle (fourth step). .....147
Figure 5.7 The precedence relations for the provided instance problem (Figure 5.1). .....148
Figure 5.8 Determining the assigned robot to perform the $x^{th}$ transportation operation.....149
Figure 5.9 The Gantt chart of the instance problem indicating the sequence of all transportation operations and their assigned robots. .....150
Figure 5.10 The general structure of the proposed MB-ACO algorithm and the regions that the artificial ants have to visit. .....151
Figure 5.11 The flow chart of the proposed mb-aco algorithm for the considered CJS-RCS problem.....152

Figure 5.12 Encoding the feasible solution; the sequence of transportation operations. .....153
Figure 5.13 Encoding the feasible solution; the assigned robot to perform each transportation operation $[j, i]$. .....153
Figure 5.14 The obtained cycle time by the proposed MB-ACO for 10 jobs. .....161
Figure 5.15 The obtained cycle time by the proposed MB-ACO for 15 jobs. .....162
Figure 5.16 The obtained cycle time by the proposed MB-ACO for 20 jobs. .....162
Figure 5.17 The determined maximum height value of the cycle for different number of jobs and machines by the proposed MB-ACO algorithm. .....163
Figure 5.18 The main effects plot for determined maximum height value of the cycle. .....163

## LIST OF TABLES

**PAGE**

Table 2.1 Classification of the researches on multiple-part-type robotic cell scheduling problems. ....21
Table 2.2 Estimated number of constraints and variables for the proposed MILP model with respect to the number of parts, stages, and machines (*j, s, m*). ....26
Table 2.3 The extracted critical path for the example given in Figure 2.5. ....35
Table 2.4 Problems categorized into four groups. ....40
Table 2.5 Preferred levels of SA parameters for each group of problems separately. ....40
Table 2.6 Average makespan obtained by proposed $SA^{BP}$ and $SA^{IN}$ algorithms. ....41
Table 2.7 Computational results for the small size problems solved by $SA^{BP}$ and GAMS ....42
Table 2.8 Cooling schedule rates. ....60
Table 2.9 Problems categorized into four groups. ....65
Table 2.10 Considered levels of SA parameters for each group of problems. ....65
Table 2.11 Preferred levels of SA parameters for each group of problems separately. ....66
Table 2.12 ANOVA table of S/N ration for each group of problems. ....66
Table 2.13 Average makespan obtained by proposed $SA^{BP}$ and $SA^{IN}$ algorithms. ....69
Table 2.14 Average makespan obtained by proposed $SA^{BP}$ and $SA^{IN}$ algorithms. ....69
Table 2.15 The gap in each problem between $SA^{BP}$ and $SA^{IN}$ for different number of operative robots. ....70
Table 2.16 Maximum number of operative robots (MOR) for each problem. ....70
Table 2.17 Computational results for the small size problems solved by $SA^{BP}$ and GAMS ....73
Table 3.1 Literature of cyclic flow shop robotic cell. ....77
Table 3.2 Absolute value function transformations for two constraint forms. ....94
Table 3.3 Additional variables defined for absolute value function transformation ..94
Table 3.4 The related data of the considered instance problem. ....97

Table 3.5 The start time of all transportation operations found for the instance problem. ..... 97
Table 3.6 The assigned robots to perform each transportation operation of the instance problem. ..... 97
Table 3.7 (1) The proposed MILP model statistics for the randomly generated instance problems. ..... 101
Table 3.7 (2) The proposed MILP model statistics for the randomly generated instance problems. ..... 102
Table 3.8 (1) The proposed MILP model statistics for different problem sizes. ..... 103
Table 3.8 (2) The proposed MILP model statistics for different problem sizes. ..... 104
Table 4.1 Literature of cyclic flow shop robotic cell. ..... 109
Table 4.2 The related data of the considered instance problem. ..... 118
Table 4.3 The sub-sets for the consequent transportation operations of each class in Figure 4.3. ..... 122
Table 4.4 The assigned robot to perform each transportation operation of instance problem. ..... 125
Table 4.5 The generated problem sizes and the proposed MB-ACO algorithm's obtained solutions. ..... 134
Table 5.1 The jobs operations related data (processing time and processing machine). ..... 142
Table 5.2 The needed transportation time after the completion of each processing operation. ..... 142
Table 5.3 The empty move times of robots between the machines. ..... 142
Table 5.4 The start time of transportation operations for the feasible solution of the instance problem (Figure 5.1). ..... 143
Table 5.5 The sub-sets for the consequent transportation operations of each job in Figure 5.1. ..... 147
Table 5.6 The assigned robot to perform each transportation operation of the instance problem. ..... 149
Table 5.7 The generated 10-job problem and the proposed MB-ACO algorithm's obtained solutions. ..... 158

Table 5.8 The generated 15-job problem and the proposed MB-ACO algorithm's obtained solutions. ..... 159
Table 5.9 The generated 20-job problem and the proposed MB-ACO algorithm's obtained solutions. ..... 160

# CHAPTER ONE
# INTRODUCTION

## 1.1 Background and Motivations

Computer-controlled material handling systems that convey raw materials through the multiple processing stages required to produce a finished part or product are common in industry. One such implementation, a robotic cell, has become a standard tool in modern manufacturing. The efficient use of such cells requires algorithmic solutions to a variety of challenging combinatorial optimization problems; typical problems include cell design, optimal sequencing of robot moves, and scheduling of the products to be produced.

Many diverse industries use robotic cells. A dominant area of application is semiconductor manufacturing. Other documented implementations include electroplating lines for a variety of products ranging from printed circuit boards to airplane wings, where parts are transferred between tanks of chemicals by hoists. Robotic cells are also used for testing and inspecting boards used in mainframe computers, machining of castings for truck differential assemblies, crane scheduling for computer integrated manufacturing, textile mills, and engine block manufacturing. As manufacturers implement larger and more complex robotic cells, more sophisticated models and algorithms are required to optimize the operations of these cells. A number of studies have been conducted to find ways to meet this demand. Some date as far back as the late 1970s, but the majority has been performed since 1990.

A robotic cell consists of an input device; processing stages, each of which performs a different operation on each part in a fixed sequence; an output device; and one or more robots that transport the parts within the cell. Each stage has one or more machines that perform the processing for that stage. The default configuration of a robotic cell is to have the robots that can hold the parts, one or more machines per stage, and no buffers for intermediate storage between the stages within the cell.

Because each machine can hold only one part, a robotic cell is, in essence, a flow shop or job shop with blocking that has common servers that perform all transfers of materials between processing stations. In robotic cells, each function in the process is performed by a machine; there are no human-tended workstations. Because the material handling – movement between machines and loading or unloading of machines – is performed by robots, a variety of remote center compliance devices are available to ensure that the parts can be reliably loaded onto the machines in the correct orientation. Once this is accomplished, robots are advantageous for material handling because they can operate with speed and precision consistently for long periods. In addition, in some environments they are preferred because their use prevents contamination; examples include pharmaceutical compounding and semiconductor manufacturing. Other manufacturing environments are inhospitable to humans, so robots are a natural alternative: some semiconductor processing is done in a vacuum; welding and iron-working applications may be in high-temperature environments; and painting or applying other types of coatings may emit noxious fumes.

Different types of robots are used in industrial applications. In a common implementation for semiconductor manufacturing, the robot has a fixed base and an arm that rotates. Such a cell is commonly called a robot-centered cell. In another configuration, often used in electroplating printed circuit boards, the robot is attached by a hoist to an overhead track, and the entire robot moves linearly along this track (see Figure 1.1). A more general case combines these two: the robot's arm rotates on its base, and the robot itself moves linearly along a track (see Figure 1.1). In addition, multi robot cells with parallel machines included in processing stages have been indicated in Figure 1.2.

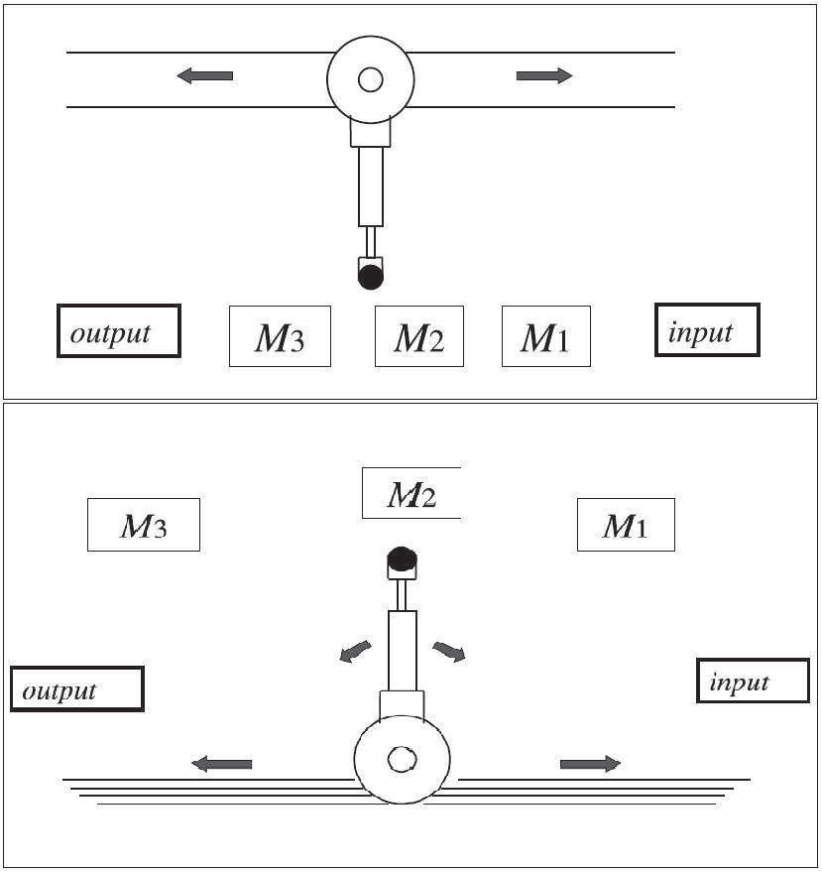



Figure 1.1 A three-machine overhead and track-based robot cells.

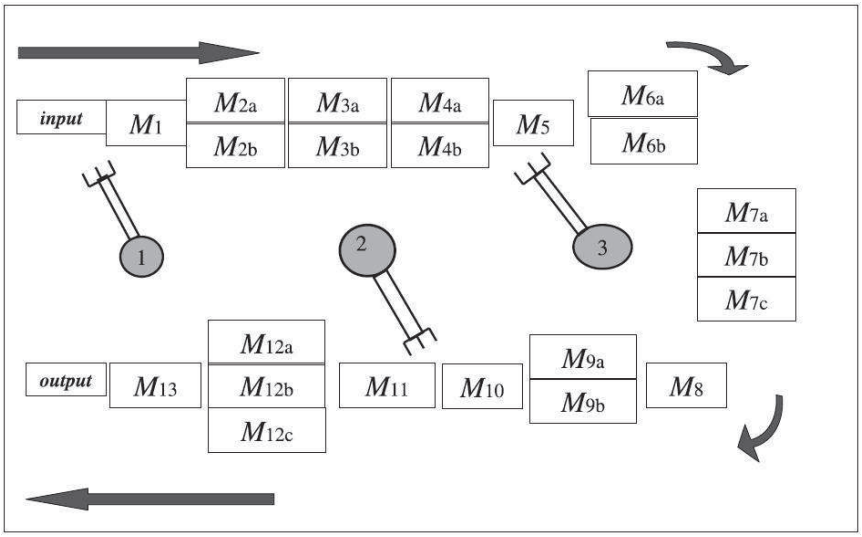



Figure 1.2 A Robotic cell with parallel machines and multiple robots.

Standardization of the processing requirements of the parts or products together with the volume required creates an ideal environment for repetitive production. In their typical use in practice, robotic cells are employed to produce significant

volumes of either a single part or a few closely related parts. Given the processing requirements, the objective that is most interested by manufacturers is the maximization of cell productivity. A natural and widely used measure of productivity is throughput; the number of finished parts produced per unit of time.

Given the goal of maximizing the throughput of the cell, two remarks need immediate mention. One, small improvements in throughput can improve revenues significantly for one or both of the following reasons: the significant volumes produced by the cell and the high market value of the products. Second, a number of cell characteristics impact its throughput. These include processing speeds of the machines and robots, cell layout, and the sequence of robot actions. For a specific manufacturing environment, an a priori judgment about the relative impact of these characteristics is often difficult to ascertain.

In practice, many cell parameters are fixed by physical constraints and cannot be altered. There is generally little flexibility in the layout of the cell, and changes to it would have relatively less influence on throughput. In most applications, processing requirements are strict; reducing the processing time at a stage would change the nature of that operation and its result. The processing speeds at the different stages of the cell are constrained by the latest technology available. Although various robots with different speeds are available, once a robot is purchased, it comes with a specified processing speed that cannot be changed.

Over the years, with an increase in the size and the processing complexity of robotic cells, optimization of the schedule of robot moves has emerged as a dominant tool for achieving throughput maximization. Besides, the best schedule of robot moves can be chosen relatively inexpensively. This requires no changes to the layout, the machines, or the robots. Consequently, implementing a schedule change requires little nonproductive time for the cell. Furthermore, these tactics can be used to determine the benefits that can be realized by adding advanced hardware, such as parallel machines or dual-gripper robots.

## 1.2 A Classification Scheme for Robotic Cells

We use the classification scheme for robotic cell scheduling problems described in M. W. Dawande, Geismar, Sethi & Sriskandarajah (2007). As in the classification scheme for classical scheduling problems, robotic cell scheduling problems are based on three characteristics: machine environment ($\alpha$), processing characteristics ($\beta$), and objective function ($\gamma$). A problem is then represented by the form $\alpha|\beta|\gamma$ (Figure 1.3).

### *1.2.1 Machine Environment*

Machine environment in robotic cells has different classification schemes which depend on number of machines in processing stages, number of robots, and types of robots.

#### *1.2.1.1 Number of Machines*

If each processing stage has only one machine, the robotic cell is a simple robotic cell, a robotic flow shop, or robotic job shop. Such a cell contrasts with a robotic cell with parallel machines, in which at least one processing stage has two or more identical machines.

#### *1.2.1.2 Number of Robots*

Manufacturers employ additional robots in a cell in order to increase throughput by increasing the material handling capacity. Cells with one (resp., more than one) robot are called single-robot (resp., multiple-robot) cells. Most studies in the literature analyze single-robot cells.

#### *1.2.1.3 Types of Robots*

A single-gripper robot can hold only one part at a time. In contrast, a dual-gripper robot can hold two parts simultaneously. In a typical use of this capability, the robot

holds one part while the other gripper is empty; the empty gripper unloads a machine, the robot repositions the second gripper and it loads that machine. In a single-gripper simple robotic cell, the robot cannot load a part to any machine if it is not empty.

### *1.2.2 Processing Characteristics*

Generally there are three different processing characteristics encountered in manufacturing.

#### *1.2.2.1 Pickup Criterion*

Robotic cells can be partitioned into three types − free pickup, no-wait, and time window − based on the pickup criterion. For all three types, a part that has completed processing on $m$ cannot be loaded onto $(m+1)$ for its next processing unless $m+1$ is unoccupied.

- *Free pickup:* In free pickup cells, this is the only pickup restriction; there is no limit on the amount of time a part that has completed processing on a machine can remain on that machine.
- *No-wait:* For the more restrictive no-wait cells, a part must be removed from machine $m$, and transferred to machine $(m+1)$ as soon as $m$ completes processing that part.
- *Interval:* In time window robotic cells, each stage has a specific interval of time – a processing time window – for which a part can be processed at that stage. Thus, if $[a_m, b_m]$ is the processing time window on machine $m$, then a part must be processed for $a_m$ time units on machine $m$, and must be transferred to machine $(m+1)$ within $(b_m - a_m)$ time units after its completion of processing on machine $m$.

*1.2.2.2 Travel-Time Metric*

Three different robot travel-time metrics have been addressed in the literature: *additive*, *constant*, and *Euclidean*. Most studies assume one of these. Therefore, many results in the field have been proven only for one travel-time metric rather than for all three.

*1.2.2.3 Number of Part-Types*

A cell producing identical parts is referred to as a single-part-type cell. In contrast, a multiple-part-type cell processes lots that contain different types of parts. Generally, these different part types require different processing times on a given machine.

### *1.2.3 Objective Function*

From an optimization aspect, the objective that is predominantly addressed in the literature is that of maximizing the throughput – the long-term average number of completed parts placed into the output buffer per unit time.

### *1.2.4 An $\alpha|\beta|\gamma$ Classification for Robotic Cells*

A problem is represented using the form $\alpha|\beta|\gamma$, where:

*1.2.4.1 Notation* $\alpha = RS_{s,r}^{g}\ (m_1, \ldots, m_s)$

Here, RS stands for Robotic System and it is notified by $RF$and $RJ$, if it was in a "Robotic Flow Shop" or "Robotic Job Shop" respectively. The symbol $s$ is the number of processing stages, and the vector $(m_1, \ldots, m_s)$ indicates the number of unrelated machines at each stage. When this vector is not specified, $m_s = 1$, $s \in \{1, \ldots, S\}$, and the cell is a simple cell. The second subscript r denotes the number of robots; when not specified, $r = 1$. The first superscript g denotes the

type of robot used. For example, $g = 1$ (resp., $g = 2$) denotes a single-gripper (resp., dual-gripper) cell. If $g$ is not specified, then $g = 1$.

#### *1.2.4.2 Notation* $\beta = (pickup, travel - metric, part\ type, production\ type)$

- $\text{pickup} \in \{\text{free}, \text{no} - \text{wait}, \text{interval}\}$ specifies the pickup criterion.
- $\text{travel} - \text{metric} \in \{\text{A}, \text{C}, \text{E}\}$specifies the travel-time metric, where $A$, $C$, and $E$ denote the *additive*, the *constant*, and the *Euclidean* travel-time metric, respectively.
- If *part type* is not specified, the cell produces a single part-type. Otherwise, *part-type MP* denotes a cell producing *multiple part-types*.
- The *production type* indicates the cyclic or non-cyclic production systems.

#### *1.2.4.3 Notation* $\gamma = (\mu, Cmax)$

Denotes the objective function of maximizing the throughput or minimizing the makespan.

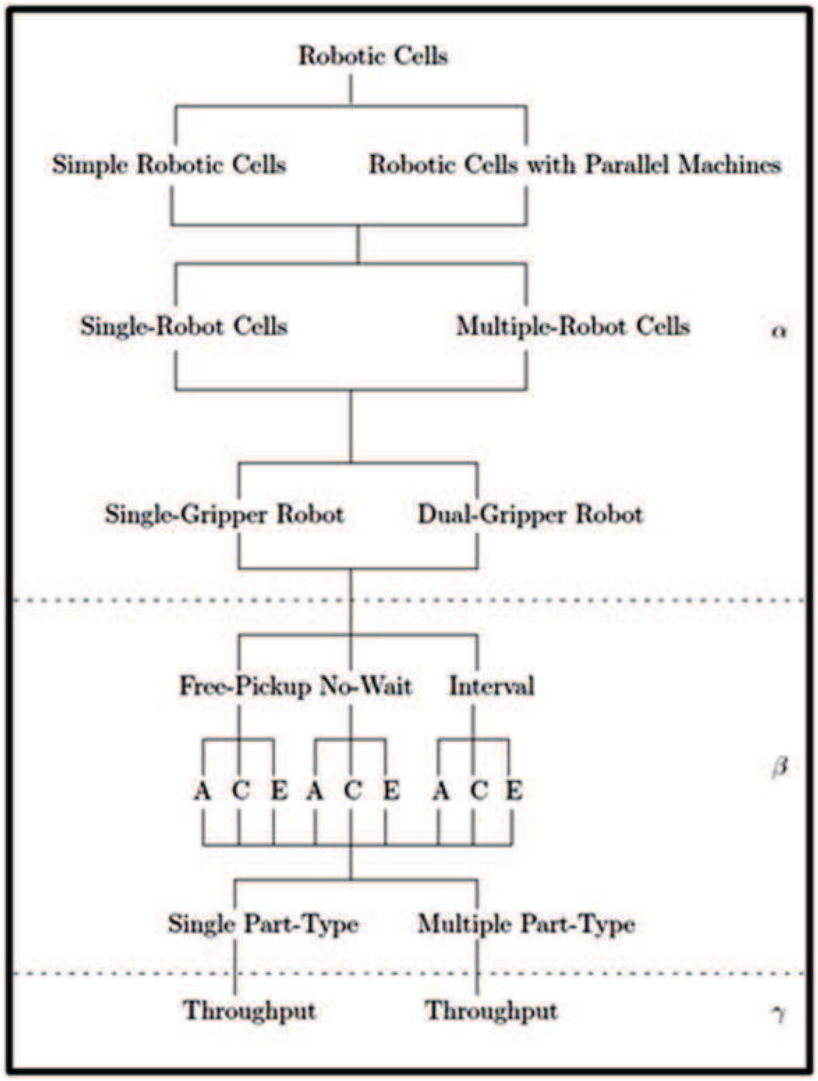



Figure 1.3 A classification of robotic cells.

## 1.3 Aims and Scope

Our research in this thesis refers to throughput optimization in robotic cells. As previously mentioned, the best way and not the only way of throughput optimization in robotic cells refers to solving the scheduling problem in these cells. Therefore, studying the following unsolved open scheduling problems in robotic cells is the main scope of this thesis.

❖ $RF_s(m_1, \dots, m_s)|(free, A, MP, non-cyclic)|Cmax$

Represents a robotic cell containing *s* number of *flow line* production stages, each one including $m_s$ number of *unrelated* machines. Also the referred cell has *one single-gripper robot*, a *free-pickup* criterion, and *additive travel-time* metric. It produces *multiple part types*. The objective function is that of minimizing the *makespan*.

❖ $RF_{s,r}(m_1, \dots, m_s)\big|(free, A, MP, non-cyclic)|Cmax$

Represents a robotic cell containing *s* number of *flow line* production stages, each one including $m$ number of *unrelated* machines. Also the referred cell has *multiple single-gripper robots*, a *free-pickup* criterion, and *additive travel-time* metric. It produces *multiple part types*. The objective function is that of minimizing the *makespan*.

❖ $RF_{s,r}\big|(interval, C, Cyclic)|\mu$

Represents a robotic cell containing *s* number of *flow line* production stages, each one including just *one machine*. Also the referred cell has *multiple single-gripper robots*, an *interval-pickup* criterion (time window processing), and *constant travel-time* metric. It produces a *single part-type*. The objective function is that of maximizing the *throughput*.

❖ $RF_{s,r}\big|(free, C, Cyclic)|\mu$

Represents a robotic cell containing *s* number of *flow line* production stages, each one including just *one machine*. Also the referred cell has *multiple single-gripper robots*, a *free-pickup* criterion, and *constant travel-time* metric. It produces a *single part-type*. The objective function is that of maximizing the *throughput*.

❖ $RJ_{s,r}|(free, C, MP, Cyclic)|\mu$

Represents a robotic cell containing *s* number of *job shop* production stages, each one include just *one machine*. Also the referred cell has *multiple single-gripper robots*, a *free-pickup* criterion, and *constant travel-time* metric. It produces *multiple part-types*. The objective function is that of maximizing the *throughput*.

### 1.4 Outline

In the following we summarize the organization of this thesis.

Chapter 2:

The blocking hybrid flow shop robotic cell (BHFS-RC) scheduling problem with unrelated parallel machines and machine eligibility constraints has been investigated in this chapter.

- The BHFS-RC scheduling problem with one single gripper robot is studied, $\{RF_s(m_1, \dots, m_s)|(free, A, MP, non-cyclic)|Cmax\}$.
  An introduction, literature review and problem statement have been given. Initially, a novel mixed-integer linear programming (MILP) model has been proposed for solving small size problems. Subsequently, a simulated annealing (SA) algorithm has been developed to solve large size problems effectively and efficiently. Additionally, a new neighborhood structure is used in the developed algorithm, which is based on block properties of operations.
- The BHFS-RC scheduling problem with multiple single gripper robots is studied, $\{RF_{s,r}(m_1, \dots, m_s)|(free, A, MP, non-cyclic)|Cmax\}$.

Chapter 3:

The multi-degree cyclic flow shop robotic cell scheduling (*k*DCFS-RCS) problem with time window processing times and multiple single gripper robots has been investigated in this chapter.

- An introduction, literature review and problem statement of the kDCFS-RCS problem with time window processing times have been given, $\{\mathrm{RF}_{s,r}|(\mathrm{interval}, \mathrm{C}, \mathrm{Cyclic})|\mu\}$.
- The newly proposed *non*-linear and linear mathematical models have been presented.
- An instance problem has been given to discuss the special characteristics of the studied problem.
- A computational analysis has been conducted to evaluate the performance of the developed model.

Chapter 4:

The *k*DCFS-RCS problem with free-pickup criterion and multiple single gripper robots has been investigated in this chapter.

- An introduction, literature review and problem statement of the *k*DCFS-RCS problem with free-pickup criterion have been given, $\{RF_{s,r}|(free, C, Cyclic)|\mu\}$.
- An ant colony optimization (ACO) approach featuring a multi-bridge structure has been proposed to solve the considered problem.
- The general structure of the multi-bridge ACO algorithm has been presented using an instance problem.
- A computational analysis has been conducted.

Chapter 5:

The multi-cyclic job shop robotic cell scheduling (CJS-RCS) problem with multiple single gripper robots and free-pickup criterion has been investigated in this chapter.

- An introduction, literature review and problem statement of the CJS-RCS problem with free-pickup criterion have been given, $\{RJ_{s,r}|(free, C, MP, Cyclic)|\mu\}$.
- An ACO approach featuring a similar multi-bridge structure as proposed in Chapter 4 has been developed to solve the considered problem.

- The general structure of the multi-bridge ACO algorithm has been presented using an instance problem.
- A computational analysis has been conducted.

## 1.5 Publications Produced Based on the Work in this Thesis

The following work has been produced during the creation of this thesis. Al though material from all of them has been assimilated into the thesis as a whole, they can be assigned as belonging particularly to the following specific following chapters or sections.

Chapter 2, Section 2.3:

- Elmi, A. & Topaloglu, S. (2014). Scheduling multiple parts in hybrid flow shop robotic cells served by a single robot. *International Journal of Computer Integrated Manufacturing*, 1-16.

Chapter 2, Section 2.4:

- Elmi, A. & Topaloglu, S. (2013). A scheduling problem in blocking hybrid flow shop robotic cells with multiple robots. *Computers & Operations Research*, 40, 2543-2555.

Chapter 3:

- Elmi, A. & Topaloglu, S. Multi-degree cyclic flow shop robotic cell scheduling problem with multiple robots: a mixed-integer linear programming model. *Applied Mathematical Modelling*, Under review.

Chapter 4:

- Elmi, A. & Topaloglu, S. Multi-degree cyclic flow shop robotic cell scheduling problem: ant colony optimization. *European Journal of Operational Research*, Under review.

Chapter 5:

- Elmi, A. & Topaloglu, S. Solving the cyclic job shop robotic cell scheduling problem using ant colony optimization. *Computers & Industrial Engineering*, Under review.

# CHAPTER TWO
# SCHEDULING MULTIPLE PARTS IN HYBRID FLOW SHOP ROBOTIC CELLS

## 2.1 Introduction

Industrial robots are used extensively in manufacturing companies performing tasks ranging from assembly to testing and inspection. Material handling is one of the important applications of robots. A robotic cell is defined as a manufacturing cell that is composed of a number of machines and a material handling robot. In a robotic cell, loading and unloading operations are made by the robots. These computer-controlled material handling and production systems provide many benefits, including speed and precision, reduced labor costs, the potential for contamination-free processing and the ability to operate in inhospitable environments. The efficient use of such a cell increases the production rate and thus is essential in order to compete in many different industries, including the manufacture of semiconductors, glass products, cosmetics, fiber-optics, printed circuit boards and building products. On the other hand, as manufacturers implement larger and more complex robotic cells, more sophisticated models and algorithms are required to optimize their operations.

In the classical flow shop problem, a set of multiple part-types flows through multiple stages in the same machine order, where each stage consists of only one machine. In some companies, to increase the processing capacity, additional machines may be required at some stages. This extended layout with the following assumptions are considered as hybrid flow shop systems (Zhihong Jin, Yang & Ito, 2006; Alaykyran, Engin & Döyen, 2007; Ruiz & Vázquez-Rodríguez, 2010; Ribas, Leisten & Framiñan, 2010).

- The number of processing stages $S$ is at least 2.
- At each stage $s$ the number of parallel machines $M_s \geq 1$, and at least at one of the stages $M_s > 1$. Job $j$ has to be processed on any one machine at any stage.

- Each job $j$ requires a processing time $P_{j,s}$ at each stage $s$.
- Preemption is not allowed and each machine can process one operation at a time.

In this chapter a class of hybrid flow shop scheduling problems is considered which contains more additional real world constraints: machines with different processing speeds at each stage, a single gripper robot to convey parts between stages and machine eligibility constraints. When machine eligibility constraints are present, a part can only be processed on a specified set of machines at each processing stage.

The considered robotic cell is in essence a *blocking* hybrid flow shop. Most robotic cells have no internal buffers. That is, a part cannot be moved from its current machine to the next one if the next one is occupied. This is called *blocking* in classical flow shop scheduling (Pinedo, 2002).

This research addresses the problem of sequencing robot moves to facilitate efficient material handling in a blocking hybrid flow shop with the single gripper robots to convey the parts between stages. To precisely describe the studied robotic cell scheduling problem, the $\alpha|\beta|\gamma$ classification scheme outlined in M. W. Dawande, et al. (2007) has been used.

$$\alpha|\beta|\gamma \Longrightarrow RF_s^r(um_1, \dots, um_s)|(free, Ad, MP)|C_{max} \tag{2.1}$$

Here, $RF$ stands for "Robotic Flow-line," subscript $s$ is the number of processing stages, and the vector $(um_1, um_2, \dots, um_s)$ indicates the number of unrelated parallel machines at each stage. The subscript $r$ denotes the number of robots.

$\beta = (free, Ad, MP)$ where, $free$ points the free pickup criterion, $Ad$ hints the additive travel-time metric, and $MP$ indicates multiple part-type.

In addition, $\gamma = C_{max}$ addresses the objective function of minimizing the makespan for the aforementioned problem.

The first field $\alpha$ denotes the *RF* robotic flow line system. The symbol *S* is the number of processing stages, and the vector $(um_1, um_2, \dots, um_s)$ indicates the number of parallel machines with different speeds at each stage. The second field denotes the $\beta = (pickup, travel - metric, part - type$ . Accordingly, a free pickup criterion is allowed in the robotic cell such that there is no upper bound on the amount of time that a completed part remains on a machine. The additive travel-time metric is used between any two machines, and besides multiple part-types are allowed. The third field denotes the objective of minimizing the makespan as an objective function.The particular case of this robotic cell problem where the transportation times are negligible and there is one machine at each processing stage reduces to the much studied flow shops cheduling problem with blocking (Pinedo, 2002). Hall & Sriskandarajah (1996) prove that this latter problem is strongly *NP*-hard for $m \geq 3$. Consequently, our problem is strongly *NP*-hard for $m \geq 3$ as well.

The blocking hybrid flow shop robotic cells (BHFS-RCs) can be found in different real world settings such as printed circuit board assembly lines in electronics industry (ZH Jin, Ohno, Ito & Elmaghraby, 2002), production of concrete in photographic film manufacturing industry (Tsubone Tsubone, Ohba, Takamuki & Miyake, 1993), or in non-manufacturing areas like container handling systems (Bostel & Dejax, 2006; Chen, Bostel, Dejax, Cai & Xi, 2007).

## 2.2 Literature Review

Most of the literature deals with cyclic scheduling in robotic cells with the objective of maximizing the throughput rate, or equivalently, minimizing the cycle time using repetitions of an optimal robot move cycle (for example see Sriskandarajah, Hall & Kamoun, 1998). Indeed, trying to determine the best sequence for a Minimal Part Set (MPS) with multiple part-types and producing this set repetitively can be considered as a special case of scheduling of different part types applied to cyclic systems. For this reason, minimizing the makespan for the MPS can be considered as maximizing throughput or minimizing cycle time in these cyclic systems.

Sethi, Sriskandarajah, Sorger, Blazewicz & Kubiak (1992) considered the two- and three-machine robotic cells with multiple part-types. They investigated the complexity of the problem and proposed a polynomial time algorithm that determines an optimal multi-part cycle given the robot move sequence. Bilge & Ulusoy (1995) studied the scheduling of Automatic Guided Vehicles (AGV) in an *m*-machine flexible manufacturing system. They provided a nonlinear mixed integer programming model and an iterative procedure in which the objective is to minimize the makespan. H. Chen, Chu & Proth (1997) investigated the two- and *m*-machine flow shop robotic cell. They proposed a branch and bound algorithm to find the optimal sequence of parts when the sequence of robot moves is given. Hall, Kamoun & Sriskandarajah (1998) studied the complexity and steady state performance of a three-machine robotic cell. They tried to minimize the cycle time through determining the robot moves and part sequences.

Sriskandarajah, et al. (1998) studied the *m*-machine robotic cell with multiple part-types. They classified the part sequencing problems associated with all the one-unit robot move cycles into four categories: (*i*) sequence independent; *(ii)* capable of formulation as a traveling salesman problem (TSP), but polynomially solvable; (*iii*) capable of formulation as a TSP and unary NP-hard; and (*iv*) unary NP-hard, but not having TSP structure. Aneja & Kamoun (1999) provided an algorithm of complexity O(*n log n*) that jointly optimizes the robot movesequence and part scheduling problem in a two-machine robotic cell. Agnetis (2000) and Agnetis & Pacciarelli (2000) investigated the complexity of a no-wait flow shop problem in which one robot is used to move the parts and found the optimal part schedules for cells with two and three machines. Hurink & Knust (2001) studied flow shop scheduling with explicit transportation capacity and transportation times. Moreover, they assumed an unlimited buffer space between the machines and negligible empty moving times. Furthermore, Hurink & Knust (2002) proved that a single robot job shop scheduling problem is *NP*-hard and proposed a tabu search algorithm. Soukhal & Martineau (2005) considered the flow shop robotic cell scheduling problem with multiple part-types and a single transportation robot. They proposed an integer linear programming model and a genetic algorithm to solve the problem. Soukhal, Oulamara & Martineau

(2005) investigated the two-machine flow shop scheduling problem taking transportation constraints into account. In their case, the finished jobs are transferred from the processing facility and then delivered to customers by truck. They proved that the mentioned problem with additional constraints, such as blocking, is strongly *NP*-hard.

Carlier, Haouari, Kharbeche & Moukrim (2010) proposed an approximate decomposition algorithm for the flow shop robotic cell scheduling problem with a single transportation robot and a blocking constraint. They initially determined the sequence of parts and then sequenced the robot moves for the given sequence of parts. Kharbeche, Carlier, Haouari & Moukrim (2010) provided a new MILP model for the flow shop robotic cell scheduling problem with a single robot and proposed an exact branch and bound algorithm. In addition, they proposed a genetic algorithm to solve large scale problems. Zahrouni & Kamoun (2012) studied the three-machine flow shop robotic cell with different part-types and proposed a constructive heuristic in the spirit of the NEH algorithm (Nawaz, Enscore Jr & Ham, 1983). Fazel Zarandi, Mosadegh & Fattahi (2013) considered the two-machine robotic cell with sequence dependent setup times for loading/unloading each part. They provided a MILP model to find the sequence of robot moves for a given sequence of parts. In addition they proposed a branch and bound algorithm and an improved SA algorithm to solve large scale problems.

Additionally, Geismar, Dawande & Sriskandarajah (2004) considered the cyclic flow shop single gripper robotic cells with parallel machines and constant travel time. Geismar, Dawande & Sriskandarajah (2006) also considered the same problem with dual gripper robot and mentioned that the obtained cycles have greater throughput than the best possible cycles in single gripper robotic cells. M. Dawande, Geismar, Sethi & Sriskandarajah (2005) focused on cyclic production and provided a survey on robotic cell scheduling problem. Shafiei-Monfared, Salehi-Gilani & Jenab (2009) studied the flow shop and open shop robotic cell with stochastic processing times for a single part type. Sarkheil & Jenab (2012) studied the optimality of one-unit cycles in the classical robotic cell. They proved that the optimal cycle is a 1-unit

cycle for an arbitrary number of machines in robotic cells served by a single gripper robot. Moreover, Geismar, Pinedo & Sriskandarajah (2008) considered robotic cells with parallel machines and multiple dual gripper robots. They reported that the method of exchanging parts between the robots has little bearing on the throughput. On the other hand, they asserted that the assignment of processing stages to the robots can have a significant effect on the potential throughput of a cell. Batur, Karasan & Akturk (2012) considered the scheduling problem in two-machine robotic cells producing multiple part-types. They proposed a two-stage heuristic to optimize the cycle time. Foumani & Jenab (2013) studied the robotic cell scheduling problem with two machines and identical parts assuming the swap ability for the busy robots.

The researches that studied the multiple-part-type robotic cell scheduling problem are shown in Table 2.1. The table also illustrates the position of this research among the related literature.

As far as the related research is concerned, transportation robots have not been considered in hybrid flow shop systems with multiple part-types in the literature. This research incorporates the transportation constraints into these systems as well.

Table 2.1 Classification of the researches on multiple-part-type robotic cell scheduling problems.

| Research | SN | MS | PCri | RMS | PS | Sol-Proc | Contributions |
|---|---|---|---|---|---|---|---|
| Sethi, et al. (1992) | 2, 3 | 1 | Free | Given | ✓ | - | Application of Gilmore and Gomory algorithm. |
| Bilge & Ulusoy (1995) | ***S*** | 1 | Free | ✓ | ✓ | Parallel | MILP model and iterative solution procedure. |
| H. Chen, et al. (1997) | 3, ***S*** | 1 | Free | Given | ✓ | - | A Branch and Bound algorithm. |
| Hall, et al. (1998) | 3 | 1 | Free | ✓ | ✓ | Separate | NP-completeness of robot move sequences. |
| Sriskandarajah, et al. (1998) | ***S*** | 1 | Free | ✓ | ✓ | Separate | Categorization of part sequences into 4 groups. |
| Aneja & Kamoun (1999) | 2 | 1 | Free | ✓ | ✓ | Separate | An *O(n log n)* algorithm. |
| Agnetis (2000) | 2, 3 | 1 | No-wait | ✓ | ✓ | Parallel | An *O(n log n)* algorithm for the 2-machine case. Optimality analysis of 3-machine case. |
| Hurink & Knust (2001) | 3, ***S*** | 1 | Free | ✓ | ✓ | Parallel | Complexity analysis of 2,3 and m-machine flow shop robotic cell. |
| Hurink & Knust (2002) | ***S*** | 1 | Free | ✓ | ✓ | Parallel | A tabu search approach for job shop robotic cell. |
| Soukhal & Martineau (2005) | ***S*** | 1 | Free | ✓ | ✓ | Parallel | MILP model and genetic algorithm. |
| Soukhal, et al. (2005) | 2 | 1 | Free | ✓ | ✓ | - | Complexity analysis considering transportation capacity. |
| Carlier, et al. (2010) | ***S*** | 1 | Free | ✓ | ✓ | Separate | Branch and Bound algorithm and a genetic algorithm for robot move sequences. |
| Kharbeche, et al. (2010) | ***S*** | 1 | Free | ✓ | ✓ | Parallel | A MILP model and a new Lower Bound and Branch and Bound algorithm. |
| Zahrouni & Kamoun (2012) | 3 | 1 | Free | ✓ | ✓ | Parallel | A constructive heuristic *MinMPSCycle* for a 3-machine robotic cell. |
| Fazel Zarandi, et al. (2013) | 2 | 1 | Free | ✓ | ✓ | Parallel | Extension of the analysis of robotic flow shops to sequence-dependent setup time. |
| Current Research | ***S*** | ***M*** | Free | ✓ | ✓ | Parallel | A novel MILP model and an effective simulated annealing algorithm. Different speed machines at each stage and machine eligibility constraints. |

***SN***: Stage Number; ***MS***: Machine Number (at each stage); ***PCri***: Pickup Criterion;
***RMS***: Robot Move Sequence; ***PS***: Part Sequence; ***Sol-Proc***: Solution Procedure for *RMS* and *PS*.

## 2.3 Single Robot

This section addresses the robotic scheduling problem in blocking hybrid flow shop cells that consider multiple part-types, different speed parallel machines at each stage, machine eligibility constraints, and a single transportation robot to move the parts between stages.

### *2.3.1 Problem Statement and Mixed Integer Programming Model*

This section provides required definitions, notations, and basic properties for the BHFS-RCs in which the handling resource is a single gripper robot. Just as a simple flow shop robotic cell is analogous to a flow shop with blocking, a flow shop robotic cell with parallel machines is analogous to a flexible or hybrid flow shop with blocking. In a robotic flow shop cell with parallel machines, there are $J$ parts (where $j \in \{1, 2, \ldots, J\}$) that would be processed on $S$ stages. For each processing stage $s \in S$ there are $M_s \geq 1$parallel machines. As with simple cells, each part is successively processed at stages 1through $S$. A part can be processed at the $s^{th}$ stage by any machine from the set $SE_{j,s}$ that denotes the eligible set of machines for part $j$.

Figure 2.1 illustrates the considered robotic cell with parallel machines. The rectangles present the machines placed at each stage. The transporting robot moves on the disjunctive lines between the machines to load or unload them. The bold arcs illustrate the sequence of stages that the parts are processed (flow line system).

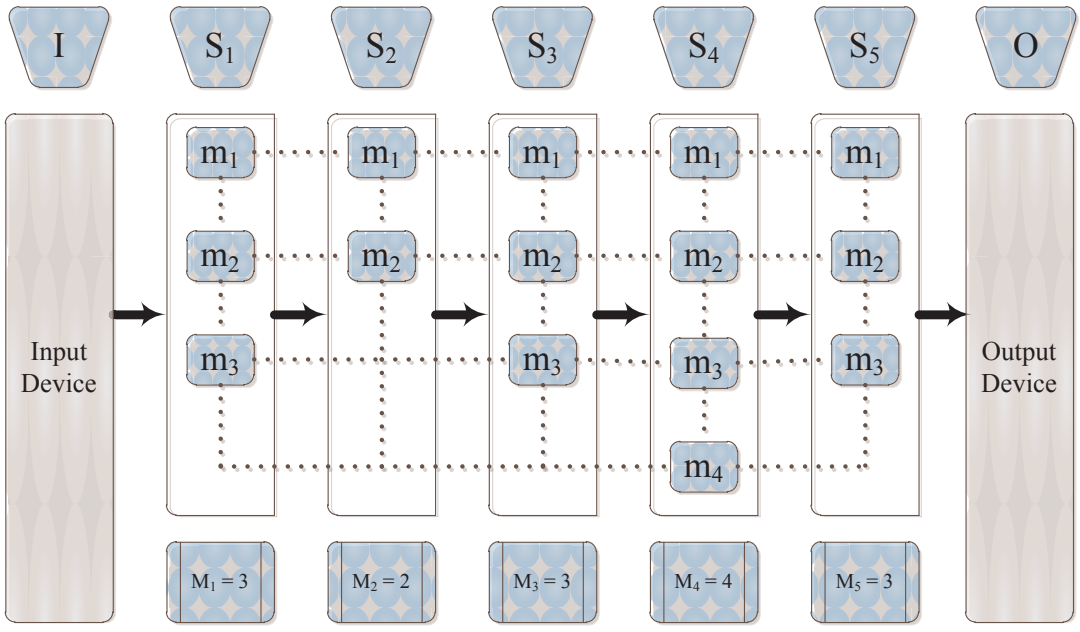



Figure 2.1 An example of a hybrid robotic cell with parallel machines considered in this research.

A mixed integer linear programming (MILP) model for the BHFS-RC scheduling problem has been proposed to minimize the makespan based on the completion times

of operations. Obviously, minimizing the makespan increases the throughput. The developed model is discussed in the following.

*Indices*

$J$ the number of parts, $j \in \{1,2,\dots,J\}$,

$S$ the number of stages considering input and output stages, $s \in \{1,2,\dots,S\}$,

$M_s$ the number of related machines at stage $s$, where input and output stages have just one machine, $m \in \{1,2,\dots,M_s\}$,

$W$ the number of operations that the robot is engaged with unloading, $f \in \{1,2,\dots,W\}$,

*Parameters*

$P_{j,s}$ the standard processing time of $j^{th}$ part at stage$s$, where the processing time of all parts at input and output stages are equal to zero,

$R_{s,m}$ the processing speed of $m^{\text{th}}$ machine at stage $s$, where the speed of the machine at input and output stages are zero,

$SE_{j,\text{s}}$ the set of eligible machines at stage $s$ which can process $j^{\text{th}}$ part,

SP the handling speed of the robot,

$BM$ a very large number, the traditional *"Big M"*.

*Decision variables*

$X_{j,j',s}$ equal to 1 if $j^{th}$ part is processed before $j'^{th}$ part on stage $s$, 0 otherwise,

$Z_{j,s,f}$ equal to 1 if part $j$ on stage $s$ is the $f^{th}$ operation that would be transferred by robot to the next stage,

$Y_{j,s,m}$ equal to 1 if machine $m$ processes part $j$ at stage $s$, 0 otherwise,

$D_{j,s}$ the depart time of $j^{th}$ part from stage $s$,

$C_{MAX}$ the time that all parts reach the output device.

*Objective function and constraints*

$Minimize: C_{MAX}$

$$D_{j,s} + BM \times \left(2 - \left(Y_{j,s,m} + Y_{j,s-1,m'}\right)\right) \geq D_{j,s-1} + \left(SP \times (|s - (s-1)| + |m - m'|)\right) + \left(P_{j,s} \times R_{s,m}\right)$$
$$j \in \{1,2,\dots,J\}; s \in \{2,\dots,S-1\}; m \in SE_{j,s}; m' \in SE_{j,s-1} \quad (2.2)$$

$$\left(\sum_{m \in SE_{j,s}} Y_{j,s,m}\right) = 1;$$
$$j \in \{1,2,\dots,J\}; s \in \{1,2,\dots,S\} \quad (2.3)$$

$$\left(\sum_{m}^{M_s} Y_{j,s,m}\right) = 1$$
$$j \in \{1,2,\dots,J\}; s \in \{1,2,\dots,S\} \quad (2.4)$$

$$\left(\sum_{f=1}^{W} Z_{j,s,f}\right) = 1$$
$$j \in \{1,2,\dots,J\}; s \in \{1,2,\dots,S-1\} \quad (2.5)$$

$$\left(\sum_{j=1}^{J}\sum_{s=1}^{S-1} Z_{j,s,f}\right) = 1$$
$$f \in \{1,2,\dots,W\} \quad (2.6)$$

$$\left(\sum_{f=1}^{f'} Z_{j,s,f}\right) - BM \times \left(1 - Z_{j,s-1,f'}\right) \leq 0$$
$$j \in \{1,2,\dots,J\}; s \in \{2,\dots,S-1\}; f' \in \{1,2,\dots,W\} \quad (2.7)$$

$$\left(\sum_{f=1}^{f'} Z_{j,s-1,f}\right) - BM \times \left(3 - Y_{j',s,m} - Y_{j,s,m} - Z_{j',s,f'} + X_{j,j',s}\right) \leq 0$$
$$j, j' \in \{1,2,\dots,J\}, j \neq j'; s \in \{2,\dots,S-1\}; m \in SE_{j,s}; f' \in \{1,2,\dots,W\} \quad (2.8)$$

$$\left(\sum_{f=1}^{f'} Z_{j',s-1,f}\right) - BM \times \left(4 - Y_{j',s,m} - Y_{j,s,m} - Z_{j,s,f'} - X_{j,j',s}\right) \leq 0 \quad (2.9)$$

$$j, j' \in \{1,2,\dots,J\}, j \neq j'; s \in \{2,\dots,S-1\}; m \in SE_{j',s}; f' \in \{1,2,\dots,W\}$$

$$\begin{aligned} D_{j,s} + \Big(BM \times \big(5 - Y_{j,s,m} - Y_{j',s',m'_1} - Y_{j',s'+1,m'_2} - Z_{j',s',f-1} - Z_{j,s,f}\big)\Big) \\ \geq D_{j',s'} \\ + \Big(SP \\ \times \big((|s' - (s'+1)| + |m'_1 - m'_2|) \\ + (|(s'+1) - s| + |m'_2 - m|)\big)\Big) \end{aligned} \tag{2.10}$$

$$j, j' \in \{1,2,\dots,J\}, j \neq j'; s, s' \in \{1,2,\dots,S-1\}; m'_1 \in SE_{j',s'}; m'_2 \in SE_{j',s'+1}; \quad m \in SE_{j,s}; f \in \{2,\dots,W\}$$

$$\begin{aligned} C_{MAX} + BM \times \big(2 - Y_{j,S-1,m} - Z_{j,S-1,W}\big) \\ \geq D_{j,S-1} + \big(SP \times (|S - (S-1)| + |1 - m|)\big) \end{aligned} \tag{2.11}$$

$$j \in \{1,2,\dots,J\}; m \in SE_{j,S-1}$$

$$X_{j,j',s}, Y_{j,s,m}, Z_{j,s,f} \in \{0,1\}$$

Constraint (2.2) states that a part can be processed at a stage if only processed at the previous stage. In another way, the depart time of a part from a stage should be equal to or bigger than the depart time from the previous stage plus the transportation and processing times at this stage. Constraints (2.3) and (2.4) ensure that a part is processed only on one of the eligible machines at each stage. Constraints (2.5) and (2.6) insure that the robot transfers just one part at a time. Constraint (2.7) states that a part is transferred to a stage if only transferred to the previous stage. Constraints (2.8) and (2.9) state that the sequence of operations processed on the same machine at a stage is determined according to the unloading sequence of the robot. Constraint (2.10) insures that no two unloading operations can be handled by the robot at the same time. This constraint also determines the sequence of unloading operations handled by the robot. Constraint (2.11) states that makespan is equal to or bigger than the depart time of the last part unloaded from the last stage plus the transportation time.

Since the proposed MILP model determines simultaneously the optimal sequence of parts and transportation operations, it differs from the existing models in literature. The total number of constraints and variables required by the model according to the size of the problem is estimated as in Table 2.2 assuming that the number of machines at each stage is equal to *M*. Due to the large number of decision variables and constraints required as the size of the problem increases, it is difficult to solve this model by exact methods for even medium-size problems. Hence, the Simulated Annealing (SA) algorithm is proposed in the next section to solve the problem using meta-heuristic search.

Table 2.2 Estimated number of constraints and variables for the proposed MILP model with respect to the number of parts, stages, and machines (J, S, M).

| ***Number of Constraints*** | ***Number of Variables*** |
|---|---|
| $\Big((J \times S)\big(1 + (J \times M)\big(2 + (S \times M^2)\big)\big) + M^2 + 4\Big)(J \times S)$ | $(J \times (10 \times S + 2) + 10 \times M + 4)(J \times S)$ |

### *2.3.2 The Proposed Simulated Annealing Algorithm*

Since the blocking hybrid flow shop scheduling (BHFS-RCS) is NP-hard, SA algorithm is proposed to solve the considered problem. SA is inspired by an analogy to annealing in solids. This idea was firstly propounded by Metropolis, Rosenbluth, Rosenbluth, Teller & Teller (1953) and Kirkpatrick & Vecchi (1983) applied it to combinatorial and other optimization problems. Often the solution space of an optimization problem has many local optima. A simple local search algorithm proceeds by choosing a random initial solution and generating a neighbor from that solution. The neighboring solution is accepted if it is a cost decreasing transition. Such a simple algorithm has the drawback of often converging to a local optimum. Though SA is a local search algorithm by itself, it avoids getting trapped in a local optimum by also accepting cost increasing neighbors with some probability. The performance of SA is influenced by some factors such as initial temperature ($T_0$), final temperature ($T_f$), number of temperature declines ($N$), number of iterations at each temperature (*Iter*), and cooling schedule (*CS*). In SA, first an initial solution is randomly generated, and a neighbor is found around the initial solution. This new

neighbor solution is accepted if it improves the value of cost function. SA uses intensification scheme in this manner to search the initial solution neighborhood efficiently. In order not to get trapped in a local optimum, SA has to use a diversification scheme and search all around the solution space. As a diversification scheme, in a condition where the value of cost function of a new neighbor solution is worse, SA accepts it if it satisfies the criterion $u < e^{-\frac{\Delta}{T}}$, where $u$ is generated using a uniform distribution between 0 and 1; $\Delta$ is the difference between the value of the cost function of the current solution and the new neighbor solution, and $T$ is the current temperature. Therefore, on slow reduction of temperature, the algorithm converges to the global optimum solution.

*2.3.2.1 Encoding the Initial Solution*

A feasible solution for the BHFS-RCS problem is a permutation of all unloading operations of parts, and thereby the length of the permutation is the total number of unloading operations. Each unloading operation includes transportation to the next stage and loading on a related machine at that stage. The operations are numbered and each operation's preceding operations are saved in $RX_{e,,i}$ matrix, where $e, i \in \{1,2,\dots,W\}$. Then each operation's preceding operations chain is defined by the first step of following algorithm. Thereupon, the operations sequence that will be unloaded by robot (illustrated by $\delta$) is generated through the remaining steps of the following algorithm.

**Begin**

$f \leftarrow 1$,

**If** $RX_{e,i} == 1$ and $RX_{u,e} == 1$**then**: $RX_{u,i} \leftarrow 1$,

**Repeat**

**For** each operation $i \in \{1,2,\dots,W\}$which is not included to the$\delta$,

**If** for any operation noted by $i$, $RX_{e,i} == 0$ for all operations $e \in \{1,2,\dots,W\}$ **then**:

Select the operation $i$ and add it to the *fth* position of the $\delta$,

$RX_{i,d} \leftarrow 0$ for all operations $d \in \{1,2,\dots,W\}$,

$f \leftarrow (f + 1)$,

**Until** $f == W + 1$,

**End.**

*2.3.2.2 Decoding Algorithm*

After the generation of solutions (initial solution or neighbor solution) through the encoding algorithm, decoding algorithm is applied to evaluate its objective function value (makespan). As mentioned previously, $P_{j,s}$, $R_{s,m}$, $SE_{j,s}$ are the input data required to find the makespan for each solution and $C_{MAX}$ is the time that all part reach the output device. Here some new terms used in the decoding algorithm are defined.

$SS_f$ The stage which the part at the $f^{th}$ position in the solution would be unloaded.

$FF_f$ The release time of the robot at the $f^{th}$ position of the solution.

$FS_f$ The last stage that the robot loaded the part at the $f^{th}$ position of the solution.

$FM_f$ The last machine that the robot loaded the part at the $f^{th}$ position of the solution.

$JM_{j,s}$ The machine which processes part *j* at stage *s*.

$PR_{s,m}$ The last part that visited machine *m* at stage *s*.

$SM_s$ Number of parts being simultaneously processed at stage *s* according to the generated solution.

Figure 2.2 illustrates the solution representation and the way to determining $SS_f$ at each position. The presented example contains four parts with one processing stage and two unloading operations: one from the input stage to the processing stage and the other from the processing stage to the output stage. It is clear that $SS_f$ gives the stage number of the part at the $f^{th}$ position of solution. This kind of solution representation helps to simultaneously determine the sequence of robot moves and the processing operations of parts. In the figure, the position $(f = 3)$ states that the part number $(\delta(3) = 4)$ would be unloaded from stage $(SS_3 = 2)$, which is the processing stage, and handled and loaded on stage $(SS_3 + 1 = 3)$, which is the output stage.

| $\delta_j$ | $j_4$ | $j_1$ | $j_4$ | $j_3$ | $j_2$ | $j_1$ | $j_2$ | $j_3$ |
|---|---|---|---|---|---|---|---|---|
| $f$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| $SS_f$ | 1 | 1 | 2 | 1 | 1 | 2 | 2 | 2 |

$SS_1 = 1, SS_3 = 2, SS_5 = 1$

Figure 2.2 Solution representation of BHFS-RCS problem.

The proposed Decoding algorithm is in fact a constructive algorithm that finds the best eligible machine at each stage to perform the processing operation on each part. In the following, for each candidate machine the release time of the machine plus the processing time of the part on the machine is computed. The minimum value among the computed values for each candidate machine is assigned to variable(temp), and by this way the best eligible machine for the processing of the part at the next stage and the completion time of the part on this machine is determined.

- $temp \leftarrow \min_{\{m \in E_{j,SS_f+1}\}} \left\{ D_{PR_{SS_f+1,m},SS_f+1} + \left( P_{j,SS_f+1} \times R_{SS_f+1,m} \right) \right\}$ , //

  *Determining the best machine at stage* $(SS_f + 1)$ *that part* $(j)$ *would be loaded on it* //
- $JM_{j,SS_f+1} \leftarrow m$, // *Machine that gives the* $(temp)$ //

There may be a point in time that all machines of a stage are occupied and any other part at the previous stage cannot be transferred to this stage before any one of the machines is released. The mentioned state causes the stage to be blocked. Hence, to rectify the blocking state the robot should unload at least one part from the next stage before transporting any of the parts from the previous stage. During the encoding procedure or generation of new neighbor solutions there may be blocking states that should be rectified before computing the makespan. Proposed *Rectifying algorithm* is provided below and Figure 2.3 illustrates the related flowchart.

**Rectifying blocking states:**

$f \leftarrow 0$

**Repeat**

$f \leftarrow f + 1,$

Determine the part $(j)$ in position $(f)$ of the solution, and the related stage $(SS_f)$ that would be unloaded from,

**If** $\left(SM_{SS_f+1} < E_{\delta(f),SS_f+1}\right)$ **then:**

$SM_{SS_f} \leftarrow SM_{SS_f} - 1$

$SM_{SS_f+1} \leftarrow SM_{SS_f+1} + 1$

**Else If** $\left(SM_{SS_f+1} \geq E_{\delta(f),SS_f+1}\right)$ and $\left(SS_f + 1 \neq S\right)$ **then:** // *If all eligible machines for part* $\delta(f)$ *are occupied*//

$g \leftarrow f$

**Repeat**

$g \leftarrow g + 1$

Determine the stage $(k)$ that the part at position $(g)$ would be unloaded from,

**Until** $\left(k == SS_f + 1\right)$ // *Until finding the position that releases one of machines of the next stage* //

Insert the part at position $(g)$ before position $(f)$ in the solution,

$f \leftarrow f - 1,$

**Until** $(f == W)$

**End.**

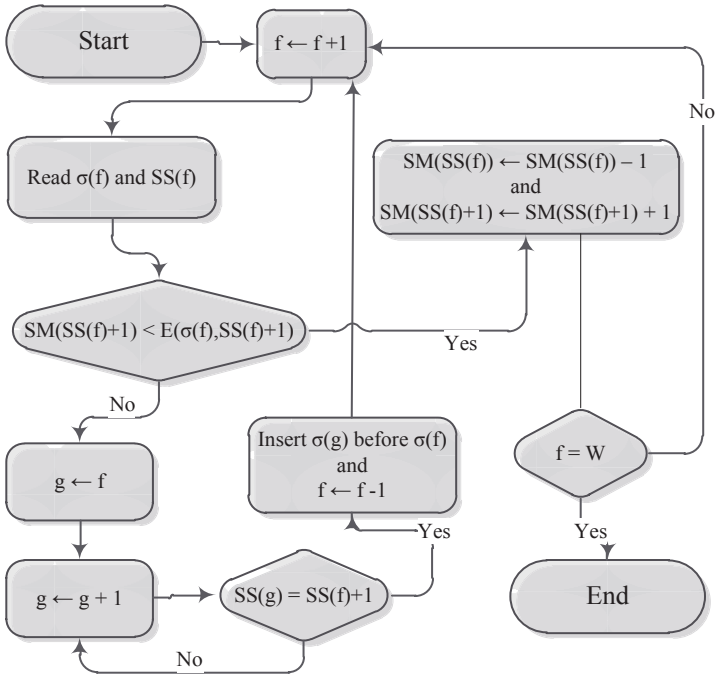



Figure 2.3 Flow chart of the proposed Rectifying algorithm.

Proposed *Decoding algorithm*:

**Begin**

**Initializing:**

Let $\left(PR_{s,m} \leftarrow \emptyset\right)$, $\left(FF_f \leftarrow \emptyset\right)$, $\left(FS_f \leftarrow \emptyset\right)$*and* $\left(FM_f \leftarrow \emptyset\right)$, //*Initializing parameters*//

Let $(f \leftarrow 1)$, // *Robot's first operation that would be unloaded* //

**Repeat**

**Rectifying blocking states** // Previously presented //

$j \leftarrow \delta(f)$,

**If**$\left(SS_f == 1\right)$ **then**:

$temp \leftarrow \min_{\{m \in E_{j,2}\}} \left\{D_{PR_{2,m},2} + \left(P_{j,2} \times R_{2,m}\right)\right\}$, // *Determining the best machine at stage* $\left(SS_f + 1\right)$ *that part* $(j)$ *would be loaded on* //

$JM_{j,2} \leftarrow m$,// *Machine that minimizes* $(temp)$//

$$D_{j,1} \leftarrow max\left\{\begin{matrix} D_{PR_{2,JM_{j,2}},2} - SP \times \left(1 + \left|1 - JM_{j,2}\right|\right) \\ and \\ FF_{f-1} + SP \times \left(\left|FS_{f-1} - 1\right| + \left|FM_{f-1} - JM_{j,1}\right|\right) \end{matrix}\right\},$$ // *Finding the depart time of part* $(j)$ *from stage* $\left(SS_f\right)$ //

**Else if**$(SS_f > 1)$ **then**:

$temp \leftarrow \min_{\{m \in E_{j,SS_f+1}\}} \left\{ D_{PR_{SS_f+1,m},SS_f+1} + \left( P_{j,SS_f+1} \times R_{SS_f+1,m} \right) \right\}$ //

*Determining the best machine at stage* $(SS_f + 1)$ *that part* $(j)$ *would be loaded on* //

$JM_{j,SS_f+1} \leftarrow m,$ // *Machine that minimizes* $(temp)$ //

$D_{j,SS_f} \leftarrow$

$$max \left\{ \begin{array}{c} D_{PR_{SS_f+1,JM_{j,SS_f+1}},SS_f+1} - SP \times \left(1 + \left| JM_{j,SS_f} - JM_{j,SS_f+1} \right| \right) \\ and \\ D_{j,SS_f-1} + SP \times \left(1 + \left| JM_{j,SS_f} - JM_{j,SS_f-1} \right| \right) + \left( P_{j,SS_f} \times R_{SS_f,JM_{j,SS_f}} \right) \\ and \\ FF_{f-1} + SP \times \left( \left| FS_{f-1} - SS_f \right| + \left| FM_{f-1} - JM_{j,SS_f} \right| \right) \end{array} \right\} //$$

*Finding the depart time of part* $(j)$ *from stage* $(SS_f)$ //

**Fixing parameters**:

$PR_{SS_f+1,JM_{j,SS_f+1}} \leftarrow j,$ // *The last part that visited machine* $\left(JM_{j,SS_f+1}\right)$ *at stage* $(SS_f + 1)$ //

$FF_f \leftarrow \left( D_{j,SS_f} + SP \times \left(1 + \left| JM_{j,SS_f} - JM_{j,SS_f+1} \right| \right) \right),$ // *Release time of robot at the position*$(f)$ *of the solution* //

$FS_f \leftarrow SS_f + 1,$ // *The last stage that robot visited at the position* $(f)$ *of solution* //

$FM_f \leftarrow JM_{j,SS_f+1},$ // *The last machine that robot visited at the position*$(f)$ *of solution* //

$f \leftarrow f + 1,$ // *Robot's next operation (transfer)* //

**Until**$(f == W),$ // *Maximum number of operations that the robot should unload is reached* //

**End**.

Figure 2.4 provides the general flowchart of the proposed Decoding algorithm.

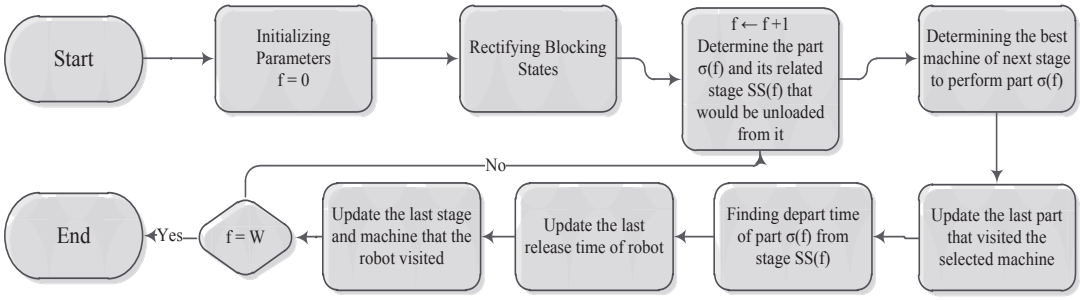


Figure 2.4 General flowchart of the proposed Decoding algorithm.

The robot is indeed a machine that performs transfer operations on parts. The BHFS-RCS problem can be considered as a grid graph where the nodes represent the processing and transfer operations on the parts. Figure 2.5 gives a graph representation of a feasible solution for scheduling six parts in a BHFS-RC with four stages and two machines at each stage. The hexagon shapes represent the transfer operations of each part which include unloading from the previous stage and loading on a machine at the next stage. The dashed arrows illustrate the sequence of these transfer operations. Obviously, the key component of a feasible solution is the critical path which is the longest path from the start node to the end node in the grid graph and the length of the critical path gives the makespan. The red arrows present the critical path in Figure 2.5. The hexagon shapes represent the transfer of a part to the next stage by the robot. In order the robot to perform a transfer operation, both the processing of the part should be completed at the previous stage and the transfer of the previously handled part by the robot should be completed. The blocking conditions of machines are also considered; the robot cannot transfer any part to any machine that has not been released. Any operation on the critical path is called a critical operation. It is also possible to decompose the critical path into a number of blocks. A block is a maximal sequence of adjacent critical operations that are required to be processed on the same machine or the transfer operations performed by the robot.

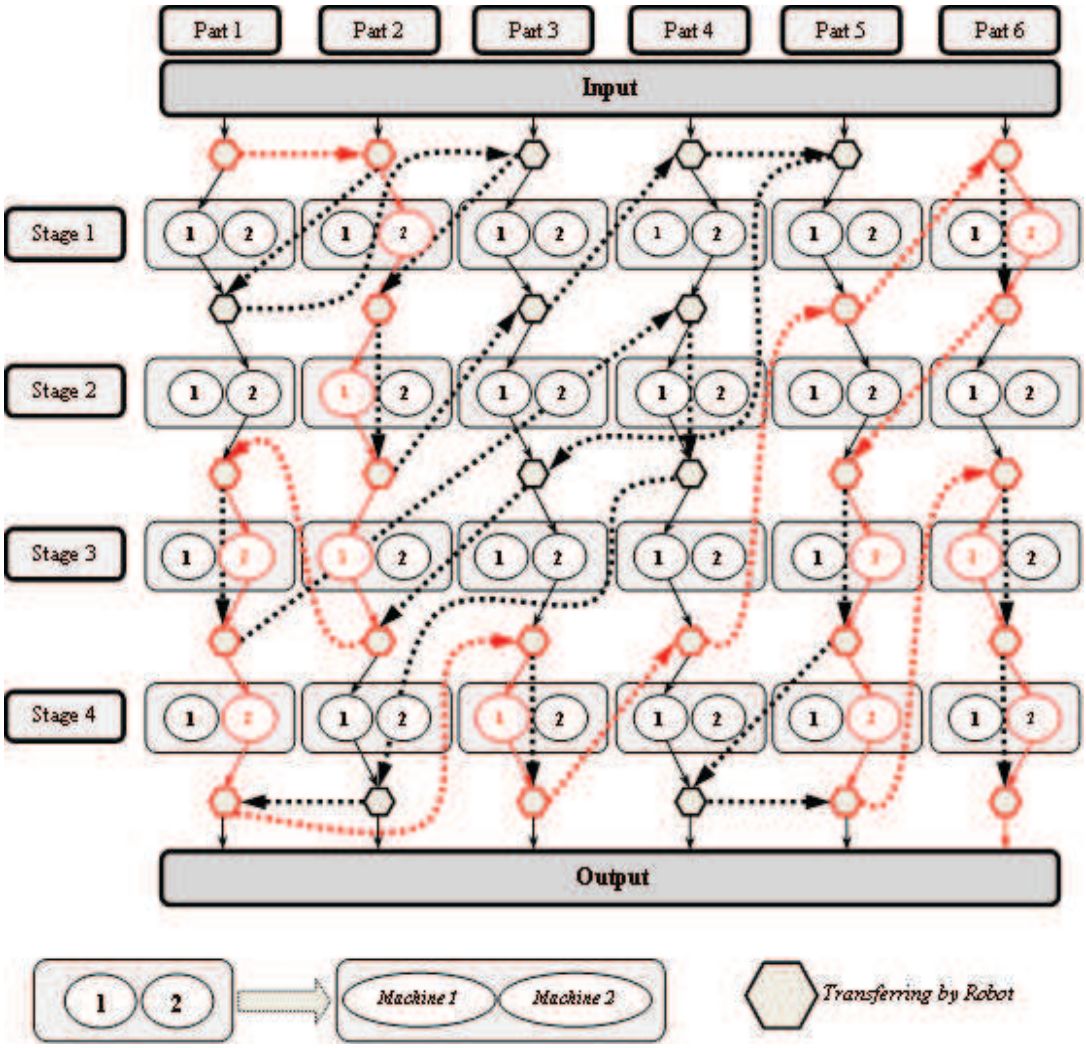



Figure 2.5 Graph representation of BHFS-RCS problem with six parts and four stages containing two machines.

The extracted critical path from Figure 2.5 has been given in Table 2.3 where critical operations including processing and transportation operations and their related machines (whether a machine or robot) are illustrated.

Table 2.3 The extracted critical path for the example given in Figure 2.5.

| Number | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| Operation Part/Stage ($P_p$, $S_s$) | $P_1,S_I$ | $P_2,S_I$ | $P_2,S_1$ | $P_2,S_2$ | $P_2,S_3$ | $P_1,S_2$ | $P_1,S_3$ | $P_1,S_4$ | $P_3,S_3$ | $P_3,S_4$ |
| Machine or Robot ($M_m$, $S_s$ or R) | R | R | $M_1,S_2$ | $M_2,S_1$ | $M_3,S_1$ | R | $M_3,S_2$ | $M_4,S_2$ | R | $M_4,S_1$ |

| Number | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
|---|---|---|---|---|---|---|---|---|---|---|
| Operation Part/Stage ($P_p$, $S_s$) | $P_4,S_3$ | $P_5,S_1$ | $P_6,S_I$ | $P_6,S_1$ | $P_5,S_2$ | $P_5,S_3$ | $P_5,S_4$ | $P_6,S_2$ | $P_6,S_3$ | $P_6,S_4$ |
| Machine or Robot ($M_m$, $S_s$ or R) | R | R | R | $M_1,S_2$ | R | $M_3,S_2$ | $M_4,S_2$ | R | $M_3,S_1$ | $M_4,S_2$ |

*2.3.2.3 The Neighborhood Structure*

A neighborhood structure is a mechanism which can obtain a new set of neighbor solutions by applying a small perturbation to a given solution (Glover & Laguna, 1999). Each neighbor solution is reached immediately from a given solution by a move. The efficiency of block properties based neighborhood generation has been studied for job shop and flow shop scheduling problems in the literature (Grabowski & Wodecki, 2004; Solimanpur, Vrat & Shankar, 2004; Nowicki & Smutnicki, 2005; B. Liu, Wang & Jin, 2008; Elmi, Solimanpur, Topaloglu & Elmi, 2011). Neighborhood structures and move evaluation directly affect the efficiency of local search for the BHFS-RCS problem and unnecessary moves must be eliminated if it is possible. Furthermore, Ruiz & Vázquez-Rodríguez (2010) stated that the properties of graph representation in Hybrid Flow Shop Scheduling problem, implied by critical block theory, can be used to speed up the search procedure.

In this study two SA algorithms that use different neighborhood structures are proposed. The first algorithm uses simple insertion method (Figure 2.6), namely $SA^{IN}$, whereas the other SA uses block properties based insertion method (Figure 2.7), namely $SA^{BP}$. Figure 2.6 illustrates the simple insertion method, whereas the extended insertion method using block properties is illustrated in Figure 2.7. The critical path and blocks are taken from Table 2.3. For example, operations no.11-13 form a block, which are the transfer operations of the robot, and the block properties

based insertion method is performed considering these operations as shown in Figure 2.7.

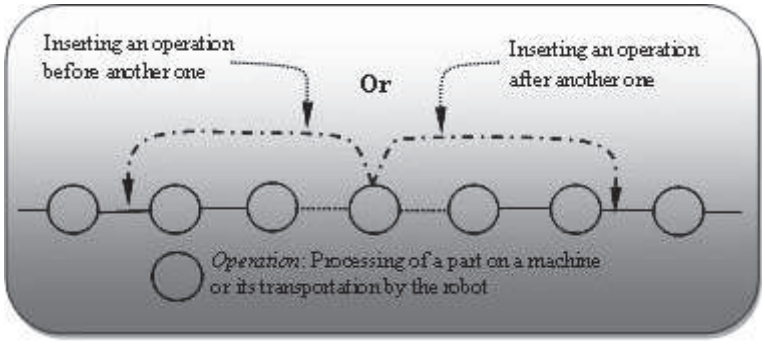



Figure 2.6 Generating neighborhood structure via insertion method.

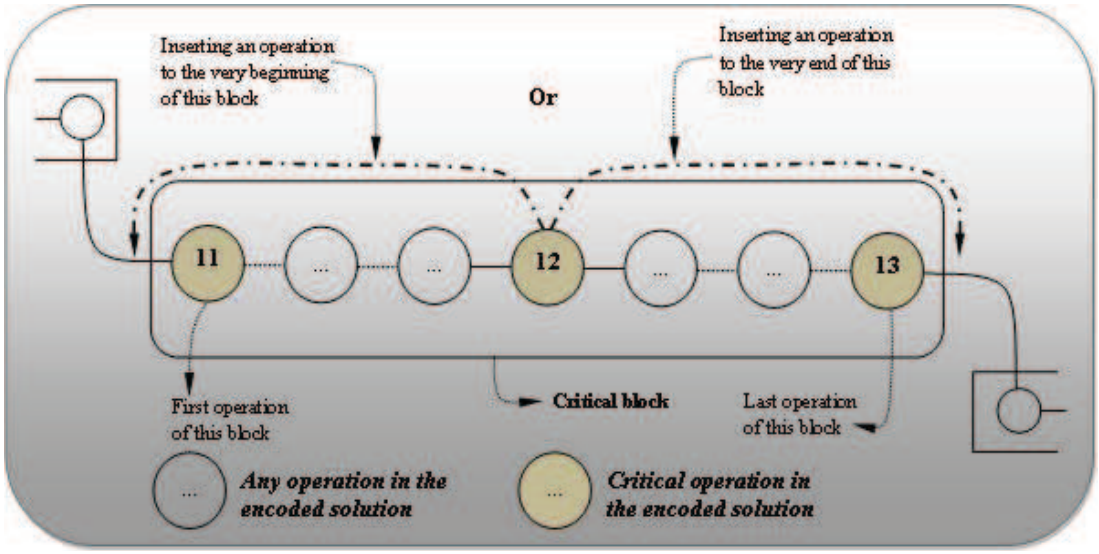



Figure 2.7 Generating neighborhood structure via block properties based insertion method (done on critical operations no.11-13 mentioned in Table 2.3).

The block properties based neighborhood search can only be applicable in SA or Tabu Search. For this reason, the SA algorithm is used for the BHFS-RC scheduling problem to search more efficiently and speedily.

The following terms are used in the proposed SA algorithms:

| | |
|---|---|
| *Best-solution* | The solution with the minimum makespan during the search procedure. |
| *Current-solution* | Current solution whose neighborhood is going to be searched. |
| *Neighbor-solution* | The solution generated by a perturbation from the Current-solution. |
| *OBJ-best* | Objective function value of Best-solution. |
| *OBJ-current* | Objective function value of Current-solution. |
| *OBJ-neighbor* | Objective function value of Neighbor-solution. |
| $P_{acc}$ | Probability of acceptance. |
| $U$ | Uniformly distributed random number between 0 and 1. |
| $T_0$ , $T_f$, $T_q$ | Initial, final, and current temperatures, respectively. |
| $FR$ | Freeze counter ( used to denote frozen state). |
| $N$ | The number of times temperature is decreased. |
| $Iter$ | Maximum number of solutions generated at each temperature. |
| $CS$ | Cooling schedule rate. |

The steps of proposed SA are provided below.

**Begin**

***Initialization***

Get $T_0$, $T_f$, $N$, $Iter$, and $CS$,

Generate initial solution ($Current_solution$) through *encoding algorithm*,

Calculate $OBJ_current$ using *Decoding algorithm*,

$Best_solution \leftarrow Current_solution$,

$OBJ_best \leftarrow OBJ_current$,

$T \leftarrow T_0, \quad FR \leftarrow 0, \quad q \leftarrow 0,$

**Repeat**

***Improvement step***

Set $I \leftarrow 0, \quad ac \leftarrow 0,$

**Repeat**

Generate $Neighbor_solution$ through $Current_solution$,

$I \leftarrow (I + 1)$,

Calculate *OBJ-neighbor* using *Decoding algorithm*,

$\Delta \leftarrow (OBJ_neighbor \ - OBJ_current/OBJ_current) \times 100$,

**If** $(\Delta < 0)$ **then**:

$ac \leftarrow (ac + 1)$,

$Current_solution \leftarrow Neighbor_solution$,

$OBJ_current \leftarrow OBJ_neighbor$,

**If** $OBJ_neighbor < OBJ_best$ **then**:

$Best_solution \leftarrow Neighbor_solution$,

$OBJ_best \leftarrow OBJ_neighbor$,

$f \leftarrow 0$,

**Else**

$P_{acc} \leftarrow \left(e^{-\frac{\Delta}{T_q}}\right)$,

Generate $U$ between $(0, 1)$,

**If** $(U < P_{acc})$ **then**:

$ac \leftarrow (ac + 1)$,

$Current_solution \leftarrow Neighbor_solution$,

$OBJ_current \leftarrow OBJ_neighbor$,

**Until** $(I == Iter)$.

***Annealing***

$q \leftarrow (q + 1)$,

Degenerate $(T_q)$ using *CS,*

$P_{acc} \leftarrow \left(\frac{ac}{Iter}\right)$,

**If** $(P_{acc} \leq 0.15)$ **then**:

$f \leftarrow (f + 1)$,

**Until**$(f == FR \ || \ T_q == T_f)$,

**Return** $Best_solution \ \&\& \ OBJ_best$,

**End**.

### *2.3.3 Computational Results*

In order to assess the empirical performance of the proposed SA algorithms, the computational experiments have been carried out on a set of randomly generated instances. The problem sizes are determined by varying the levels of three parameters, which are respectively the number of part types, the number of stages, and the number of machines at each stage. The number of parts varies at four levels: 10, 20, and 40. There are also five levels for the number of stages$\{2, 4, 6, 8, 10\}$, through which parts have to pass and be processed. The set of eligible machines $SE_{j,s}$ is generated randomly for each part and stage. For the first two levels of part numbers, 10 and 20, three parallel machines are assumed at each stage. Whereas, for 40 parts the number of parallel machines is assumed to vary at two levels: 3 and 5. The combination of above mentioned parameter levels generates 20 different problem categories.

As previously mentioned various parameters affect the performance of the SA algorithm. The aforementioned problem categories have been divided into 4 groups due to the solution length ($Sol_{length}$) which is shown in Table 2.4.

The values assigned to the parameters are determinant in getting high efficiency. The used parameter values for the proposed SA are reported in Table 2.5 for each group of problems.

Moreover, for each problem category 50 problem instances have been generated. Thus, a total of 1000 test problems are yielded and used in the experiment. For each part, an integer standard processing time at every stage are generated from the uniform distribution in [1, 100]. The machines at each stage operate at three levels of processing speed: 1, 2 and 3. It means that each part's standard processing time is multiplied by the processing speed of the eligible machine.

Table 2.4 Problems categorized into four groups.

| Part | Stage | Groups of problems (Categorized by $Sol_{Length} = J \times S$) | | | | |
|---|---|---|---|---|---|---|
| | | 30 - 90 | 100 - 160 | 170 - 230 | 230 - 290 | 300 - 440 |
| 10 | 2 | ■ | | | | |
| | 4 | ■ | | | | |
| | 6 | ■ | | | | |
| | 8 | ■ | | | | |
| | 10 | | ■ | | | |
| 20 | 2 | ■ | | | | |
| | 4 | | ■ | | | |
| | 6 | | ■ | | | |
| | 8 | | | ■ | | |
| | 10 | | | ■ | | |
| 40 | 2 | | ■ | | | |
| | 4 | | | ■ | | |
| | 6 | | | | ■ | |
| | 8 | | | | | ■ |
| | 10 | | | | | ■ |

Table 2.5 Preferred levels of SA parameters for each group of problems separately

| SA Parameters | Groups of problems | | | | |
|---|---|---|---|---|---|
| | 30 - 90 | 100 - 160 | 170 - 230 | 230 - 290 | 300 - 440 |
| N | 75 | 80 | 120 | 200 | 250 |
| Iter | 60 | 90 | 125 | 150 | 200 |
| $T_0$ | 20 | 20 | 30 | 40 | 40 |
| CS | Exponential | Exponential | Hyperbolic | Hyperbolic | Hyperbolic |

Furthermore, to achieve a consistent evaluation of the performance of proposed $SA^{BP}$ and $SA^{IN}$ algorithms and to reduce the effect of their stochastic nature, all problems have been solved 10 times. The SA algorithms for the BHFS-RCS problem were coded in C#.NET and run on a Pentium 4, 100 GHz PC. Table 2.6 presents the average makespan of each category obtained by the SA algorithms. The percentage of reduction in average makespan (*GAP*) for $SA^{BP}$ compared to $SA^{IN}$ is also calculated as follows and these *GAP* values are illustrated in Table 2.6.

$$GAP = \left(\frac{SA^{IN}_{Makespan} - SA^{BP}_{Makespan}}{SA^{IN}_{Makespan}}\right) \times 100 \qquad (2.12)$$

Table 2.6 Average makespan obtained by proposed $SA^{BP}$ and $SA^{IN}$ algorithms.

| Part | Stage | Machine | Makespan | | CPU Time | | GAP % | NO |
|---|---|---|---|---|---|---|---|---|
| | | | $SA^{BP}$ | $SA^{IN}$ | $SA^{BP}$ | $SA^{IN}$ | | |
| 10 | 2 | 3 | 991.41 | 1128.87 | 2.38 | 2.35 | 12.1768 | 1 |
| | 4 | 3 | 1621.92 | 1848.72 | 4.15 | 3.61 | 12.2679 | 2 |
| | 6 | 3 | 2316.64 | 2575.31 | 7.54 | 6.26 | 10.0442 | 3 |
| | 8 | 3 | 3064.28 | 3388.38 | 11.54 | 8.81 | 9.5650 | 4 |
| | 10 | 3 | 3818.65 | 4198.84 | 16.64 | 11.1 | 9.0546 | 5 |
| 20 | 2 | 3 | 2117.16 | 2517.28 | 6.26 | 4.85 | 15.8949 | 6 |
| | 4 | 3 | 3242.65 | 3726.13 | 18.33 | 12.19 | 12.9754 | 7 |
| | 6 | 3 | 4340.31 | 4699.52 | 39.87 | 25.73 | 7.6435 | 8 |
| | 8 | 3 | 6055.21 | 6370.22 | 83.77 | 45.95 | 4.9450 | 9 |
| | 10 | 3 | 6711.57 | 7448.79 | 117.33 | 84.11 | 9.8972 | 10 |
| 40 | 2 | 3 | 3725.12 | 4201.53 | 51.97 | 28.15 | 11.3390 | 11 |
| | | 5 | 3831.12 | 4598.15 | 52.27 | 27.18 | 16.6813 | 12 |
| | 4 | 3 | 6388.21 | 7181.11 | 274.12 | 194.59 | 11.0415 | 13 |
| | | 5 | 7044.01 | 7326.12 | 289.11 | 213.75 | 3.8507 | 14 |
| | 6 | 3 | 8371.85 | 9082.58 | 512.71 | 429.94 | 7.8252 | 15 |
| | | 5 | 10124.18 | 10987.35 | 952.82 | 728.51 | 7.8560 | 16 |
| | 8 | 3 | 12496.41 | 12850.91 | 1071.68 | 939.07 | 2.7586 | 17 |
| | | 5 | 13479.85 | 13989.67 | 1198.15 | 981.47 | 3.6443 | 18 |
| | 10 | 3 | 14540.83 | 15172.54 | 1288.37 | 1053.95 | 4.1635 | 19 |
| | | 5 | 15248.62 | 15893.66 | 1367.31 | 1196.47 | 4.0585 | 20 |
| **Avg** | | | | | | | **8.884** | |

Due to the stochastic nature of the SA algorithm, it may lead to a different solution in each run. Therefore, *GAP* can be treated as a random variable and the data reported in the last column of Table 2.6 can be considered as a sample from this variable. Let $\mu$ denote the mean value of *GAP*.

The aforementioned data is used to test the following hypothesis:

$$H_0 : \mu \geq 10.25$$
$$H_1 : \mu < 10.25 \qquad (2.13)$$

Let us consider probability of type II error as $\beta = 0.1$ for$\mu = 5$. For a $1 - \alpha = 0.95$ confidence, it can be realized from statistics that a sample of size $n = 20$ is sufficient to test this hypothesis. The value of *t* statistic is obtained by the following equation:

$$t = \frac{\sqrt{n}\,(\overline{GAP} - 10.25)}{S} \tag{2.14}$$

Where, $\overline{GAP}$ and $S$ stand for sample mean and standard deviation, respectively. These parameters are calculated as $\overline{GAP} = 8.884$ and $S = 4.088$ for data reported in Table 2.6 and hence $t = -1.419$. Since $t_{0.05,19} = -1.729 < -1.494$, the hypothesis $H_0$ is accepted and it can be claimed with 95% confidence that the proposed $SA^{BP}$ algorithm has reduced the average makespan of $SA^{IN}$ algorithm by 10.25 percent.

Furthermore, to investigate the efficiency of $SA^{BP}$ and compare with the obtained optimal solutions from the proposed MILP model, small size problems are generated. The size of each problem, in terms of total number of parts and stages, has been reported in Table 2.7. The number of machines at each stage for these problems is taken as two. The proposed MILP model has been coded and run via GAMS. The gap value ($GAP_{OPT}$) this time in Table 2.7 demonstrates the percentage of deviation of the mean and best makespan values obtained by $SA^{BP}$ from the optimal value obtained by GAMS. The proposed $SA^{BP}$ found the optimal solution for the first four test problems. For the last set of problems (Num. 5- 6) GAMS failed to provide the optimal solution even after 120 hours. The best objective function values found within this time period are included in the table with an asterisk, and accordingly the proposed $SA^{BP}$ algorithm produced 21.08% and 62.81% better quality solutions in rather short computing times respectively for these problems.

Table 2.7 Computational results for the small size problems solved by $SA^{BP}$ and GAMS .

| Part | Stage | Gams-CPLEX | | SA | | | $GAP_{OPT}$ | |
|---|---|---|---|---|---|---|---|---|
| | | Makespan | Time | Makespan | | $Time^{Mean}$ | Mean | Best |
| | | | | Mean | Best | | | |
| 5 | 2 | 323 | 46:41.714 | 325.8 | 323 | 00:0.706 | 0.86 | 0 |
| | 3 | 514 | 67:44.965 | 527.3 | 514 | 00:1.128 | 2.58 | 0 |
| 6 | 2 | 385 | 86:43.452 | 392.8 | 385 | 00:0.834 | 2.02 | 0 |
| | 3 | 617 | 101:50.847 | 631.7 | 617 | 00:1.241 | 2.38 | 0 |
| 7 | 2 | *574** | *120:00.000* | 467.4 | 453 | 00:1.265 | -18.57 | -21.08 |
| | 3 | *1724** | *120:00.000* | 662.4 | 641 | 00:1.757 | -61.57 | -62.81 |

* the obtained solution after 120 minutes, which is not optimal.

### *2.3.4 Conclusion*

This research introduces the Blocking Hybrid Flow Shop Robotic Cell Scheduling (BHFS-RCS) problem with a single transportation robot. The problem incorporates additional real world constraints such as machines with different processing speeds at each stage, machine eligibility constraints and transportation constraints. Initially, a mixed-integer linear programming model (MILP) is developed that simultaneously addresses the problem of determining the best sequence for the robot moves and for the processing of parts. This is a new MILP model that minimizes the makespan for the hybrid flow shop scheduling problem that also considers the transportation constraints. Since the problem of interest is NP-hard, a simulated annealing (SA) based algorithm is proposed to heuristically solve the developed model. Since the sequencing of both processing operations and unload operations influence the makespan, they should be done simultaneously for an efficient optimization. Hence, this study for the first time considers sequencing of these operations in parallel for the solution of the BHFS-RCS problem.

The $SA^{IN}$ using simple insertion method and the $SA^{BP}$ using block based insertion method have been tested on 20 problem categories using 1000 randomly generated problem instances. The $SA^{BP}$ algorithm has been shown to be a more successful heuristic in solving the BHFS-RCS problem. According to the computational results, the $SA^{BP}$ algorithm reduces the average makespan by 10.25 percent more over the $SA^{IN}$ algorithm. Furthermore, six small size problems have been generated to investigate the performance of proposed $SA^{BP}$ over the optimum values obtained from the MILP model. The $SA^{BP}$ has found the optimum solution for the first four problems and also has produced better quality solutions in rather short computing times for the remaining ones compared to the MILP model.

## 2.4 Multiple Robots

In this section we provide required definitions, notations, and basic properties for BHFS-RC with multiple single gripper robots.

### *2.4.1 Problem Statement and Mixed Integer Programming Model*

In BHFS-RCS, there are $J$ parts (where $j \in \{1, 2, \ldots, J\}$) that would be processed on $S$ stages. For each processing stage $s \in S$ there are $M_s \geq 1$ identical (or unrelated) machines. As with simple cells, each part is successively processed at stages 1 through $S$. A part can be processed at the $s^{th}$ stage by any one of the $E_{j,s}$ number of eligible machines at that stage.

Figure 2.8 illustrates the considered BHFS-RCS problem and unrelated parallel machines at each stage. In the considered cell, rectangles present the machines placed at each stage. Here the transporting robots move on the disjunctive lines between machines to load or unload them. The bold arcs illustrate the sequence of stages that the parts would be processed (flow line system).

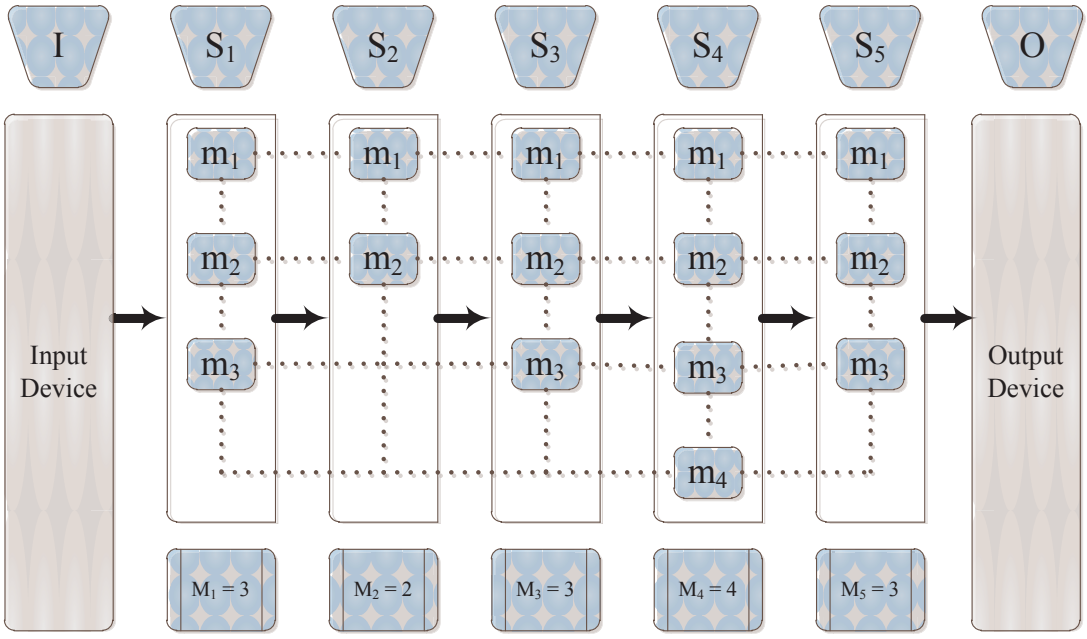



Figure 2.8 An example of BHFS-RCS problem considered in this research.

We formulated a MILP model for the BHFS-RCS problem to minimize the makespan based on the completion times of operations. Obviously, minimizing the makespan would increase the throughput sufficiently. The developed model is discussed below.

*Indices*

$J$ the number of parts where, $j \in \{1,2,\dots,J\}$,

$S$ the number of stages considering input and output stages where, $s \in \{1,2,\dots,S\}$,

$M_s$ the number of unrelated machines at stage $s$, where input and output stages have just one machine, $m \in \{1,2,\dots,M_s\}$,

$R$ the number of single gripper robots where, $r \in \{2,\dots,R\}$,

$W$ the number of operations that the robot is engaged with unloading; this operation also includes transportation to the next stage and loading on a related machine at that stage ( the operations belong to input stage and other processing stages, excluding output stage) where,
$f \in \{1,2,\dots,W\}$ and $W = J \times (S+1)$.

*Parameters*

$P_{j,s}$ the standard processing time for the *jth* part at stage $s$, where the processing times of all parts at input and output stages are equal to zero,

$R_{s,m}$ the processing speed of the *mth* machine at stage $s$, where the speed of the machines at input and output stages is zero,

$E_{j,s}$ the number of eligible machines that can process the *jth* part at stage $s$, $m \in \{1,2,\dots,E_{j,s}\}$

$SP$ the time required for the robot to travel each distance,

$BM$ a very large number, the traditional "Big M".

*Decision variables*

$X_{j,j',s}$ equal to 1 if the *jth* part is processed before the $j'th$ part at stage $s$, 0 otherwise,

$Z_{j,s,f}$ equal to 1 if part $j$ at stage $s$ is the *fth* operation that would be unloaded by any robot to be transported to the next stage,

$Y_{j,s,m}$ equal to 1 if machine $m$ processes part $j$ at stage $s$, 0 otherwise,

$RS_{r,f}$ equal to 1 if the *rth* robot unloads the *fth* operation, 0 otherwise,

$D_{j,s}$ the variable that determines the depart time of the *jth* part from stage $s$,

$C_{MAX}$ the time that all parts reach the output device.

*Objective function and constraints*

$$Minimize: C_{MAX}$$

$$D_{j,s} + BM \times \left(2 - \left(Y_{j,s,m} + Y_{j,s-1,m'}\right)\right) \geq D_{j,s-1} + \left(SP \times (|s-(s-1)| + |m-m'|)\right) + \left(P_{j,s} \times R_{s,m}\right) \quad (2.15)$$

$j \in \{1,2,\dots,J\}$; $s \in \{2,\dots,S-1\}$; $m \in \{1,2,\dots,E_{j,s}\}$; $m' \in \{1,2,\dots,E_{j,s-1}\}$

$$\left(\sum_{m=1}^{E_{j,s}} Y_{j,s,m}\right) = 1 \quad (2.16)$$

$j \in \{1,2,\dots,J\}$; $s \in \{1,2,\dots,S\}$

$$\left(\sum_{m=E_{j,s+1}}^{M_s} Y_{j,s,m}\right) = 0 \quad (2.7)$$

$j \in \{1,2,\dots,J\}$; $s \in \{1,2,\dots,S\}$

$$D_{j',s-1} + BM \times \left(4 - Y_{j,s,m} - Y_{j',s,m} - Y_{j',s-1,m'} - X_{j,j',s}\right) \geq D_{j,s} + \left(SP \times (1 + |m-m'|)\right) \quad (2.18)$$

$j, j' \in \{1,2,\dots,J\}, j \neq j'$; $s \in \{2,\dots,S-1\}$; $m \in \{1,2,\dots,E_{j,s}\}$; $m' \in \{1,2,\dots,E_{j',s-1}\}$

$$D_{j,s-1} + BM \times \left(3 - Y_{j,s,m} - Y_{j',s,m} - Y_{j,s-1,m'} + X_{j,j',s}\right) \geq D_{j',s} - \left(SP \times (1 + |m-m'|)\right) \quad (2.19)$$

$j, j' \in \{1,2,\dots,J\}, j \neq j'$; $s \in \{2,\dots,S-1\}$; $m \in \{1,2,\dots,E_{j,s}\}$; $m' \in \{1,2,\dots,E_{j,s-1}\}$

$$\left(\sum_{f=1}^{W} Z_{j,s,f}\right) = 1 \quad (2.20)$$

$j \in \{1,2,\dots,J\};\ s \in \{1,2,\dots,S-1\}$

$$\left(\sum_{j=1}^{J}\sum_{s=1}^{S-1} Z_{j,s,f}\right) = 1 \qquad (2.21)$$

$f \in \{1,2,\dots,W\}$

$$\left(\sum_{f=1}^{f'} Z_{j,s,f}\right) - BM \times \left(1 - Z_{j,s-1,f'}\right) \leq 0 \qquad (2.22)$$

$j \in \{1,2,\dots,J\};\ s \in \{2,\dots,S-1\};\ f' \in \{1,2,\dots,W\}$

$$\left(\sum_{f=1}^{f'} Z_{j,s-1,f}\right) - BM \times \left(3 - Y_{j',s,m} - Y_{j,s,m} - Z_{j',s,f'} + X_{j,j',s}\right) \leq 0 \qquad (2.23)$$

$j, j' \in \{1,2,\dots,J\}, j \neq j';\ s \in \{2,\dots,S-1\};\ m \in \{1,2,\dots,E_{j,s}\};\ f' \in \{1,2,\dots,W\}$

$$\left(\sum_{f=1}^{f'} Z_{j',s-1,f}\right) - BM \times \left(4 - Y_{j',s,m} - Y_{j,s,m} - Z_{j,s,f'} - X_{j,j',s}\right) \leq 0 \qquad (2.24)$$

$j, j' \in \{1,2,\dots,J\}, j \neq j';\ s \in \{2,\dots,S-1\};\ m \in \{1,2,\dots,E_{j',s}\};\ f' \in \{1,2,\dots,W\}$

$$\sum_{r=1}^{R} RS_{r,f} = 1 \qquad (2.25)$$

$f \in \{1,2,\dots,W\}$

$$
\begin{aligned}
D_{j,s} + \Big(BM \times \big(7 - RS_{r,f} - RS_{r,f'} - Z_{j,s,f} - Z_{j',s',f'} - Y_{j,s,m} - Y_{j',s',m'_1} \\
- Y_{j',s'+1,m'_2}\big)\Big) \\
\geq D_{j',s'} + SP \\
\times \big((|s' - (s'+1)| + |m'_1 - m'_2|) \\
+ (|(s'+1) - s| + |m'_2 - m|)\big)
\end{aligned}
\tag{2.26}
$$

$j, j' \in \{1,2,\dots,J\}, j \neq j'$; $s, s' \in \{1,2,\dots,S-1\}$; $m'_1 \in \{1,2,\dots,E_{j',s'}\}$;
$m'_2 \in \{1,2,\dots,E_{j',s'+1}\}$; $m \in \{1,2,\dots,E_{j,s}\}$; $f, f' \in \{2,\dots,W\}, f' < f$;
$r \in \{2,\dots,R\}$

$$
\begin{aligned}
C_{MAX} + BM \times \big(1 - Y_{j,S-1,m}\big) \\
\geq D_{j,S-1} + \big(SP \times (|S - (S-1)| + |1 - m|)\big)
\end{aligned}
\tag{2.27}
$$

$j \in \{1,2,\dots,J\}$; $m \in \{1,2,\dots,E_{j,S-1}\}$

Constraint (2.15) states that each part can be processed at a stage if it has been already processed at the previous stage. Constraints (2.16) and (2.17) ensure that each part can be processed on just one of eligible machines at each stage. Constraints (2.18) and (2.19) state the sequence of part processing operations on the same machine at each stage. These constraints guarantee that no two part operations are processed on a machine at the same time and determine the partial sequence of related parts for each stage. Constraints (2.20) and (2.21) ensure that each robot can transport just one part at any time. Constraint (2.22) states that each part can be transported to the next stage if it has been transported to the previous stage already. Constraints (2.23) and (2.24) maintain that the sequence of robot moves should be determined according to the sequence of part processing operations on the same machine at any stage. These constraints guarantee that any part would be loaded on a machine after unloading the previous part from that machine. Constraint (2.25) prevents assigning more than one robot for each unload operation. Constraint (2.26) imposes that no two unload operations are handled by the same robot at any time. Finally, constraint (2.27) states that makespan is larger than or equal to the depart time of all parts unloaded from the last stage plus transportation time.

Due to the large number of decision variables and constraints of the proposed model, it is difficult to solve this model by exact methods for even small-size problems. Hence, a simulated annealing (SA) based solution approach is developed in the next section to solve the problem.

### *2.4.2 The Proposed Simulated Annealing Algorithm*

As mentioned in the literature, the hybrid flow shop scheduling is NP-hard problem and thus the BHFS-RCS is NP-hard as well. Therefore, a SA algorithm is proposed to solve the considered problem meta-heuristically.

SA is inspired by an analogy to annealing in solids. This idea was firstly propounded by Metropolis, et al. (1953) and Kirkpatrick & Vecchi (1983) applied it to combinatorial and other optimization problems. Often the solution space of an optimization problem has many local optima. A simple local search algorithm proceeds by choosing a random initial solution and generating a neighbor from that solution. The neighboring solution is accepted if it is a cost decreasing transition. Such a simple algorithm has the drawback of often converging to a local optimum. Though SA is a local search algorithm by itself, it avoids getting trapped in a local optimum by also accepting cost increasing neighbors with some probability. The performance of SA is influenced by some factors such as number of temperature declines ($N$), number of iterations at each temperature ($Iter$), initial temperature ($T_0$), final temperature ($T_f$), and cooling schedule ($CS$). In SA, first an initial solution is randomly generated, and a neighbor is found around the initial solution. This new neighbor solution is accepted if it improves the value of cost function. SA uses intensification scheme in this manner to search the initial solution neighborhood efficiently. In order not to get trapped in a local optimum, SA has to use a diversification scheme and search all around the solution space. As a diversification scheme, in a condition where the value of cost function of a new neighbor solution is worse, SA accepts it if it satisfies the following criterion:

$$u < e^{-\frac{\Delta}{T}}, \tag{2.28}$$

where $u$ is generated using a uniform distribution between 0 and 1; $\Delta$ is the difference between the value of the cost function of the new neighbor solution and the current solution, and $T$ is the current temperature. Therefore, on slow reduction of temperature, the algorithm converges to the global optimum solution.

*2.4.2.1 Encoding the Initial Solution*

A feasible solution for the BHFS-RCS problem is a permutation of all unload operations of all parts. Each unload operation also includes transportation to the next stage and loading on a related machine at that stage. This kind of representation uses a permutation in which unload operations are sequenced to specify in which order they are performed. The length of this representation is the total number of operations to be unloaded by robots. Operations are numbered and each operation's preceding operations are saved in $RX_{e,i}$ matrix where $e, i \in \{1,2,\dots,W\}$. Then each operation's preceding operations chain is defined by the first step of the following algorithm. Thereupon, the operations sequence (illustrated by $\delta$) that will be unloaded by the robots is generated through the following algorithm.

**Begin**

$f \leftarrow 1$,

**If** $RX_{e,i} == 1$ and $RX_{u,e} == 1$ **then**: $RX_{u,i} \leftarrow 1$,

**Repeat**

**For** each operation $i \in \{1,2,\dots,W\}$ which is not included to the $\delta$,

**If** for any operation noted by $i$, $RX_{e,i} == 0$ for all operations $e \in \{1,2,\dots,W\}$ **then**:

Select the operation $i$ and add it to the *fth* position of the $\delta$,

$RX_{i,d} \leftarrow 0$ for all operations $d \in \{1,2,\dots,W\}$,

$f \leftarrow (f + 1)$,

**Until** $f == J + 1$,

**End.**

*2.4.2.2 Rectifying Blocking States*

When all machines of a stage are occupied, any other part at the previous stage cannot be transported to this stage before any of the machines are released. Therefore, the remaining part at the previous stage causes the related machine to be blocked. Hence, to rectify the blocking states robots should unload at least one part from the next stage before transporting any of the parts from the previous stage. During the encoding procedure or generation of new neighbor solutions there may be blocking states that should be rectified before computing the makespan.

*2.4.2.3 Decoding Algorithm*

After the generation of solutions (initial solution or neighbor solution) through the encoding algorithm, decoding algorithm is applied to evaluate its objective function value (makespan). As mentioned previously in Section 2.2, $P_{j,s}$, $R_{s,m}$, and $E_{j,s}$ are the input data required to evaluate the makespan for each solution and $C_{MAX}$ is the time that all parts reach the output device. Here we define some new terms which are used in Decoding algorithm.

| | |
|---|---|
| $SS_f$ | The stage at which the part at the $f^{th}$ position in the solution would be processed. |
| $JM_{j,s}$ | The machine which processes part $j$ at stage $s$. |
| $PR_{s,m}$ | The last part that has visited machine $m$ at stage $s$. |
| $SM_s$ | Number of parts being simultaneously processed at stage $s$ according to the generated solution. |
| $TR_f$ | The robot which unloads the part at the $f^{th}$ position. |
| $RLS_r$ | The last stage that robot number $r$ has visited. |
| $RLM_r$ | The last machine that robot number $r$ has visited. |
| $RFR_r$ | The last release time of $r^{th}$ robot. |

Figure 2.9 illustrates the solution representation and the way to determining $SS_f$ at each position. The presented example contains 4 parts that all have 2 unload

operations: one from the input device to the processing stage and the other from the processing stage to the output device. It is clear that $SS_f$ mentions the stage number of the part at the $f^{th}$ position of solution. This kind of solution representation helps to determine the sequence of robots moves and the operation sequence of parts in parallel.

The proposed Decoding algorithm is based on the depart time of parts from each stage. In the BHFS-RCS problem, the depart time of parts from each stage is equal to their unload times from the referred stage by any robot. As empty moves of robots are considered in the Decoding algorithm, they are not required to be considered in the generated solution.

In Figure 2.9, the position $(f = 3)$ states that the part number $(\delta(3) = 4)$ would be unloaded from stage $(SS_3 = 2)$, handled and loaded on stage $(SS_3 + 1 = 3)$. It can also be seen from Figure 2.9 that the last stage is not taken into account since it is an output stage and the time that any part reaches this stage can be determined through the depart time of the referred part from the previous stage.

| $\delta_j$ | $j_4$ | $j_1$ | $j_4$ | $j_3$ | $j_2$ | $j_1$ | $j_2$ | $j_3$ |
|---|---|---|---|---|---|---|---|---|
| $f$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| $SS_f$ | 1 | 1 | 2 | 1 | 1 | 2 | 2 | 2 |

$SS_1 = 1, SS_3 = 2, SS_5 = 1$

Figure 2.9 Solution representation of BHFS-RCS problem.

The proposed Decoding algorithm is in fact a constructive algorithm that finds the best machine at each stage to perform the needed processing operation of each part. Also, it is capable of determining the most applicable robot for the operation. In the following Decoding algorithm, variable $vr_m$ is the minimum completion time for the processing of a part at a production stage among all the eligible machines.

Therefore, the machine with $vr_m$ is selected to perform the needed operation on the part.

- $vr_m \leftarrow \min_{\{m \in E_{j,SS_f+1}\}} \left\{ D_{PR_{SS_f+1,m},SS_f+1} + \left( P_{j,SS_f+1} \times R_{SS_f+1,m} \right) \right\}$ , //

  *Determining the best eligible machine at stage* $(SS_f + 1)$ *that completes the related processing operation fastest among other machines and that part* $(j)$ *would be loaded on* //
- $JM_{j,SS_f+1} \leftarrow m,$ // *Machine that minimizes the temp* //

Since the input device is considered at the first stage as a machine with zero processing time, $JM_{j,1} = 1$ for all parts.

Similarly, variable $vr_r$ computes the minimum time among all the robots to reach the position that related unload operation should be handled. Therefore, the earliest time that related unload operation could be started is determined and the robot with the earliest reaching time is selected to unload.

- $vr_r \leftarrow min_{\{r \in R\}} \left\{ RFR_r + SP \times \left( \left| RLS_r - SS_f \right| + \left| RLM_r - JM_{j,SS_f} \right| \right) \right\},$ //

  *Determining the most applicable robot that can unload part* $(j)$ //
- $RFR_r \leftarrow D_{j,SS_f} + SP \times \left( 1 + \left| JM_{j,SS_f} - JM_{j,SS_f+1} \right| \right)$, // *Saving the last release time of the rth robot* //
- $RLS_r \leftarrow SS_f + 1$ && $RLM_r \leftarrow JM_{j,SS_f+1}$, // *Saving the last stage and machine that the rth robot visited* //

Proposed *Rectifying algorithm* is provided in the following and Figure 2.10 illustrates its flowchart.

**Rectifying blocking states:**

$f \leftarrow 0$

**Repeat**

$f \leftarrow f + 1$,

Determine the part $(j)$ in position $(f)$ of the solution, and its related stage $(SS_f)$ that would be unloaded from,

**If** $\left(SM_{SS_f+1} < E_{\delta(f),SS_f+1}\right)$ **then:**

$SM_{SS_f} \leftarrow SM_{SS_f} - 1$

$SM_{SS_f+1} \leftarrow SM_{SS_f+1} + 1$

**Else If** $\left(SM_{SS_f+1} \geq E_{\delta(f),SS_f+1}\right)$ and $\left(SS_f + 1 \neq S\right)$ **then:** // *If all eligible machines for part* $\delta(f)$ *are occupied* //

$g \leftarrow f$

**Repeat**

$g \leftarrow g + 1$

Determine the stage $(k)$ that the part at position $(g)$ would be unloaded from,

**Until** $\left(k == SS_f + 1\right)$ // *Until finding the position that releases one of machines of the next stage* //

Insert the part at position $(g)$ before position $(f)$ in the solution,

$f \leftarrow f - 1$,

**Until** $(f == W)$

**End.**

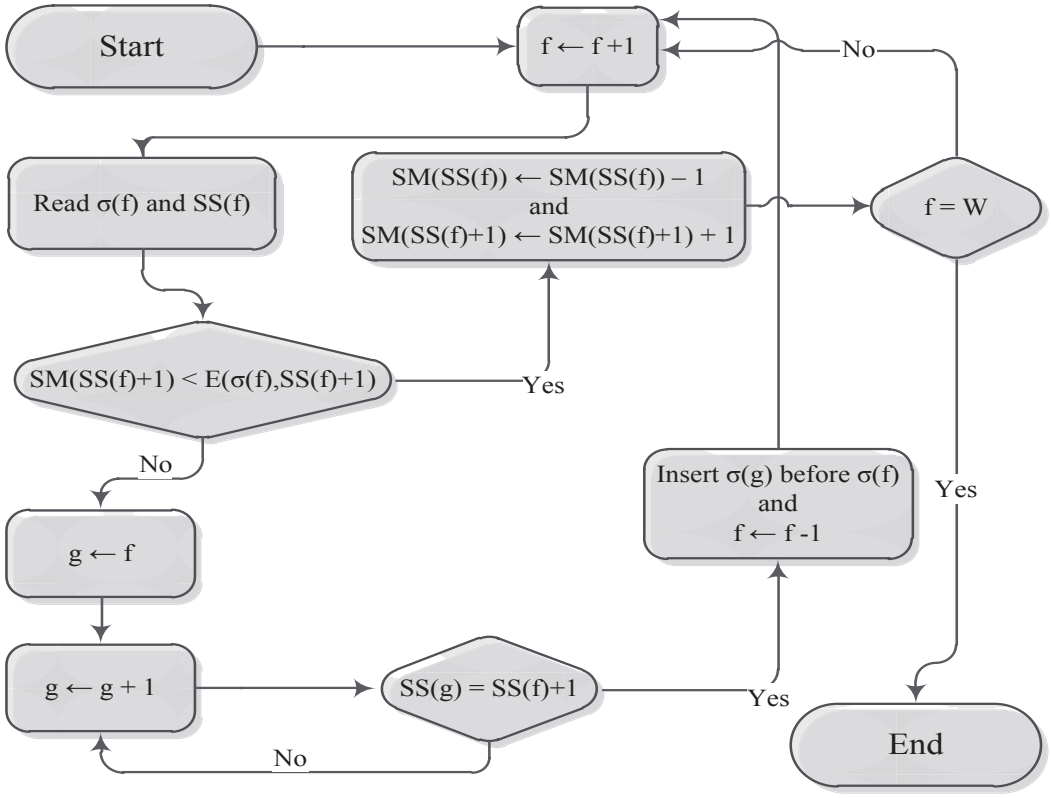



Figure 2.10 Flowchart of the proposed Rectifying algorithm

Proposed *Decoding algorithm*:

**Begin**

**Initializing:**

**Let** $PR_{s,m} \leftarrow \emptyset$ , $RLS_r \leftarrow \emptyset$ , $RLM_r \leftarrow \emptyset$ , $RLR_r \leftarrow 0$ , $TR_f \leftarrow \emptyset$ , $D_{j,s} \leftarrow 0$, and $JM_{j,1} \leftarrow 1$ **for all** $j, s, m, r, and\ f$,

**Let** $(f \leftarrow 1)$, // *The first operation that would be unloaded* //

**Repeat**

**Rectifying blocking states** // Previously presented //

$j \leftarrow \delta(f)$,

**If** $(SS_f == 1)$ **then**:

1) $vr_m \leftarrow \min_{\{m \in E_{j,2}\}} \{D_{PR_{2,m},2} + (P_{j,2} \times R_{2,m})\}$, // *Determining the best machine at stage* $(SS_f + 1 = 2)$ *that part* $(j)$ *would be loaded on it* //

2) $JM_{j,2} \leftarrow m$, $PR_{2,JM_{j,2}} \leftarrow j$ // *Assigning the machine that minimizes the* $(vr_m)$ and saving *the last part that visited machine* $\left(JM_{j,SS_f+1}\right)$ *at stage* $\left(SS_f + 1 = 2\right)$ //

3) $vr_r \leftarrow min_{\{r \in R\}}\{RFR_r + SP \times (|RLS_r - 1| + |RLM_r - 1|)\}$, // *Determining the most applicable robot that can unload part* $(j)$ //

4) $TR_f \leftarrow r$,

5) $D_{j,1} \leftarrow max\left\{D_{PR_{2,JM_{j,2}},2} - SP \times \left(1 + \left|1 - JM_{j,2}\right|\right) \;\&\&\; vr_r\right\}$, // *Finding the depart time of part* $(j)$ *from stage* $\left(SS_f\right)$ //

6) $RFR_{TR_f} \leftarrow D_{j,1} + SP \times \left(1 + \left|1 - JM_{j,2}\right|\right)$, // *Saving the last release time of robot* $TR_f$ //

7) $RLS_{TR_f} \leftarrow 2$ && $RLM_{TR_f} \leftarrow JM_{j,2}$, // *Saving the last stage and machine that the selected robot visited*//

**Else if** $\left(SS_f > 1\right)$ **then**:

1) $vr_m \leftarrow \min_{\left\{m \in E_{j,SS_f+1}\right\}}\left\{D_{PR_{SS_f+1,m},SS_f+1} + \left(P_{j,SS_f+1} \times R_{SS_f+1,m}\right)\right\}$, // *Determining the best machine at stage* $\left(SS_f + 1\right)$ *that part* $(j)$ *would be loaded on* //

2) $JM_{j,SS_f+1} \leftarrow m$ , $PR_{SS_f+1,JM_{j,SS_f+1}} \leftarrow j$ // *Assigning the machine that minimizes the* $(vr_m)$ and saving *the last part that visited machine* $\left(JM_{j,SS_f+1}\right)$ *at stage* $\left(SS_f + 1\right)$//

3) $vr_r \leftarrow min_{\{r \in R\}}\left\{RFR_r + SP \times \left(\left|RLS_r - SS_f\right| + \left|RLM_r - JM_{j,SS_f}\right|\right)\right\}$, // *Determining the most applicable robot that can unload part* $(j)$ //

4) $TR_f \leftarrow r$,

5) $D_{j,SS_f} \leftarrow$

$max\left\{D_{PR_{SS_f+1,m},SS_f+1} - SP \times \left(1 + \left|JM_{j,SS_f} - JM_{j,SS_f+1}\right|\right) \;\&\&\; D_{j,SS_f-1} + SP \times \left(1 + \left|JM_{j,SS_f-1} - JM_{j,SS_f}\right|\right) + \left(P_{j,SS_f} \times R_{SS_f,JM_{j,SS_f}}\right) \;\&\&\; vr_r\right\}$,//

*Finding the depart time of part* $(j)$ *from stage* $\left(SS_f\right)$ //

6) $RFR_{TR_f} \leftarrow D_{j,SS_f} + SP \times \left(1 + \left|JM_{j,SS_f} - JM_{j,SS_f+1}\right|\right)$, // *Saving the last release time of selected robot*//

7) $RLS_{TR_f} \leftarrow SS_f + 1$ && $RLM_{TR_f} \leftarrow JM_{j,SS_f+1}$, // *Saving the last stage and machine that the selected robot visited* //

$f \leftarrow (f + 1)$, // *The next handling operation* //

**Until** $(f == W)$, // *Maximum number of robots handling operation is reached* //

**End**.

Figure 2.11 provides the general flowchart of the proposed Decoding algorithm.

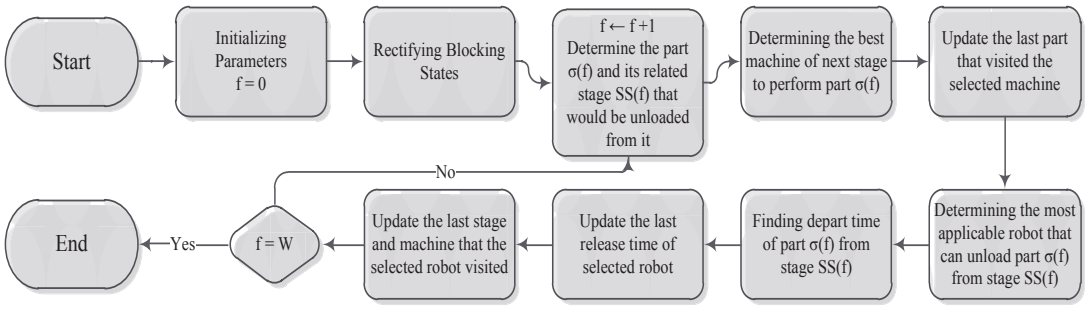



Figure 2.11 General flowchart of the proposed Decoding algorithm.

*2.4.2.4 The Neighborhood Structure*

Robots are machines that perform operations (transporting) on parts. For this reason, the BHFS-RCS problem would be considered as a grid graph where nodes represent all the unload and processing operations performed on parts.

Obviously, a key component of a feasible solution is the critical path, which is the longest path from the start node to the end node in the grid graph, and its length represents the makespan. Any operation on the critical path is called a critical operation. It is also possible to decompose the critical path into a number of blocks. A block is a maximal sequence of adjacent critical operations that require to be processed on the same machine or need to be handled, unloaded, or loaded by the same robot. A neighborhood structure is a mechanism which can obtain a new set of neighbor solutions by applying a small perturbation to a given solution (Glover &

Laguna, 1999). Each neighbor solution is reached immediately from a given solution by a move. Neighborhood structures and move evaluation directly affect the efficiency of local search for the BHFS-RCS problem, hence unnecessary moves must be eliminated if it is possible.

In this study, we generate the neighborhood structure using the insertion method based on the block concept. All operations that will be inserted are selected from the blocks and inserted at the starting or ending position of the related block. Figure 2.12 illustrates the extended insertion method to generate the neighborhood.

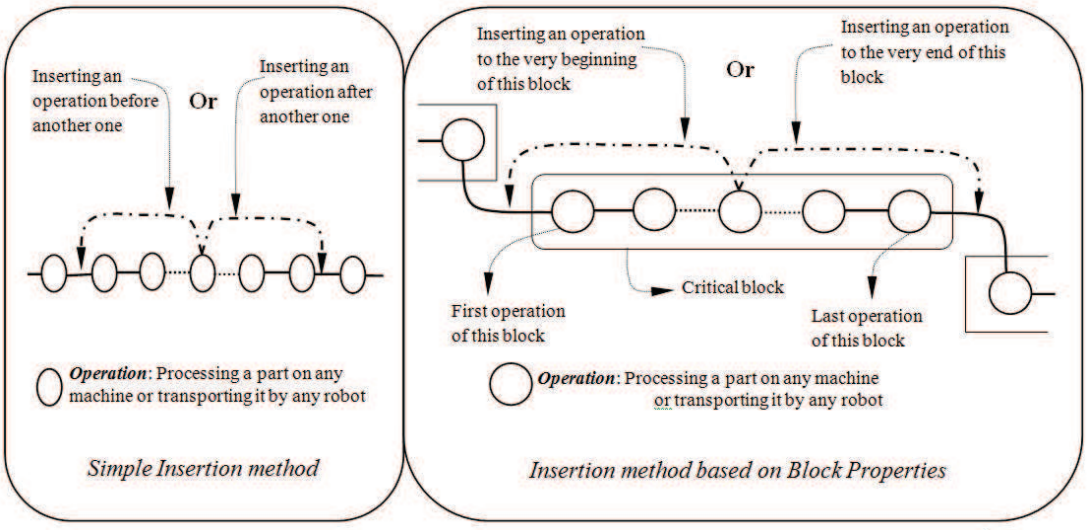



Figure 2.12 Generating neighborhood structure via insertion method based on block properties.

Figure 2.13 presents the graph representation of an example by 6 parts on 2 stages where there are 2 machines at each stage. The relevant cell contains 2 single gripper robots to unload, transport, and load the parts. Figure 2.13 illustrates the sequence of operations that any of the multiple robots (2 robots) unload and handle its referred part to the next stage. The hexagon shapes present the transportation operations; containing unloading from the previous stage, transporting and loading on the next stage, which should be done on each part. The dashed arrows (red and blue colors), present the sequence of transportation operations for each robot separately. As it is understandable from Figure 2.13, these operations (hexagon shapes) have two precedence operations; the processing of the related part at the previous stage should

be completed and also the selected suitable robot should be released from transporting the previous part.

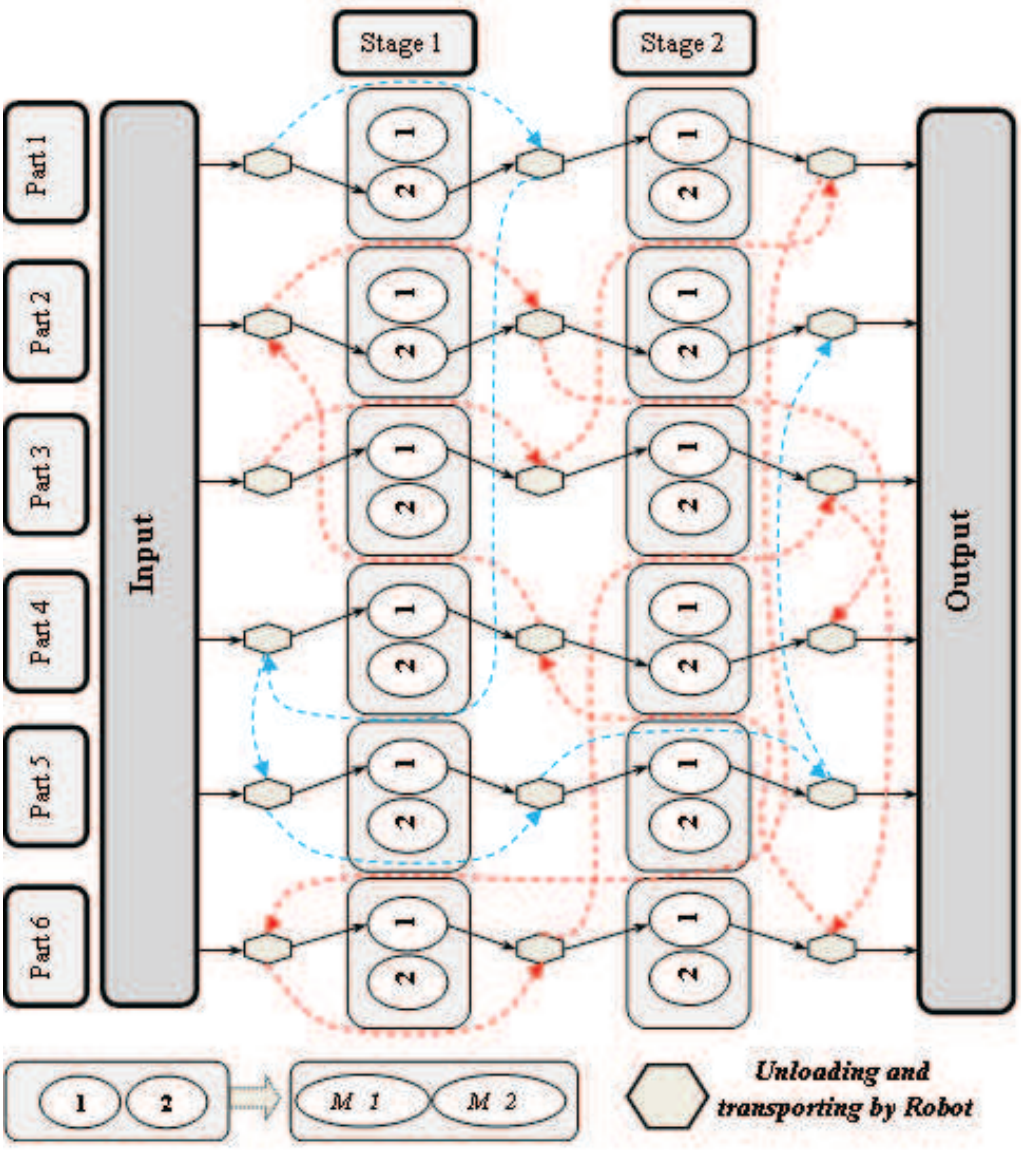



Figure 2.13 Graph representation of BHFS-RCS problem with 6 parts and 2 stages containing 2 machines.

It should also be realized that the blocking conditions of machines are considered in this example as well since robots did not transport any part to any machine which has not been released previously by any robot. All machines are released by robots before loading any other part on them.

*2.4.2.5 Cooling Schedule*

The cooling schedule affects the behavior of SA by controlling temperature degeneration. To avoid getting trapped in local minimum, solutions with worse objective values are accepted probably to diversify the search procedure. This type of solutions are accepted depending on the temperature generated by the cooling schedule. There are three kinds of cooling schedules in the literature (Lundy & Mees, 1986) and Table 2.8 illustrates these cooling schedule rates.

Table 2.8 Cooling schedule rates.

| | |
|---|---|
| Linear cooling (LiC) | $T_q \leftarrow \left(T_0 - q \times \frac{T_0 - T_f}{N}\right)$ |
| Exponential cooling (ExC) | $T_q \leftarrow \left(\frac{(T_0 - T_f)(N+1)}{N} + \left(T_0 - \frac{(T_0 - T_f)(N+1)}{N}\right)\right)$ |
| Hyperbolic cooling (HyC) | $T_q \leftarrow \left(0.5 \times (T_0 - T_f) \times \left(1 - \tanh\left(\frac{10 \times q}{N} - 5\right)\right)\right) + T_f$ |
| | $q = 1, \dots, N$ |

In this study the SA algorithm is proposed to solve the BHFS-RCS problem. The proposed algorithm uses insertion method based on block properties (Figure 2.12) to generate new neighbor solutions which is named $SA^{BP}$. Ruiz & Vázquez-Rodríguez (2010) stated that the properties of graph representation in Hybrid Flow Shop Scheduling problem, implied by critical block theory can be used to speed up the search procedure. This kind of neighborhood search can be applicable either in SA or Tabu Search algorithms. Therefore, the SA algorithm is used for BHFS-RCS problem to search more efficiently and speedily.

The following terms are used in the proposed SA algorithms:

| | |
|---|---|
| *Best-solution* | The solution with the minimum makespan during the search procedure. |
| *Current-solution* | Current solution whose neighborhood is going to be searched. |
| *Neighbor-solution* | The solution generated by a perturbation from the Current-solution. |
| *OBJ-best* | Objective function value of Best-solution. |
| *OBJ-current* | Objective function value of Current-solution. |
| *OBJ-neighbor* | Objective function value of Neighbor-solution. |
| $P_{acc}$ | Probability of acceptance. |
| *U* | Uniformly distributed random number between 0 and 1. |
| $T_0$ , $T_f$, $T_q$ | Initial, final, and current temperature, respectively. |
| *FR* | Freeze counter (is used to denote frozen state). |
| *N* | The number of times temperature is decreased. |
| *Iter* | Maximum number of solutions generated at each temperature. |
| *CS* | Cooling schedule rate. |

The steps of proposed SA are provided in the following.

---

**Begin**

***Initialization***

Get $T_0$, $T_f$, $N$, $Iter$, and *CS,*

Generate initial solution ($Current_solution$) through *encoding algorithm*,

Calculate $OBJ_current$ using *Decoding algorithm*,

$Best_solution \leftarrow Current_solution$,

$OBJ_best \leftarrow OBJ_current$,

$T \leftarrow T_0$, $FR \leftarrow 0$, $q \leftarrow 0$,

**Repeat**

***Improvement step***

Set $I \leftarrow 0$, $ac \leftarrow 0$,

**Repeat**

Generate $Neighbor_solution$ through $Current_solution$,

$I \leftarrow (I + 1)$,

Calculate *OBJ-neighbor* using *Decoding algorithm*,

$\Delta \leftarrow (OBJ_neighbor \ - OBJ_current/OBJ_current)$,

**If** $(\Delta < 0)$ **then**:

$ac \leftarrow (ac + 1)$,

$Current_solution \leftarrow Neighbor_solution$,

$OBJ_current \leftarrow OBJ_neighbor$,

**If** $OBJ_neighbor < OBJ_best$ **then**:

$Best_solution \leftarrow Neighbor_solution$,

$OBJ_best \leftarrow OBJ_neighbor$,

$f \leftarrow 0$,

**Else**

$P_{acc} \leftarrow \left(e^{\frac{\Delta}{T_q}}\right)$,

Generate $U$ between $(0, 1)$,

**If** $(U < P_{acc})$ **then**:

$ac \leftarrow (ac + 1)$,

$Current_solution \leftarrow Neighbor_solution$,

$OBJ_current \leftarrow OBJ_neighbor$,

**Until** $(I == Iter)$.

***Annealing***

$q \leftarrow (q + 1)$,

Degenerate $(T_q)$ using *CS,*

$P_{acc} \leftarrow \left(\frac{ac}{Iter}\right)$,

**If** $(P_{acc} \leq 0.15)$ **then**:

$f \leftarrow (f + 1)$,

**Until** $(f == FR \ || \ T_q == T_f)$,

**Return** $Best_solution \ \&\& \ OBJ_best$,

**End**.

---

As mentioned previously, the appropriate design of parameters has significant impact on the efficiency of SA algorithm. In this research, we have used different levels of parameters for different problem sizes. In addition, the problems are categorized by the length of solution presented by $Sol_{Length} = W$, where $W$ is defined in Section 2.2. More detailed explanation of parameter calibration is discussed in the following section.

### *2.4.3 Computational Results*

In order to assess the empirical performance of the proposed SA algorithm, the computational experiments have been carried out on a set of randomly generated instances.

The problem sizes are determined by varying the levels of 3 parameters, which are respectively the number of part types, the number of stages, and the number of single gripper robots. The number of part types varies at two levels: 10 and 20. There are also five levels for the number of stages $\{2, 4, 6, 8, 10\}$, through which parts have to pass and be processed, input and output devices. Furthermore, the number of robots varies between 2 and 20 for 10 levels. For simplicity, the same number of parallel machines is assumed for every stage. Furthermore, $E_{j,s}$ is generated randomly for each part and stage, and hence the number of eligible parallel machines for each part at each stage varies. The combination of above mentioned parameter levels generates 100 different problem instances.

For each part, an integer standard processing time at every stage are generated from the uniform distribution U[10, 100]. The number of eligible machines for processing each part is selected randomly at each stage. The unrelated machines at each stage operate at three levels of processing speed, which are respectively 1, 2 and 3. It means that each part's standard processing time is multiplied by the processing speed of the eligible machine.

To test the effectiveness of proposed block properties (BP) based neighborhood structure, the developed SA algorithm was solved for the generated problem instances with and without BP based neighborhood that leads to $SA^{BP}$ and $SA^{IN}$ algorithms, respectively.

#### *2.4.3.1 Parameter Settings of $SA^{BP}$*

In order to calibrate the proposed $SA^{BP}$, Taguchi method is used in this research to statistically design the experimental investigation. Essentially, traditional experimental design procedures are too complicated and are not easy to use because a large number of experimental works have to be carried out when the number of process parameters increases. To solve this problem, the Taguchi method uses a special design of orthogonal arrays to study the entire parameter space with only a small number of experiments. Orthogonal arrays of Taguchi, the signal-to-noise (S/N) ratio, the analysis of variance (ANOVA) are employed to find the appropriate levels and to analyze the effect of parameters on objective values obtained for different problem sizes.

The S/N ratio indicates the amount of variation present in the response variable. The aim is to maximize the signal-to-noise ratio. Taguchi categorizes objective functions into three groups: the smaller-the-better type, the larger-the-better type, and nominal-is-best type. Since almost all objective functions in scheduling are classified in the smaller-the-better type, its corresponding S/N ratio (Phadke, 1989) is:

$$\text{S/N ratio} = -10 \times \log_{10}(\text{objective function})^2 \tag{2.29}$$

As mentioned previously, different levels of parameters are considered to be analyzed for different problem sizes. The $Sol_{Length}$ is separated into 4 intervals and the generated test problems are categorized according to their related $Sol_{Length}$ in regard to these intervals as shown in Table 2.9. So, there are four groups of problems where the SA parameters and their considered levels for each group are reported in Table 2.10.

Table 2.9 Problems categorized into four groups.

| Part | Stage | Groups of problems (Categorized by $Sol_{Length} = J \times (S+1)$) | | | |
|---|---|---|---|---|---|
| | | 30 - 60 | 70 - 100 | 110 - 140 | 180 - 220 |
| 10 | 2 | ■ | | | |
| | 4 | ■ | | | |
| | 6 | | ■ | | |
| | 8 | | ■ | | |
| | 10 | | | ■ | |
| 20 | 2 | ■ | | | |
| | 4 | | ■ | | |
| | 6 | | | ■ | |
| | 8 | | | | ■ |
| | 10 | | | | ■ |

Table 2.10 Considered levels of SA parameters for each group of problems.

| SA Parameters | Groups of problems | | | |
|---|---|---|---|---|
| | 30 - 60 | 70 - 100 | 110 - 140 | 180 - 220 |
| N | 55 | 60 | 90 | 100 |
| | 65 | 80 | 120 | 150 |
| | 75 | 100 | 150 | 200 |
| Iter | 30 | 50 | 75 | 90 |
| | 40 | 70 | 100 | 120 |
| | 60 | 90 | 125 | 150 |
| $T_0$ | 15 | 15 | 20 | 20 |
| | 20 | 20 | 30 | 30 |
| | 30 | 30 | 40 | 40 |
| CS | LiC | LiC | LiC | LiC |
| | ExC | ExC | ExC | ExC |
| | HyC | HyC | HyC | HyC |

From the standard table of orthogonal arrays, the L9 is selected as the fittest orthogonal array design which fulfills all of our requirements. In order to conduct the experiment, we implemented the proposed $SA^{BP}$ five times for each selected problem from each category and used the mean makespan value to compare their performances. Having obtained the results of the Taguchi experiment, they are transformed into S/N ratios. The preferred levels of SA parameters specified according to the S/N ratios and mean makespan values are reported in Table 2.11 for each group of problems.

Table 2.11 Preferred levels of SA parameters for each group of problems separately.

| SA Parameters | Groups of problems | | | |
|---|---|---|---|---|
| | 30 - 60 | 70 - 100 | 110 - 140 | 180 - 220 |
| N | 75 | 80 | 120 | 200 |
| Iter | 60 | 90 | 125 | 150 |
| $T_0$ | 20 | 20 | 30 | 40 |
| CS | Exponential | Exponential | Hyperbolic | Hyperbolic |

To explore the relative significance of individual factors in terms of their main effects on the objective function, the analysis of variance (ANOVA) is conducted for each problem size separately. The results of analysis are presented in Table 2.12. Each parameters' percentage of influence on objective function (X) and its rank is reported in Table 2.12 for each group of problems.

Table 2.12 ANOVA table of S/N ration for each group of problems.

| Groups of problems | Factor | DF | SS | MS | F | X | Rank |
|---|---|---|---|---|---|---|---|
| 30 - 60 | N | 2 | 46.586 | 23.293 | 27.468 | 28.584 | 2 |
| | Iter | 2 | 29.743 | 14.871 | 17.537 | 18.250 | 4 |
| | $T_0$ | 2 | 37.509 | 18.754 | 22.116 | 23.015 | 3 |
| | CS | 2 | 47.441 | 23.720 | 27.972 | 29.109 | 1 |
| | Error | 2 | 1.696 | 0.848 | | | |
| | Total | 10 | 162.977 | | | | |
| 70 - 100 | N | 2 | 44.235 | 22.117 | 24.145 | 28.383 | 1 |
| | Iter | 2 | 31.593 | 15.796 | 17.245 | 20.271 | 4 |
| | $T_0$ | 2 | 36.758 | 18.379 | 20.064 | 23.586 | 3 |
| | CS | 2 | 41.427 | 20.713 | 22.613 | 26.582 | 2 |
| | Error | 2 | 1.832 | 0.916 | | | |
| | Total | 10 | 155.846 | | | | |
| 110 - 140 | N | 2 | 56.097 | 28.048 | 28.275 | 34.768 | 1 |
| | Iter | 2 | 36.930 | 18.465 | 18.614 | 22.888 | 2 |
| | $T_0$ | 2 | 32.372 | 16.186 | 16.316 | 20.063 | 4 |
| | CS | 2 | 33.963 | 16.981 | 17.118 | 21.049 | 3 |
| | Error | 2 | 1.984 | 0.992 | | | |
| | Total | 10 | 161.348 | | | | |
| 180 - 220 | N | 2 | 67.0481 | 33.524 | 29.641 | 36.600 | 1 |
| | Iter | 2 | 49.209 | 24.604 | 21.754 | 26.862 | 2 |
| | $T_0$ | 2 | 33.122 | 16.561 | 14.643 | 18.080 | 3 |
| | CS | 2 | 31.548 | 15.774 | 13.947 | 17.221 | 4 |
| | Error | 2 | 2.262 | 1.131 | | | |
| | Total | 10 | 183.190 | | | | |

*2.4.3.2 Comparison Between $SA^{BP}$ and $SA^{IN}$*

After determining the preferred level of SA parameters for each problem, to achieve a consistent evaluation of the performance of proposed $SA^{BP}$ and $SA^{IN}$ algorithms and to reduce the effect of their stochastic nature, all problems have been solved 10 times. Tables 1.13 and 1.14 show the average makespan values over 10 runs. The average solution times are very negligible, which are between hundreds of a second and a minute.

The SA algorithms for the BHFS-RCS problem were coded in C#.NET and run on a Pentium 4, 100 GHz PC. The percentage of reduction in average makespan (GAP) for $SA^{BP}$ compared to $SA^{IN}$ has been shown in Table 2.15.

$$GAP = \left(\frac{SA^{IN}_{Makespan} - SA^{BP}_{Makespan}}{SA^{IN}_{Makespan}}\right) \times 100 \tag{2.30}$$

Due to the stochastic nature of the proposed SA algorithms, it may lead to a different GAP for different problem instances. Therefore, *GAP* can be treated as a random variable and the data reported in Table 2.15 is considered as a sample from this variable. Let $\mu$ denote the mean value of *GAP*. The aforementioned data is used to test the following hypothesis:

$$H_0 : \mu \geq 8.2$$
$$H_1 : \mu < 8.2 \tag{2.31}$$

Let us consider probability of type II error as $\beta = 0.1$ for $\mu = 5$. For a $(1 - \alpha) = 0.95$ confidence, it can be realized from statistics that a sample of size $n = 51$ is sufficient to test this hypothesis. The value of *t*-statistic is obtained by the following equation:

$$t = \frac{\sqrt{n}\,(\overline{GAP} - 8.2)}{S} \tag{2.32}$$

Where, $\overline{GAP}$ and $S$ stand for sample mean and standard deviation, respectively. These parameters are calculated as $\overline{GAP} = 7.6228$ and $S = 2.6163$ for the data reported in Table 2.15 and hence $t = -1.5755$. Since $t_{0.05,50} = -1.6759 < -1.5755$, the hypothesis $H_0$ is accepted and the proposed $SA^{BP}$ algorithm has reduced the average makespan of $SA^{IN}$ algorithm by 8.2 % with 95% confidence.

Table 2.13 Average makespan obtained by proposed $SA^{BP}$ and $SA^{IN}$ algorithms.

| Parts | Stages | Obtained Makespan for different Numbers of Robots (NR) | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | NR = 2 | | NR = 4 | | NR = 6 | | NR = 8 | | NR = 10 | |
| | | $SA^{BP}$ | $SA^{IN}$ | $SA^{BP}$ | $SA^{IN}$ | $SA^{BP}$ | $SA^{IN}$ | $SA^{BP}$ | $SA^{IN}$ | $SA^{BP}$ | $SA^{IN}$ |
| | **2** | **293.34** | **328.65** | **194.75** | **229.26** | 193.38 | 221.75 | 195.31 | 219.29 | 192.84 | 220.73 |
| | **4** | **804.54** | **902.57** | **584.62** | **619.59** | **521.48** | **594.54** | 523.64 | 603.91 | 519.31 | 598.17 |
| **10** | **6** | **874.48** | **961.75** | **798.96** | **853.12** | **737.84** | **812.48** | **646.74** | **705.96** | 643.71 | 700.79 |
| | **8** | **1201.74** | **1285.82** | **1033.58** | **1147.75** | **925.56** | **1084.74** | **897.82** | **961.58** | **862.75** | **947.56** |
| | **10** | **1422.82** | **1537.69** | **1203.87** | **1314.87** | **1077.13** | **1157.82** | **1057.69** | **1116.87** | **1052.87** | **1104.13** |
| | **2** | **752.29** | **851.98** | **680.62** | **736.87** | **599.37** | **683.04** | 595.46 | 677.39 | 600.89 | 681.64 |
| | **4** | **1905.76** | **2126.50** | **1707.69** | **1827.93** | **1679.63** | **1864.40** | **1559.07** | **1714.93** | 1560.13 | 1705.97 |
| **20** | **6** | **2878.22** | **3068.12** | **2653.48** | **2866.21** | **2909.00** | **3131.25** | **2553.33** | **2728.40** | **2454.51** | **2632.03** |
| | **8** | **4439.92** | **4683.65** | **3695.11** | **3894.60** | **3583.18** | **3756.02** | **3411.63** | **3524.28** | **3188.03** | **3367.40** |
| | **10** | **5187.69** | **5446.28** | **4494.18** | **4851.47** | **4319.81** | **4661.68** | **4205.55** | **4587.32** | **4066.22** | **4435.67** |



Table 2.14 Average makespan obtained by proposed $SA^{BP}$ and $SA^{IN}$ algorithms.

| Parts | Stages | Obtained Makespan for different Numbers of Robots (NR) | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | NR = 12 | | NR = 14 | | NR = 16 | | NR = 18 | | NR = 20 | |
| | | $SA^{BP}$ | $SA^{IN}$ | $SA^{BP}$ | $SA^{IN}$ | $SA^{BP}$ | $SA^{IN}$ | $SA^{BP}$ | $SA^{IN}$ | $SA^{BP}$ | $SA^{IN}$ |
| | **2** | 194.71 | 219.67 | 191.89 | 228.12 | 195.68 | 220.75 | 193.95 | 225.19 | 197.24 | 221.98 |
| | **4** | 524.57 | 590.56 | 519.72 | 592.39 | 520.37 | 599.84 | 521.51 | 589.90 | 520.79 | 600.27 |
| **10** | **6** | 647.81 | 711.51 | 646.11 | 698.79 | 648.13 | 700.07 | 645.74 | 701.99 | 646.97 | 704.96 |
| | **8** | 862.41 | 946.84 | 860.74 | 940.53 | 863.62 | 950.11 | 862.86 | 948.24 | 858.92 | 945.82 |
| | **10** | **998.59** | **1079.32** | **981.91** | **1049.78** | **960.29** | **1057.88** | 958.81 | 1042.59 | 960.71 | 1047.61 |
| | **2** | 599.12 | 679.63 | 602.81 | 680.49 | 600.48 | 689.70 | 595.93 | 678.57 | 598.74 | 679.41 |
| | **4** | 1555.63 | 1698.91 | 1554.93 | 1710.68 | 1563.11 | 1721.66 | 1560.43 | 1715.04 | 1561.01 | 1720.74 |
| **20** | **6** | **2184.41** | **2271.37** | 2181.07 | 2274.39 | 2181.86 | 2265.29 | 2187.57 | 2270.40 | 2185.08 | 2273.63 |
| | **8** | **3115.81** | **3276.89** | **3059.29** | **3251.01** | 3056.12 | 3244.71 | 3055.79 | 3253.48 | 3049.21 | 3261.03 |
| | **10** | **4001.21** | **4224.92** | **3954.44** | **4194.01** | **3841.22** | **4037.78** | **3753.22** | **4035.00** | 3741.33 | 4031.93 |

Table 2.15 The GAP in each problem between $SA^{BP}$ and $SA^{IN}$ for different number of operative robots.

| Part | Stage | Number of Robots | | | | | | | | | Average |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 2 | 4 | 6 | 8 | 10 | 12 | 14 | 16 | 18 | |
| 10 | 2 | 10.746 | 15.053 | - | - | - | - | - | - | - | 12.899 |
| | 4 | 10.861 | 5.645 | 12.288 | - | - | - | - | - | - | 9.598 |
| | 6 | 9.074 | 6.348 | 9.186 | 8.388 | - | - | - | - | - | 8.249 |
| | 8 | 6.539 | 9.947 | 14.674 | 6.630 | 8.950 | - | - | - | - | 9.348 |
| | 10 | 7.470 | 8.441 | 6.969 | 5.298 | 4.642 | 7.479 | 6.465 | 9.225 | - | 6.998 |
| 20 | 2 | 11.701 | 7.633 | 12.249 | - | - | - | - | - | - | 10.527 |
| | 4 | 10.380 | 6.577 | 9.910 | 9.088 | - | - | - | - | - | 8.988 |
| | 6 | 6.189 | 7.422 | 7.097 | 6.416 | 6.744 | 3.828 | - | - | - | 6.282 |
| | 8 | 5.203 | 5.122 | 4.601 | 3.196 | 5.326 | 4.915 | 5.897 | - | - | 4.894 |
| | 10 | 4.748 | 7.364 | 7.333 | 8.322 | 8.329 | 5.295 | 5.712 | 4.868 | 6.983 | 6.550 |
| Total | | Sample size for the variable GAP (n) = 51 | | | | | | | | | 84.333 |

Furthermore, Figure 2.14 clarifies the effect of using different numbers of robots on the makespan and illustrates the reduction in makespan for each problem as the number of robots increases. The points in this figure are the mean makespan values for different numbers of robots. It can be seen that the makespan will not be further improved after a certain number of robots used. Also, it can be seen from Figure 2.15 that as the number of robots increases, the effect of increasing the number of stages on the makespan will reduce. These points demonstrate that as the number of stages (and machines) increases in the system, the number of robots added eventuate significant reduction in the makespan. Moreover, Table 2.16 presents that the needed maximum number of robots would not be larger than the maximum number of utilized machines in the cell.

Table 2.16 Maximum number of operative robots (MOR) for each problem

| Parts | Stages | Max number of used machines in the cell | $SA^{BP}$ – Multiple Robots | |
|---|---|---|---|---|
| | | | MOR | Makespan |
| 10 | 2 | 6 | 4 | 194.75 |
| | 4 | 12 | 6 | 521.48 |
| | 6 | 18 | 8 | 646.74 |
| | 8 | 24 | 10 | 862.75 |
| | 10 | 30 | 16 | 960.29 |
| 20 | 2 | 6 | 6 | 599.37 |
| | 4 | 12 | 8 | 1559.07 |
| | 6 | 18 | 12 | 2184.41 |
| | 8 | 24 | 14 | 3059.29 |
| | 10 | 30 | 18 | 3753.22 |

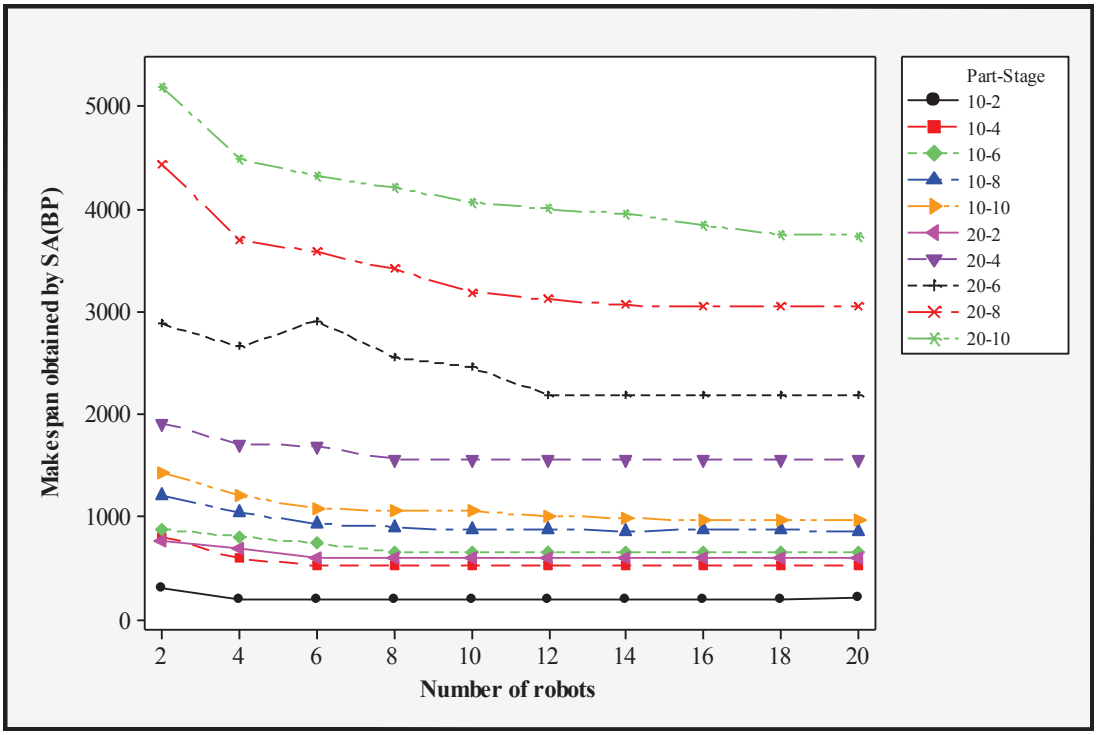



Figure 2.14 Average makespan reduction using different numbers of robots for each problem.

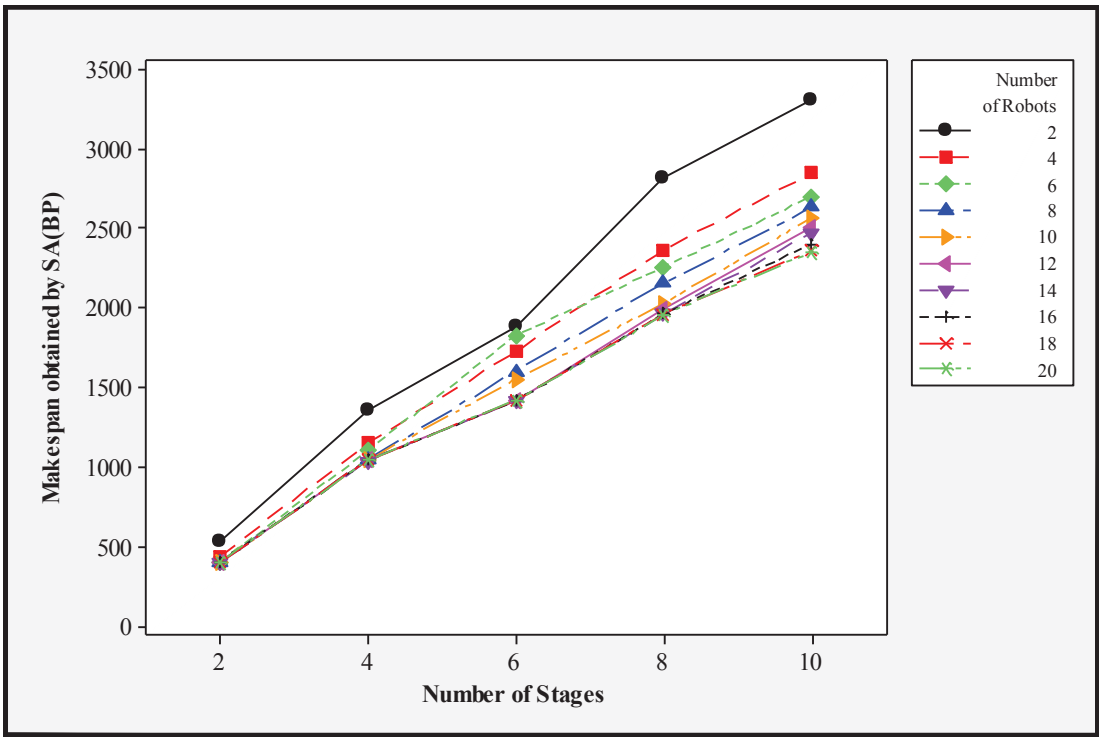



Figure 2.15 Interaction Plot for the makespan of each problem obtained by $SA^{BP}$

### *2.4.3.3 Comparing $SA^{BP}$ with Optimum Methods*

Furthermore, to investigate the efficiency of $SA^{BP}$ and compare with the obtained optimal solutions from the proposed MILP model, small size problems are generated. The size of each problem, in terms of total number of parts, stages, machines, and number of robots have been reported in Table 2.17. The proposed MILP model has been coded in GAMS-23.7 using two ways respectively; without and with Shortest Ordered Sets First (SOS-1) variables. The variables $Y_{j,s,m}$ , $Z_{j,s,f}$ , and $RS_{r,f}$ are considered as binary variables and SOS-1 variables respectively in the coded models.

To reduce the effect of stochastic nature of proposed algorithm, $SA^{BP}$ has been run five times for each problem. The mean and the best makespan values for each problem in five runs of $SA^{BP}$ are reported in Table 2.17. The gap value in Table 2.17 demonstrates the percentage of deviation of the mean and best makespan obtained by the proposed SA from the optimal value obtained by GAMS, and it is calculated as follows.

$$GAP_OPT = \left(\frac{SA^{BP} - OPT_{makespan}}{OPT_{makespan}}\right) \times 100 \tag{2.33}$$

Also the CPU times for both $SA^{BP}$ and GAMS are included in Table 2.17. The proposed $SA^{BP}$ found the optimal solution for 7 out of 12 test problems. As shown in Table 2.17, for the last set of problems (Num. 8 - 12) GAMS failed to provide the optimal solution even after 120 hours. Using SOS-1 variables did not increase the capability of the MILP model. The best objective function values found within this time period are included in the table, and accordingly the proposed $SA^{BP}$ algorithm produced 11.20%, 21.08%, 13.46%, 62.81%, and 71.34% better quality solutions in rather short computing times respectively for these problems.

Table 2.17 Computational results for the small size problems solved by SABP and GAMS .

| J | S | R | GAMS | | | | $SA^{BP}$ | | | GAP-OPT | | NO |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | Binary Variables | | SOS 1 Variables | | Obj | | $Time^{Mean}$ (sec) | Mean | Best | |
| | | | Obj | Time(hr) | Obj | Time(hr) | Mean | Best | | | | |
| 5 | 2 | 1 | 323 | 46:41.71 | 323 | 43:29.96 | 325.8 | 323 | 00:0.70 | 0.85 | 0 | 1 |
| | | 2 | 272 | 55:00.71 | 272 | 48:54.35 | 280.8 | 272 | 00:0.77 | 3.13 | 0 | 2 |
| | 3 | 1 | 514 | 67:44.96 | 514 | 61:24.51 | 527.3 | 514 | 00:1.12 | 2.52 | 0 | 3 |
| | | 2 | 356 | 91:50.15 | 356 | 82:43.84 | 362.2 | 356 | 00:1.14 | 1.71 | 0 | 4 |
| 6 | 2 | 1 | 385 | 86:43.45 | 385 | 81:16.38 | 392.8 | 385 | 00:0.83 | 1.98 | 0 | 5 |
| | | 2 | 304 | 104:03.57 | 304 | 95:39.12 | 314.3 | 304 | 00:0.99 | 3.27 | 0 | 6 |
| | 3 | 1 | 617 | 101:50.84 | 617 | 96:21.72 | 631.7 | 617 | 00:1.24 | 2.32 | 0 | 7 |
| | | 2 | *549** | *120:00.00* | *491** | *120:00.00* | 449.2 | 436 | 00:1.42 | -8.51 | -11.20 | 8 |
| 7 | 2 | 1 | *574** | *120:00.00* | *574** | *120:00.00* | 467.4 | 453 | 00:1.26 | -18.57 | -21.08 | 9 |
| | | 2 | *401** | *120:00.00* | *401** | *120:00.00* | 362.2 | 347 | 00:1.28 | -9.67 | -13.46 | 10 |
| | 3 | 1 | *1724** | *120:00.00* | *1724** | *120:00.00* | 662.4 | 641 | 00:1.75 | -61.57 | -62.81 | 11 |
| | | 2 | *1724** | *120:00.00* | *1724** | *120:00.00* | 511.6 | 494 | 00:1.88 | -70.32 | -71.34 | 12 |

* the obtained solution after 120 minutes, which is not optimal.

### *2.4.4 Conclusion*

This research addresses the Blocking Hybrid Flow Shop Robotic Cell Scheduling (BHFS-RCS) problem considering unrelated parallel machines at each stage, multiple part types, and machine eligibility constraints. A mixed-integer linear programming (MILP) model is proposed for the BHFS-RCS problem to minimize makespan. Since this model is incapable of producing good feasible solutions in reasonable computing times for real size problems, two SA based algorithms named $SA^{IN}$ and $SA^{BP}$ are proposed to heuristically solve the mentioned problem. In addition, to evaluate the performance of the proposed algorithms, 10 problem categories are generated to be analyzed by different numbers of robots and solved by both proposed $SA^{IN}$ and $SA^{BP}$ algorithms. The proposed $SA^{IN}$ and $SA^{BP}$ algorithms have been tested on 100 problem instances. $SA^{BP}$ algorithm has shown to be a successful heuristic in solving the problem. According to the computational results, $SA^{BP}$ algorithm reduced the makespan by 8.2 % over $SA^{IN}$ algorithm. As a result, using block properties made the search process more efficient for the BHFS-RCS problem. Also, small size problems are used to compare the performance of $SA^{BP}$ with the optimal solutions.

# CHAPTER THREE
# MULTI-DEGREE CYCLIC FLOW SHOP ROBOTIC CELL SCHEDULING PROBLEM WITH MULTIPLE ROBOTS: A MIXED-INTEGER LINEAR PROGRAMMING MODEL

## 3.1 Introduction

In this research, we focus on the sequencing and scheduling problem of a particular type of automated material handling system in cellular manufacturing: cyclic robotic cells. A flow shop robotic cell usually consists of an input device, an output device, a sequence of processing machines, and a crew of identical programmable robots. Parts to be processed enter the system from the input device. After they are processed successively through a sequence of machines, they leave the robotic cell from the output device. Each machine can process only one part at a time. Since there is no buffer for intermediate storage between the machines and each machine can hold only one part, this robotic cell is, in essence, a flow shop with blocking constraint, and the transfer of the parts between machines is handled by single gripper robots. In this chapter, the multi-robot multi-degree cyclic flow shop robotic cell scheduling problem is considered, in which the same type of parts are produced in large quantities. In cyclic scheduling, the robots perform a series of material handling operations in a repetitive way in order to achieve the production target. In a single-degree cyclic schedule, a single part is completed when the robot or the robots complete the sequence of move operations. Whereas, in a multi-degree cyclic schedule ($k$-degree cyclic schedule), $k$ parts are completed when the robot or the robots complete the sequence of move operations. Although shorter cycle times can be achieved with multi-degree cyclic schedules, the studies have been confined to a limited framework in the relevant literature, and mostly they have been engaged in developing one-degree cyclic schedules in single robot systems. This study proposes a novel mixed-integer linear programming (MILP) model to simultaneously find the optimal degree ($k$) of the cyclic schedule and the optimal sequence of robots' moves corresponding to the $k$-degree cyclic schedule, which eventually minimize the cycle time and thus maximize the throughput rate.

Given the processing requirements, the objective that is most interested by manufacturers is the maximization of cell productivity. The mean cycle time of a *k*-degree cyclic schedule is defined as the whole cycle time divided by *k*. The throughput rate is the inverse of the mean cycle time.

A natural and widely used measure of productivity in robotic cells is throughput rate: the number of finished parts produced per unit of time (M. W. Dawande, et al., 2007). In the blocking flow shop robotic cells, a crew of computer-controlled robots performs the movements of parts between the machines. By optimally assigning the robots to these moves and sequencing the transportation operations at the same time, the throughput rate of the cell is maximized.

### 3.2 Literature Review

Sethi, et al., 1992 performed the purely analytic study of a flow shop robotic cell containing two and three machines and a single robot for one-degree cyclic schedules. Logendran and Sriskandarajah Logendran & Sriskandarajah (1996) extended this work to cover different configurations of cells and more general times for the actions of the robot. Crama & Van De Klundert (1997) and M. Dawande, Sriskandarajah & Sethi (2002), each developed polynomial-time algorithms to find the optimal one-unit cycles considering machine numbers. More thorough reviews of the cyclic robotic flow shop cells literature can be found in surveys by Hall (2007), Crama, Kats, Van de Klundert & Levner (2000), and M. Dawande, et al. (2005), and in the book by M. W. Dawande, et al. (2007).

Agnetis (2000) studied the two- and three-machine one-degree cyclic robotic cells with a single robot and developed polynomial algorithms to solve the considered problem. Levner, Kats & Levit (1997) extended the same problem to *m*-machine using the notion of prohibited intervals. They also considered the usage of more than one robot in large robotic cells in order to eliminate material handling bottlenecks. Karzanov & Livshits (1978) studied a system with parallel tracks where robots run along their respective tracks, and they proposed an algorithm to find the minimal

number of robots for a given cycle time. Furthermore, J. Liu & Jiang (2005) proposed an efficient algorithm for one-degree cyclic scheduling of two robots in a no-wait robotic cell.

Moreover, many researchers have noted and verified that multi-degree cyclic schedules usually have larger throughput than one-degree cyclic schedules. Kats, Levner & Meyzin (1999) studied the *k*-degree cyclic robotic cells with a single robot and proposed a sieve method based algorithm to optimize the cycle time. On the basis of their problem, Che, Chu & Chu (2002) proposed an exact algorithm based on branch-and-bound procedure for the no-wait robotic cells. They have noted that the proposed algorithm is polynomial for a fixed *k*-degree ($k$ >1), but becomes exponential if $k$ is assumed arbitrary. Lately, Che & Chu (2005) showed that the no-wait multi-degree cyclic robotic cell scheduling problem with two parallel robots can be also solved in polynomial time for a fixed *k*-degree ($k$ >1), but exponential if $k$ is arbitrary. However, the algorithm cannot be mechanically extended to the multi-robot case because it explores some properties specific to the cells with two robots.

Furthermore, Che & Chu (2009) considered the *k*-degree cyclic flow shop robotic cell scheduling problem with multiple robots in a no-wait system. They proposed a polynomial algorithm to find the minimum number of robots for all *k*-degree ($k$ >1) cyclic schedules and consequently the optimal cycle time for any given number of robots. On the basis of this study, Che, Chabrol, Gourgand & Wang (2012) extended the proposed method to the cyclic scheduling of a no-wait re-entrant robotic flow shop with multiple robots. Zhou, Che & Yan (2012) proposed a mixed-integer programming model for the fixed *k*-degree cyclic flow shop robotic cells with a single robot and considering time window constraints. However, their model should be solved separately for different values of $k$ to determine the optimum degree of the cyclic schedule.

Table 3.1 Literature of cyclic flow shop robotic cell.

| Num | Research | Robot Number | Degree of Cyclic Schedule | Machine Number | Pickup Criterion | Main contribution |
|---|---|---|---|---|---|---|
| **1** | H. Chen, Chu & Proth (1998) | 1 | 1 | M | Time Window | A branch-and-bound algorithm |
| **2** | Chu Che, et al. (2002) | 1 | K | M | No-Wait | A branch-and-bound algorithm |
| **3** | Che & Chu (2005) | 2 | K | M | No-Wait | A polynomial-time algorithm |
| **4** | Neil Geismar, Dawande & Sriskandarajah (2005) | 1 | K | M | Free | A polynomial-time algorithm for the three most common classes: additive, constant, and Euclidean travel-time. |
| **5** | Leung & Levner (2006) | R | 1 | M | No-Wait | A polynomial-time algorithm to find the minimum number of robots for all feasible cycle times, and to find the optimal cycle time for any given number of robots. |
| **6** | Alcaide, Chu, Kats, Levner & Sierksma (2007) | R | K | M | Time Window | Graph model and PERT–CPM solution techniques |
| **7** | Jiang & Liu (2007) | R | 1 | M | No-Wait | A graph model of the production process is developed, making it possible to apply PERT–CPM solution techniques. |
| **8** | Kats & Levner (2009) | 1 | 2 | M | No-Wait | A polynomial-time algorithm |
| **9** | Che & Chu (2009) | R | K | M | No-Wait | A polynomial-time algorithm |
| **10** | Che, Hu, Chabrol & Gourgand (2011) | R | 2 | M | No-Wait | A mathematical model and a polynomial algorithm for the multi-robot 2-cyclic scheduling problem |
| **11** | Zhou, et al. (2012) | 1 | K | M | Time Window | A mixed-integer linear programming model |
| **12** | Che, et al. (2012) | R | K | M | No-Wait | A polynomial-time algorithm |
| **13** | Jiang & Liu (2014) | R | 1 | M | Time Window | A mixed-integer linear programming model and an efficient branch and bound strategy |
| **14** | This research | R | K (Optimum value) | M | Time Window | A mixed-integer linear programming model |

As far as the related research is concerned, there is no mathematical modeling formulation in the literature for the multi-degree cyclic flow shop robotic cell scheduling (*k*DCFS-RCS) problem with multiple robots and time window constraints

for processing times. The proposed model also assumes the degree of the cyclic schedule as a variable, and determines the optimal degree of the cyclic schedule and the optimal sequencing of the robots moves simultaneously such that the throughput rate is maximized.

### 3.3 Problem Definition

The robotic flow shop under study consists of multiple computer-controlled robots for material handling and $M$ machines assigned in a row. These machines are indexed as $m \in \{m_1, m_2, \dots, m_{IM}\}$, where $m_1$ and $m_{IM}$ are the loading and the unloading stations, respectively. All the parts are of the single type and processed in the robotic flow shop by successively visiting the machines. The part flow can be described as follows: After a part is moved into the robotic flow shop from the loading station $m_1$, it is processed sequentially through machines $m \in \{m_2, m_3, \dots, m_{IM-1}\}$, and finally leaves the flow shop from the unloading station $m_{IM}$.

In the cyclic flow shop, all the parts are categorized to classes and the parts of the same class have the same conditions in consecutive cycles. In other words, if there are $J$ parts that would be processed in a cycle, the parts are indexed as classes $\{1, 2, \dots, J\}$ according to their sequence of entering the system during the cycle. There may be parts from the same class in each cycle. Thereupon, some of the consecutive processing operations of parts may continue in the upcoming cycles. This means that only one part from the same class is completed per cycle. Therefore, if there are $p$ parts from the same class, the processing operations of this class are divided to $p$ sets of consecutive operations in a cycle.

With the problem description and notation given above, minimizing the mean cycle time needed for per part, requires finding an optimal $k$-degree cyclic schedule. Obviously, taking the degree ($k$) of the cyclic schedule as a variable eliminates the need for solving the cyclic scheduling problem for given degrees separately and comparing their resultant cycle times to decide which one is best. Here the objective

is to find the optimal *k*-degree cyclic schedule that sequences the transportation operations and that assigns the appropriate robots to perform each transportation operation such that the mean cycle time per part is minimized. Figure 3.1 illustrates an example of a robotic cell containing ten machines (including input and output devices) with two single-gripper robots.

Before presenting the proposed mathematical models in detail, the following characteristics about the flow shop robotic cell are stated to make the description of the models easier.

- There are *IM* machines including input and output devices assigned in a row.
- The parts are processed by successively visiting the machines.
- All the parts are of identical part type.
- Each machine can process only one part at any time.
- The parts cannot be processed on more than one machine at the same time.
- There is no intermediate buffer between the machines.
- Each part is transported to the next machine within a given time window after completing its processing on a machine.
- Each robot can transport only one part at the same time (single-gripper).
- Each transportation operation is performed by only one robot.
- The required time for the robots to transport the parts between machines and for their empty moves are deterministic constants.

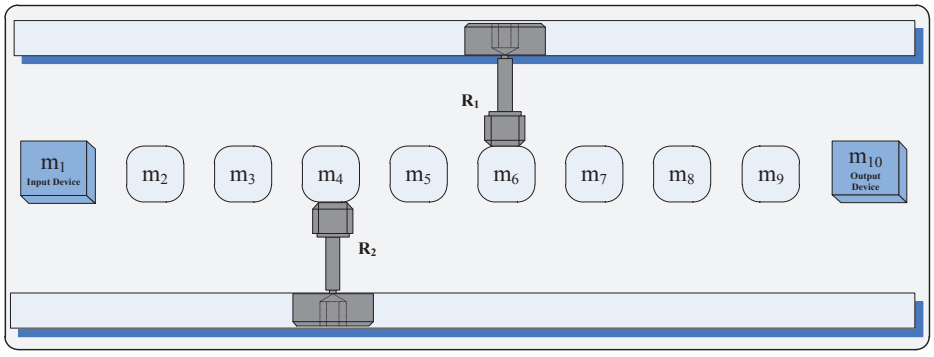



Figure 3.1 An example of the cyclic flow shop robotic cell with multiple single-gripper robots.

### 3.4 Mathematical Programming Model

The transport of a part of class $j$ from machine $m$ to machine $m+1$ by a robot is defined as transportation operation $[j,m]$. A transportation operation $[j,m]$ is made up of three simple robot sub-operations: (1) unloading a part of class $j$ from machine $m$; (2) transporting a part of class $j$ to the next machine $m+1$; and (3) loading it onto machine $m+1$. Thus, the total number of transportation operations in a $k$-degree cyclic schedule is $k \times (IM-1)$. Under this definition, we define the time required for any robot to perform transportation operation $[j,m]$ (such a travel is also called a completemove $[j,m]$) as $\theta_m$, and the time required for any robot to travel from machine $m$ to machine $n$ without transporting any part (such a travel is also called a void move $[\![m,n]\!]$) as $\delta_{[\![m,n]\!]}$. Note that a cyclic schedule can be described by the starting times of all moves in the system. All the robots are considered to be free at the start of a cycle but not all the processing machines.

*Indices and sets:*

| | |
|---|---|
| $J$ | Maximum number of identical parts that can enter during a cycle, $j \in \{1,2,\dots,J\}$, |
| $IM$ | Number of machines including input and output devices, where $m \in \{1,2,\dots,IM\}$, |
| $R$ | Number of single gripper robots, where $r \in \{1,\dots,R\}$, |
| $[j,m]$ | A complete robot move to transport a part of class $j$ from machine $m$ to machine $(m+1)$, |
| $[\![m,n]\!]$ | A void robot move to travel from machine $m$ to machine $n$ without any part, |

*Parameters*

| | |
|---|---|
| $PLo_m$ | Lower bound for the processing time of a part on machine $m$, where it is equal to zero for all the parts at input and output devices, |
| $PUp_m$ | Upper bound for the processing time of a part on machine $m$, where it is equal to zero for all the parts at input and output devices, |
| $\theta_m$ | Time required for a robot to perform complete move $[j, m]$, |

| | |
|---|---|
| $\delta_{[\![m,n]\!]}$ | Time required for a robot to perform void move $[\![m,n]\!]$, |
| $BM$ | A sufficiently large positive value (big M). |

*Decision variables*

| | |
|---|---|
| $X_j$ | 1 if a part of class $j$ enters the system, 0 otherwise, |
| $Z_{[j,m],r}$ | 1 if complete move $[j,m]$ is done by the $r^{th}$ robot, 0 otherwise, |
| $Y_{[j,m],[j',m']}$ | 1 if complete move $[j,m]$ is done before complete move $[j',m']$, 0 otherwise, |
| $S_{[j,m]}$ | Starting time of complete move $[j,m]$, |
| $T$ | The total cycle time for all the parts that have entered the system, |
| $\Phi$ | The mean cycle time for a part, |
| $\mu$ | Throughput rate. |

### *3.4.1 Mixed-Integer Non-Linear Programming Model*

Considering the special characteristics of the *k*DCFS-RCS problem, the objective function and the constraints sets of the proposed model are discussed separately in the following sections.

#### *3.4.1.1 Objective Function*

As mentioned previously, the number of part classes that enter the system during the cycle influences the throughput rate. As the number of part classes in a cycle increases, the total cycle time also increases; however, although the total cycle time increases, it is likely to find a solution that minimizes the mean cycle time and thus maximizes the throughput rate, which can be shown with the expressions (3.1) and (3.2) given below:

$$Throughput: \mu = \frac{Number\ \ of\ parts}{Total\ \ cycle\ \ time} \iff Mean\ cycle\ time: \Phi = \frac{Total\ \ cycle\ \ time}{Number\ \ of\ parts} \tag{3.1}$$

$$\begin{cases} \mu_1 = \frac{p_1}{t_1} & \Leftrightarrow \quad \Phi_1 = \frac{t_1}{p_1} \\ \mu_2 = \frac{p_2}{t_2} & \Leftrightarrow \quad \Phi_2 = \frac{t_2}{p_2} \\ p_2 > p_1 \, , t_2 > t_1 \end{cases} \quad \Rightarrow \quad \begin{cases} \Phi_1 < \Phi_2 \ \vee \ \Phi_1 > \Phi_2 \\ \Phi_1 < \Phi_2 \ \vee \ \Phi_1 > \Phi_2 \end{cases} \tag{3.2}$$

Therefore, the *k*-degree of the cyclic schedule has been considered to be determined utilizing the variable $X_j$ in the proposed model while minimizing the cycle time as follows.

$$Minimize : \Phi = T / \sum_{j=1}^{J} X_j \ , \tag{3.3}$$

where the expression ($\sum_{j=1}^{J} X_j$) determines the number of parts that have entered the system during the cycle.

*3.4.1.2 Basic Constraints*

Constraint (3.4) guarantees that at least one part should enter the system at each cycle. Constraint set (3.5) ensures that only consecutively numbered parts enter the system. Constraint set (3.6) implies that if a part does not enter the system during the cycle, there is no need to schedule its transportation operations.

$$\sum_{j=1}^{J} X_j \geq 1 \tag{3.4}$$

$$\begin{cases} j \times X_j \leq \sum_{j'=1}^{j} X_{j'} \\ \forall \{ j \mid j \in \{1,2, \dots, J\} \} \end{cases} \tag{3.5}$$

$$\begin{cases} S_{[j,m]} - BM \times X_j \leq 0 \\ \forall \{ j, m \mid j \in \{1,2, \dots, J\}; \ m \in \{1, \dots, IM - 1\} \} \end{cases} \tag{3.6}$$

*3.4.1.3 Processing Time Constraints*

In classic flow-line production systems, the processing of apart on a machine is started upon the completion of its processing on the previous machine. The *k*DCFS-

RCS problem is a specific version of the flow shop scheduling problem with identical part types. If machine $m$ is free at the beginning of the cycle, then the processing time constraints should consider the following restrictions:

$$S_{[j,m-1]} + \theta_{m-1} + PLo_m \leq S_{[j,m]} \quad (3.7)$$

$$S_{[j,m-1]} + \theta_{m-1} + PUp_m \geq S_{[j,m]}\,, \quad (3.8)$$

where (3.7) and (3.8) state respectively that apart requires at least $PLo_m$ and at most $PUp_m$ time to be processed on machine $m$. On the other hand, if machine $m$ is occupied at the beginning of the cycle, then the processing of a part on the machine is started upon the completion of its processing on the previous machine a cycle before (Figure 3.2) as shown below:

$$S_{[j,m-1]} + \theta_{m-1} + PLo_m \leq S_{[j,m]} + T \quad (3.9)$$

$$S_{[j,m-1]} + \theta_{m-1} + PUp_m \geq S_{[j,m]} + T \quad (3.10)$$

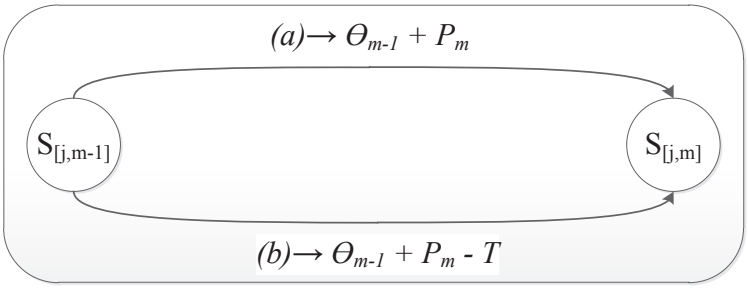



Figure 3.2 The graph representation of processing time constraints related to successive processing operations of a part (Constraint set (3.11)).

The variable $Y_{[j,m-1],[j,m]}$ is used to determine the sequence of transportation operations $[j, m-1]$ and $[j, m]$. In addition, processing time constraints for a part of class $j$ are required whenever the part enters the system during the cycle ($X_j = 1$). Thereupon, the related processing time constraints are formulated as follows:

$$\begin{cases} S_{[j,m-1]} + \theta_{m-1} + PLo_m - BM \times \left(2 - Y_{[j,m-1],[j,m]} - X_j\right) \leq S_{[j,m]} \\ S_{[j,m-1]} + \theta_{m-1} + PLo_m - BM \times \left(1 + Y_{[j,m-1],[j,m]} - X_j\right) \leq S_{[j,m]} + T \\ \forall \left\{j, m \mid j \in \{1,2,\dots,J\};\ m \in \{2,\dots,IM-1\}\right\} \end{cases} \quad (3.11)$$

$$\begin{cases} S_{[j,m-1]} + \theta_{m-1} + PUp_m + BM \times \left(2 - Y_{[j,m-1],[j,m]} - X_j\right) \geq S_{[j,m]} \\ S_{[j,m-1]} + \theta_{m-1} + PUp_m + BM \times \left(1 + Y_{[j,m-1],[j,m]} - X_j\right) \geq S_{[j,m]} + T \\ \forall \left\{j, m \mid j \in \{1,2,\dots,J\};\ m \in \{2,\dots,IM-1\}\right\} \end{cases} \quad (3.12)$$

**Lemma 1**

$$\left(Y_{[j',m'],[j,m]} = 1\right) \Longrightarrow \left(Y_{[j,m],[j',m']} = 0\right) \quad (3.13)$$

**Proof 1**

In the classical logic, the compound $p \Longrightarrow q$ is logically equivalent to the negative compound: not both $p$ and not $q$.

$$p \Longrightarrow q \ \therefore \ {\sim}q \Longrightarrow {\sim}p \quad (3.14)$$

Therefore, considering the logical definition (3.8) we have:

$$\left(Y_{[j',m'],[j,m]} = 1\right) \Longrightarrow \left(S_{[j',m']} + value' \leq S_{[j,m]}\right) \quad (3.15)$$

$$\left(S_{[j',m']} + value' \leq S_{[j,m]}\right) \equiv {\sim}\left(S_{[j,m]} + value \leq S_{[j',m']}\right) \quad (3.16)$$

$${\sim}\left(S_{[j,m]} + value \leq S_{[j',m']}\right) \Longrightarrow {\sim}\left(Y_{[j,m],[j',m']} = 1\right) \quad (3.17)$$

$${\sim}\left(Y_{[j,m],[j',m']} = 1\right) \equiv \left(Y_{[j,m],[j',m']} = 0\right) \quad (3.18)$$

Due to expressions (3.15) - (3.18) we can derive:

$$\left(Y_{[j',m'],[j,m]} = 1\right) \Longrightarrow \left(Y_{[j,m],[j',m']} = 0\right) \quad (3.19)$$

As a result, an additional constraint set as "$Y_{[j,m],[j',m']} + Y_{[j',m'],[j,m]} = 1$" is not required. Lemma 1 is used in some of the remaining constraint sets as well.

*3.4.1.4 Robot Assignment Constraints*

In this study, robotic cells with multiple robots are considered, and therefore, any one of the robots can be used for a transportation operation. The constraint related with the assignment of a robot to a transportation operation of a part of $j^{th}$ class is formulated as follows:

$$\begin{cases} \sum_{r=1}^{R} Z_{[j,m],r} = 1 \\ \forall \ \{j, m \mid j \in \{1,2,\dots,J\};\ m \in \{1,\dots,IM-1\}\} \end{cases} \tag{3.20}$$

As indicated before, the proposed model determines the optimal degree of the cycle as well. Taking this feature into account, constraint set (3.21) ensures that if the entrance of a part of $j^{th}$ class is guaranteed ($X_j = 1$), only one robot should be assigned to each of its transportation operations $[j, m]$.

$$\begin{cases} \sum_{r=1}^{R} Z_{[j,m],r} = X_j \\ \forall \ \{j, m \mid j \in \{1,2,\dots,J\};\ m \in \{1,\dots,IM-1\}\} \end{cases} \tag{3.21}$$

*3.4.1.5 Operational Sequence Constraints for the Machines*

The operational sequence constraints prevent the formation of conflicts in the usage of robots and machines. Each machine should be unloaded before loading the next part on it and starting its processing operation. It should be also noted that since there is no buffer between the machines, the completed parts have to remain on the machines until both the next machine and any of the robots to move the part become available.

$$S_{[j,m]} \leq S_{[j',m-1]} + \theta_{m-1} \tag{3.22}$$

As illustrated in Figure 3.3(a) the sequence of classes on machines 1 to 4 is $\{C1, C2, C3\}$. Meanwhile, machine 5 is occupied by $C3$ at the start time of the cycle. Therefore, the sequence of classes changes cyclically as $\{C3, C1, C2\}$ until the next occupied machine 7. The sequence of classes has changed to $\{C2, C3, C1\}$ and $\{C1, C2, C3\}$ on occupied machines 7 and 8, respectively. It is obvious from Figure 3.3(b) that the cyclic sequence of classes would be changed at the start time of the cycle if machine $m_n$ is occupied.

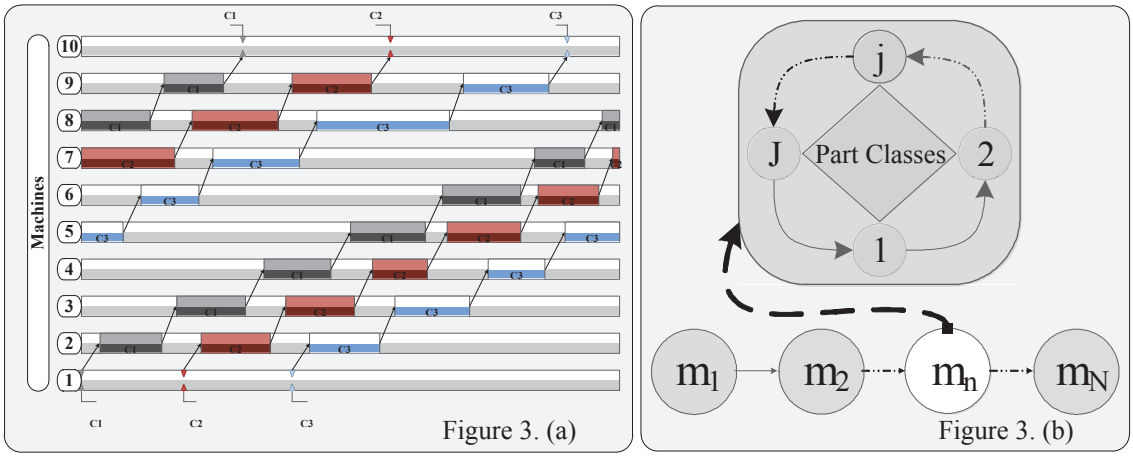



Figure 3.3 (a) The example Gantt Chart for 3-degree cyclic solution. (b) The cyclic change of the sequence of classes on machines after any occupied machine.

Considering the cyclic changes in the sequence of parts on occupied machines at the start of the cycle and inequality (3.22) we have:

$$S_{[j,m]} \leq S_{[j',m-1]} + \theta_{m-1} \Big| \Big| S_{[j',m]} \leq S_{[j,m-1]} + \theta_{m-1} \tag{3.23}$$

Due to the Lemma 1 and (3.23), we can derive the operational sequence constraints as follows:

$$\begin{cases} S_{[j,m]} - BM \times \left(3 - Y_{[j,m],[j',m-1]} - X_j - X_{j'}\right) \leq S_{[j',m-1]} + \theta_{m-1} \\ S_{[j',m-1]} - BM \times \left(2 + Y_{[j,m],[j',m-1]} - X_j - X_{j'}\right) \leq S_{[j,m]} + \theta_m \\ \forall \{j, j', m | \, j, j' \in \{1,2,\dots,J\};\ m \in \{1,\dots,IM-1\}\} \end{cases}, \quad (3.24)$$

where the expression in the parentheses at the left side of the inequalities guarantees that the considered parts have entered the system during the cycle and it also determines the operational sequence for these parts using the variable $Y_{[j,m],[j',m-1]}$.

*3.4.1.6 Operational Sequence Constraints for the Robots*

There are a limited number of robots to perform the transportation operations in the considered problem and conflictions in using the robots should be prevented. To determine the sequence of transportation operations assigned to the same robot, the following inequalities (3.25) are proposed.

$$S_{[j,m]} + \theta_m + \delta_{[\![m+1,m']\!]} \leq S_{[j',m']} \Big| \Big| S_{[j',m']} + \theta_{m'} + \delta_{[\![m'+1,m]\!]} \leq S_{[j,m]} \quad (3.25)$$

Where both $[j, m]$ and $[j', m']$ transportation operations would be performed by the same robot. As can be seen from Figure 3.4, whenever a robot performs a transportation operation, it moves to the related machine of its next transportation operation to unload the related part.

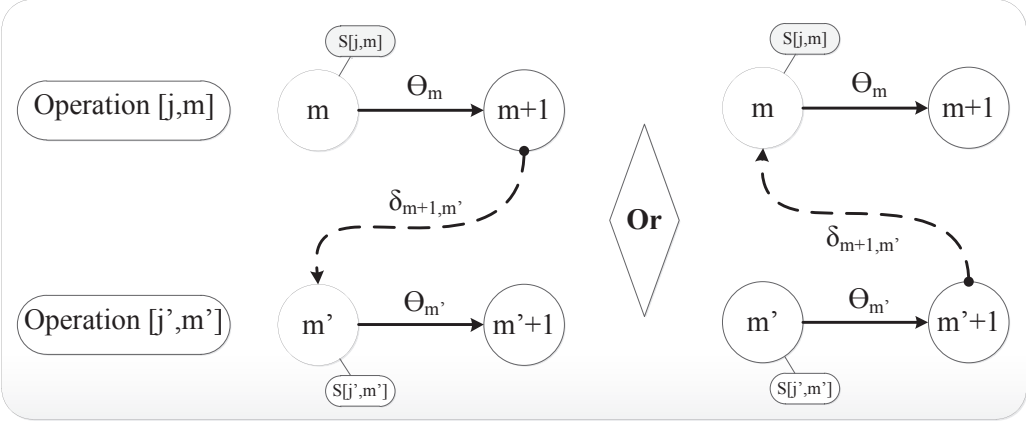



Figure 3.4 The graph representation of the operational sequence of transportation operations performed by the same robot.

Considering the aforementioned relations (3.25) and Lemma 1, constraint set (3.26) determines the sequence of transportation operations assigned to the same robot.

$$\begin{cases} S_{[j,m]} + \theta_m + \delta_{[\![m+1,m']\!]} - BM \times \left(5 - Y_{[j,m],[j',m']} - Z_{[j,m],r} - Z_{[j',m'],r} - X_j - X_{j'}\right) \le S_{[j',m']} \\ S_{[j',m']} + \theta_{m'} + \delta_{[\![m'+1,m]\!]} - BM \times \left(4 + Y_{[j,m],[j',m']} - Z_{[j,m],r} - Z_{[j',m'],r} - X_j - X_{j'}\right) \le S_{[j,m]}, \\ \forall \{j, j', m, m', r \mid j, j' \in \{1,2,\dots,J\};\ m, m' \in \{1,\dots,IM-1\};\ (m < m');\ r \in \{1,2,\dots,R\}\} \end{cases} \tag{3.26}$$

where the variable $Y_{[j,m],[j',m']}$ inside the parentheses determines the sequence of $[j,m]$ and $[j',m']$ operations which have been assigned to the same robot $r$ ($Z_{[j,m],r} = Z_{[j',m'],r} = 1$), while the entrance of part classes $j$ and $j'$ during the cycle is already ensured.

*3.4.1.7 Cyclic Machine Availability Constraints*

Due to the cyclic characteristic of the robotic cell, there would be occupied machines at the beginning of the cycle. At any time, a machine should be occupied just by one part. Figure 3.5 provides an example that briefly illustrates the influence of the states of the machines on the sequencing of transportation operations.

**Lemma 2**

If machine $m$ is not occupied at the start time of the $g$-degree cycle:

$$\begin{cases} \sum_{j=1}^{g} Y_{[j,m-1],[j,m]} = g \\ \\ \forall \{ m \,|m \in \{2, \dots, IM-1\}\} \\ m \longrightarrow not\ occupied \end{cases} \tag{3.27}$$

**Proof 2**

Considering that machine $m$ is not occupied at the beginning of the $g$-degree cycle, then:

$$\begin{cases} Y_{[j,m-1],[j,m]} = 1 \\ \\ \forall \{j \,|j \in \{1,2, \dots, g\}\} \\ \forall \{ m \,|m \in \{2, \dots, IM-1\}\} \\ m \longrightarrow not\ occupied \end{cases} \Longrightarrow \begin{cases} \sum_{j=1}^{g} Y_{[j,m-1],[j,m]} = g \\ \\ \forall \{ m \,|m \in \{2, \dots, IM-1\}\} \\ m \longrightarrow not\ occupied \end{cases} \tag{3.28}$$

**Lemma 3**

If machine $m$ is occupied at the start time of the $g$-degree cycle, then:

$$\begin{cases} \sum_{j=1}^{g} Y_{[j,m-1],[j,m]} = g-1 \\ \\ \forall \{ m \,|m \in \{2, \dots, IM-1\}\} \\ m \longrightarrow occupied \end{cases} \tag{3.29}$$

**Proof 3**

If machine $m$ is occupied at the start time of the $g$-degree cycle by the $p^{th}$ part ($p \in \{1,2, \dots, g\}$) then the transportation operation $[p, m]$ proceeds $[p, m-1]$, then:

$$\left.\begin{array}{l} Y_{[p,m-1],[p,m]} = 0 \\ \\ \begin{cases} Y_{[j,m-1],[j,m]} = 1 \\ \forall \{j \,|j \in \{1,2, \dots, g\}; j \neq p\} \end{cases} \\ \\ m \longrightarrow occupied\ \ by\ (p) \end{array}\right\} \Longrightarrow \begin{cases} \sum_{j=1}^{g} Y_{[j,m-1],[j,m]} = g-1 \\ \\ \forall \{ m \,|m \in \{2, \dots, IM-1\}\} \\ m \longrightarrow occupied \end{cases} \tag{3.30}$$

Considering Lemma 2 and Lemma 3, while $g$ part classes enter the system during a cycle, we can derive,

$$\begin{cases} \sum_{j=1}^{g} Y_{[j,m-1],[j,m]} \geq g-1 \\ \forall \{ m \mid m \in \{2, \dots, IM-1\}\} \end{cases} \tag{3.31}$$

Furthermore, the number of $g$ part classes in (3.31) is considered as a variable that would be optimized. Therefore, the constraint set (3.32) has been proposed to state (3.31) which guarantees the entrance of $j$ number of parts to the cycle.

$$\begin{cases} \sum_{j'=1}^{j} Y_{[j',m-1],[j',m]} + BM \times \left| j - \sum_{j''=1}^{J} X_{j''} \right| \geq j-1 \\ \forall \{j, m \mid j \in \{1,2, \dots, J\}; m \in \{2, \dots, IM-1\}\} \end{cases} \tag{3.32}$$

Where the absolute value function has been used to determine the degree of related cycle.

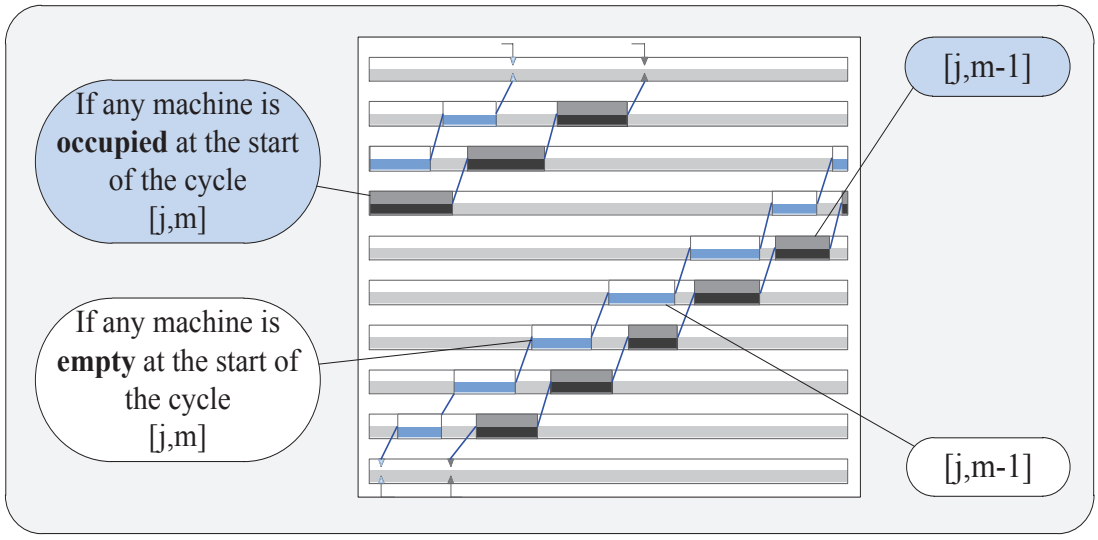



Figure 3.5 The cyclic machine availability of machines between two successive parts.

**Lemma 4**

If machine $m$ is not occupied at the start time of the $g$-degree cycle:

$$\begin{cases} \sum_{j=1}^{g} Y_{[(j\%g)+1,m-1],[j,m]} = 1 \\ (j\%g) \equiv Mod(j,g) \\ m \longrightarrow not\ occupied \end{cases} \tag{3.33}$$

**Proof 4**

Considering a cycle of $g$-degree, where machine $m$ is not occupied at the start time and there are $m'$ number of machines occupied at the start time of the cycle before machine $m$, then:

$$\begin{cases} m' = 0 \quad \longrightarrow & \begin{cases} j = 1 & \longrightarrow Y_{[2,m-1],[1,m]} = 0 \\ j = 2 & \longrightarrow Y_{[3,m-1],[2,m]} = 0 \\ \vdots & \vdots \quad\quad \vdots \\ j = g & \longrightarrow Y_{[1,m-1],[g,m]} = 1 \end{cases} \\ m' = 1 \quad \longrightarrow & \begin{cases} j = 1 & \longrightarrow Y_{[2,m-1],[1,m]} = 0 \\ j = 2 & \longrightarrow Y_{[3,m-1],[2,m]} = 0 \\ \vdots & \vdots \quad\quad \vdots \\ j = g-1 & \longrightarrow Y_{[g,m-1],[g-1,m]} = 1 \\ j = g & \longrightarrow Y_{[1,m-1],[g,m]} = 0 \end{cases} \\ \vdots \quad\quad \vdots & \vdots \quad\quad\quad\quad \vdots \\ m' = h \quad \longrightarrow & \begin{cases} j = 1 & \longrightarrow Y_{[2,m-1],[1,m]} = 0 \\ j = 2 & \longrightarrow Y_{[3,m-1],[2,m]} = 0 \\ \vdots & \vdots \quad\quad \vdots \\ j = g-h & \longrightarrow Y_{[(g-h)+1,m-1],[g-h,m]} = 1 \\ \vdots & \vdots \quad\quad \vdots \\ j = g & \longrightarrow Y_{[1,m-1],[g,m]} = 0 \end{cases} \end{cases} \tag{3.34}$$

**Lemma 5**

If machine $m$ is occupied at the start time of the $g$-degree cycle:

$$\begin{cases} \sum_{j=1}^{g} Y_{[(j\%g)+1,m-1],[j,m]} = 0 \\ (j\%g) \equiv Mod(j,g) \\ m \longrightarrow occupied \end{cases} \tag{3.35}$$

**Proof 5**

Considering a $g$-degree cycle, if machine $m$ is occupied at the start time of the $g$-degree cycle:

$$\begin{cases} j = 1 & \longrightarrow Y_{[2,m-1],[1,m]} = 0 \\ j = 2 & \longrightarrow Y_{[3,m-1],[2,m]} = 0 \\ \vdots & \vdots \qquad\qquad \vdots \\ j = g & \longrightarrow Y_{[1,m-1],[g,m]} = 0 \end{cases} \tag{3.36}$$

**Lemma 6**

Considering a $g$-degree cycle:

$$\begin{cases} \sum_{j=1}^{g} \left( Y_{[(j\%g)+1,m-1],[(j\%g)+1,m]} - Y_{[(j\%g)+1,m-1],[j,m]} \right) = g - 1 \\ \forall \{m |\, m \in \{2, \dots, IM-1\}\} \\ (j\%g) \equiv Mod(j,g) \end{cases} \tag{3.37}$$

**Proof 6**

It could be derived due to Lemma 2, Lemma 3, Lemma 4, and Lemma 5. The machines set could be divided to free and occupied machines at the start time of the cycle. According to Lemma 2 and Lemma 4, the following relation for the free machines at the start time of the cycle can be stated:

$$\sum_{j=1}^{g} Y_{[(j\%g)+1,m-1],[(j\%g)+1,m]} - \sum_{j=1}^{g} Y_{[(j\%g)+1,m-1],[j,m]} = g - 1 \tag{3.38}$$

According to Lemma 3 and Lemma 5, the following relation for the occupied machines at the start time of the cycle can be stated:

$$\sum_{j=1}^{g} Y_{[(j\%g)+1,m-1],[(j\%g)+1,m]} - \sum_{j=1}^{g} Y_{[(j\%g)+1,m-1],[j,m]} = (g-1) - 0 \qquad (3.39)$$

Furthermore, the number of $g$ parts in Lemma 6 is considered as a variable that is tried to be optimized. Therefore, the constraint set (3.40) has been proposed to state Lemma 6 which guarantees the entrance of $j$ number of parts to the cycle.

$$\begin{cases} \left|(j-1) - \sum_{j'=1}^{j} \left(Y_{[(j'\%j)+1,m-1],[(j'\%j)+1,m]} - Y_{[(j'\%j)+1,m-1],[j',m]}\right)\right| \le BM \times \left|j - \sum_{j''=1}^{J} X_{j''}\right| \\ \forall \{j,m | \, j \in \{1,2,\dots,J\}, m \in \{2,\dots,IM-1\}\}; where \; (j'\%j) \equiv Mod(j',j) \end{cases} \qquad (3.40)$$

Where the right side of the constraint set (3.40) states Lemma 6 for the *j*-degree cycle.

### *3.4.2 Linearization of the Proposed Model*

The proposed model is a nonlinear integer programming model because of the nonlinear term in the objective function. Therefore, additional constraints set are used to linearize the objective function of the model. The following equation (3.41) determines the *k*-degree of cycle if $k = j$ parts enter the system during a cycle.

$$\left|j - \sum_{j'=1}^{J} X_{j'}\right| = 0 \qquad (3.41)$$

In addition,

$$\Phi = (T/j) \quad \Longrightarrow \quad |\Phi - (T/j)| \le 0 \qquad (3.42)$$

By considering the expressions (3.41) and (3.42), the following constraint set (3.44) should be added to the mathematical model:

$$Minimize: \Phi \tag{3.43}$$

*Subject to:*

Constraint sets (2) - (11) and new constraint set (13):

$$\begin{cases} |\Phi - (T/j)| - BM \times \left| j - \sum_{j'=1}^{J} X_{j'} \right| \leq 0 \\ \forall \{j | j \in \{1,2,\dots,J\}\} \end{cases} \tag{3.44}$$

The absolute value function is a piecewise linear function which have been used in constraints sets (3.32), (3.40), and (3.44). Table 3.2 provides examples for the transformation of absolute value function for the two constraint forms that use absolute value function in the proposed model.

Table 3.2 Absolute value function transformations for two constraint forms.

| ***Example Constraint form:*** | *Constraint form (1)* | *Constraint form (2)* |
|---|---|---|
| | $x - BM \times \lvert a - b \rvert \leq 0$ | $\lvert x - y \rvert - BM \times \lvert a - b \rvert \leq 0$ |
| ***Transformed to:*** | $x - BM \times (p + q) \leq 0$<br>$a - b - p + q = 0$<br>$p - BM \times e \leq 0$<br>$q - BM \times (1 - e) \leq 0$ | $(p' + q') - BM \times (p + q) \leq 0$<br>$x - y - p' + q' = 0$<br>$a - b - p + q = 0$<br>$p - BM \times e \leq 0$<br>$q - BM \times (1 - e) \leq 0$ |
| $x, y, a, b, p, q, p', q' \geq 0$ ; $e \in \{0,1\}$; $BM$ ($Big\ M$)*: Sufficiently large value* | | |

Considering the transformation method provided in Table 3.2, additional variables are defined to entirely linearize the constraint sets (3.32), (3.40), and (3.44) that include absolute value functions as follows (Table 3.3).

Table 3.3 Additional variables defined for absolute value function transformation

| ***Binary variables*** | ***Nonnegative variables*** |
|---|---|
| $e_j$ | $a_j$ , $b_j$ , $v_j$ , $w_j$ , $v'_j$ , $w'_j$ |
| $\forall \{j \mid j \in \{1,2,\dots,J\}\}$ | $\forall \{j \mid j \in \{1,2,\dots,J\}\}$ |

Hence, the MILP model is proposed as follows after transforming the absolute value function included constraints sets:

$$Minimize: \Phi \tag{3.45}$$

*S.t.:*

$$\left\{\sum_{j=1}^{J} X_j \geq 1\right. \tag{3.46}$$

$$\begin{cases} \left(j \times X_j\right) \leq \sum_{j'=1}^{j} X_{j'} \\ \forall \{j \mid j \in \{1,2,\dots,J\}\} \end{cases} \tag{3.47}$$

$$\begin{cases} S_{[j,m]} - BM \times X_j \leq 0 \\ \forall \{j,m \mid j \in \{1,2,\dots,J\};\ m \in \{1,\dots,IM-1\}\} \end{cases} \tag{3.48}$$

$$\begin{cases} S_{[j,m-1]} + \theta_{m-1} + PLo_m - BM \times \left(2 - Y_{[j,m-1],[j,m]} - X_j\right) \leq S_{[j,m]} \\ S_{[j,m-1]} + \theta_{m-1} + PLo_m - BM \times \left(1 + Y_{[j,m-1],[j,m]} - X_j\right) \leq S_{[j,m]} + T \\ \forall \left\{j,m \mid j \in \{1,2,\dots,J\};\ m \in \{2,\dots,IM-1\}\right\} \end{cases} \tag{3.49}$$

$$\begin{cases} S_{[j,m-1]} + \theta_{m-1} + PUp_m + BM \times \left(2 - Y_{[j,m-1],[j,m]} - X_j\right) \geq S_{[j,m]} \\ S_{[j,m-1]} + \theta_{m-1} + PUp_m + BM \times \left(1 + Y_{[j,m-1],[j,m]} - X_j\right) \geq S_{[j,m]} + T \\ \forall \left\{j,m \mid j \in \{1,2,\dots,J\};\ m \in \{2,\dots,IM-1\}\right\} \end{cases} \tag{3.50}$$

$$\begin{cases} \sum_{r=1}^{R} Z_{[j,m],r} = X_j \\ \forall \left\{j,m \mid j \in \{1,2,\dots,J\};\ m \in \{1,\dots,IM-1\}\right\} \end{cases} \tag{3.51}$$

$$\begin{cases} S_{[j,m]} - BM \times \left(3 - Y_{[j,m],[j',m-1]} - X_j - X_{j'}\right) \leq S_{[j',m-1]} + \theta_{m-1} \\ S_{[j',m-1]} - BM \times \left(2 + Y_{[j,m],[j',m-1]} - X_j - X_{j'}\right) \leq S_{[j,m]} + \theta_m \\ \forall \{j,j',m \mid j,j' \in \{1,2,\dots,J\};\ m \in \{1,\dots,IM-1\}\} \end{cases} \tag{3.52}$$

$$\begin{cases} S_{[j,m]} + \theta_m + \delta_{[\![m+1,m']\!]} - BM \times \left(5 - Y_{[j,m],[j',m']} - Z_{[j,m],r} - Z_{[j',m'],r} - X_j - X_{j'}\right) \leq S_{[j',m']} \\ S_{[j',m']} + \theta_{m'} + \delta_{[\![m'+1,m]\!]} - BM \times \left(4 + Y_{[j,m],[j',m']} - Z_{[j,m],r} - Z_{[j',m'],r} - X_j - X_{j'}\right) \leq S_{[j,m]} \\ \forall \{j,j',m,m',r \mid j,j' \in \{1,2,\dots,J\};\ m,m' \in \{1,\dots,IM-1\};(m < m');\ r \in \{1,2,\dots,R\}\} \end{cases} \tag{3.53}$$

$$\begin{cases} \sum_{j'=1}^{j} Y_{[j',m-1],[j',m]} + BM \times \left(a_j + b_j\right) \geq j - 1 \\ \forall \{j,m \mid j \in \{1,2,\dots,J\}; m \in \{2,\dots,IM-1\}\} \end{cases} \tag{3.54}$$

$$\begin{cases} (j-1) - \sum_{j'=1;\,(j'<j \,\wedge\, j"=j'+1) \vee (j'=j \,\wedge\, j"=1)}^{j} \left( Y_{[j",m-1],[j",m]} - Y_{[j",m-1],[j',m]} \right) - v_j + w_j = 0 \\ \forall \{j, m \mid j \in \{1,2,\dots,J\}, m \in \{2,\dots,IM-1\}\} \end{cases} \tag{3.55}$$

$$\begin{cases} \Phi - (T/j) - v'_j + w'_j = 0 \\ \forall \{j \mid j \in \{1,2,\dots,J\}\} \end{cases} \tag{3.56}$$

$$\begin{cases} j - \sum_{j'=1}^{j} X_{j'} - a + b = 0 \\ \forall \{j \mid j \in \{1,2,\dots,J\}\} \end{cases} \tag{3.57}$$

$$\begin{cases} v_j + w_j - BM \times (a_j + b_j) \le 0 \\ v'_j + w'_j - BM \times (a_j + b_j) \le 0 \\ \forall \{j \mid j \in \{1,2,\dots,J\}\} \end{cases} \tag{3.58}$$

$$\begin{cases} a_j - BM \times e_j \le 0 \\ b_j - BM \times (1 - e_j) \le 0 \\ \forall \{j \mid j \in \{1,2,\dots,J\}\} \end{cases} \tag{3.59}$$

### 3.5 Instance Problem

An instance problem with two robots, eight machines, and input, and output devices is solved optimally using the proposed MILP model. The related data including the lower and upper bounds for the processing times on the machines are given in Table 3.4. The characteristics of the *k*DCFS-RCS problem are described using the feasible solution found.

According to the given data, the maximum number of parts completed per cycle is four, which gives the degree of the cyclic schedule and is also assumed to be the number of parts entering per cycle. Table 3.5 and Table 3.6 provide the start times of all transportation operations and their related robot assignments. Figure 3.6 illustrates the detailed Gantt chart of the solution where the arrows demonstrate the sequence of transportation operations and their assigned robots. The solid and dashed arrows

demonstrate the complete and void moves, respectively. Additionally, Figure 3.7 demonstrates the Gantt chart for three consecutive cycles on which the processing operations of the four-degree cycle are illustrated.

Table 3.4 The related data of the considered instance problem.

| **Data (sec.)** | | **Machines** | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
| $PLo_m$ | | 0 | 42 | 59 | 47 | 63 | 50 | 43 | 74 | 51 | 0 |
| $PUp_m$ | | 0 | 65 | 100 | 91 | 83 | 67 | 86 | 114 | 73 | 0 |
| $\theta_m$ | | 15 | 13 | 16 | 17 | 15 | 12 | 15 | 12 | 16 | - |
| *Empty move* $(\delta_{[m,n]})$ | ***Machines*** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
| | **1** | 0 | 3 | 6 | 9 | 12 | 15 | 18 | 21 | 24 | 27 |
| | **2** | 3 | 0 | 3 | 6 | 9 | 12 | 15 | 18 | 21 | 24 |
| | **3** | 6 | 3 | 0 | 3 | 6 | 9 | 12 | 15 | 18 | 21 |
| | **4** | 9 | 6 | 3 | 0 | 3 | 6 | 9 | 12 | 15 | 18 |
| | **5** | 12 | 9 | 6 | 3 | 0 | 3 | 6 | 9 | 12 | 15 |
| | **6** | 15 | 12 | 9 | 6 | 3 | 0 | 3 | 6 | 9 | 12 |
| | **7** | 18 | 15 | 12 | 9 | 6 | 3 | 0 | 3 | 6 | 9 |
| | **8** | 21 | 18 | 15 | 12 | 9 | 6 | 3 | 0 | 3 | 6 |
| | **9** | 24 | 21 | 18 | 15 | 12 | 9 | 6 | 3 | 0 | 3 |
| | **10** | 27 | 24 | 21 | 18 | 15 | 12 | 9 | 6 | 3 | 0 |
| ***Number of Robots***: 2 | | | | | | | | | | | |

Table 3.5 The start time of all transportation operations found for the instance problem

| **Parts** | **Machine** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| **1** | 12 | 69 | 141 | 214 | 295 | 377 | 432 | 59 | 122 |
| **2** | 88 | 162 | 234 | 297 | 377 | 444 | 80 | 169 | 249 |
| **3** | 181 | 256 | 333 | 398 | 36 | 101 | 187 | 316 | 401 |
| **4** | 275 | 355 | 6 | 113 | 205 | 277 | 334 | 423 | 37 |
| The total cycle time: $T = 462$ sec. | | | | | | | | | |
| The cycle time per part (mean cycle time) : $\Phi = 115.5$<br>The throughput rate: $\mu = 0.5194$ parts/min | | | | | | | | | |

Table 3.6 The assigned robots to perform each transportation operation of the instance problem

| **TO Class-1** | **Robots** | | **TO Class-2** | **Robots** | | **TO Class-3** | **Robots** | | **TO Class-4** | **Robots** | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | **1** | **2** | | **1** | **2** | | **1** | **2** | | **1** | **2** |
| [1,1] | | ✓ | [2,1] | | ✓ | [3,1] | ✓ | | [4,1] | ✓ | |
| [1,2] | | ✓ | [2,2] | ✓ | | [3,2] | ✓ | | [4,2] | ✓ | |
| [1,3] | | ✓ | [2,3] | ✓ | | [3,3] | ✓ | | [4,3] | ✓ | |
| [1,4] | | ✓ | [2,4] | ✓ | | [3,4] | ✓ | | [4,4] | | ✓ |
| [1,5] | | ✓ | [2,5] | | ✓ | [3,5] | | ✓ | [4,5] | ✓ | |
| [1,6] | ✓ | | [2,6] | | ✓ | [3,6] | ✓ | | [4,6] | | ✓ |
| [1,7] | ✓ | | [2,7] | ✓ | | [3,7] | | ✓ | [4,7] | | ✓ |
| [1,8] | ✓ | | [2,8] | | ✓ | [3,8] | | ✓ | [4,8] | | ✓ |
| [1,9] | ✓ | | [2,9] | | ✓ | [3,9] | | ✓ | [4,9] | ✓ | |

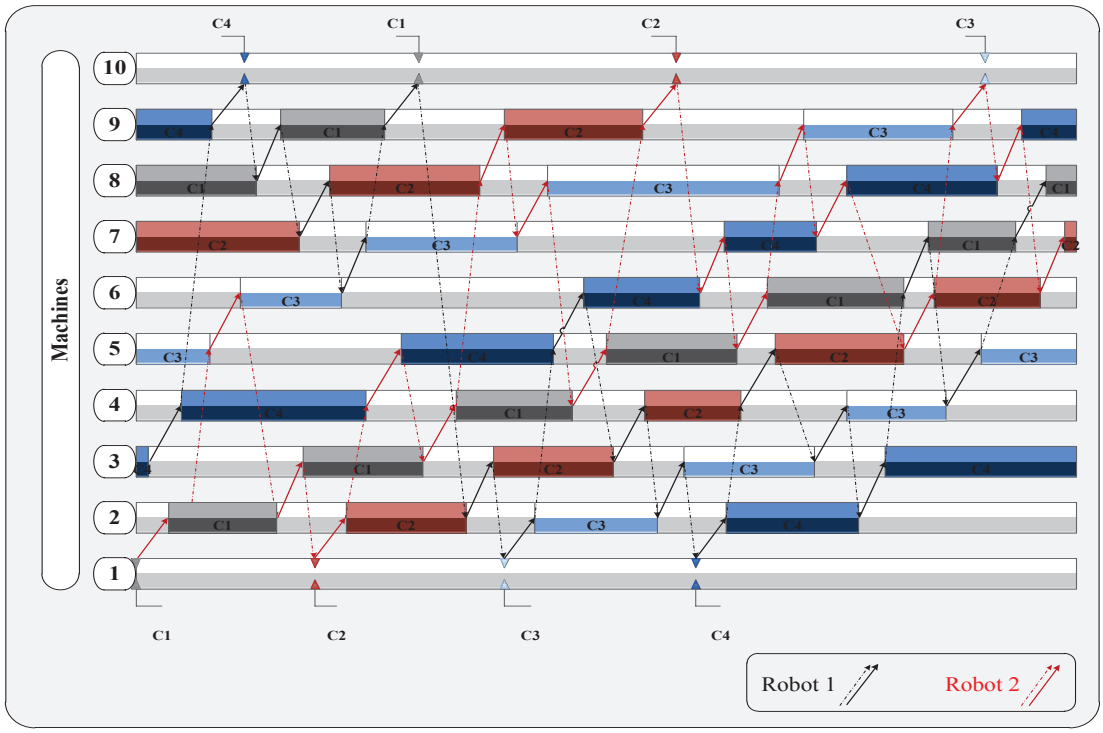



Figure 3.6. The Gantt chart of the instance problem indicating the sequence of all transportation operations and their assigned robots.

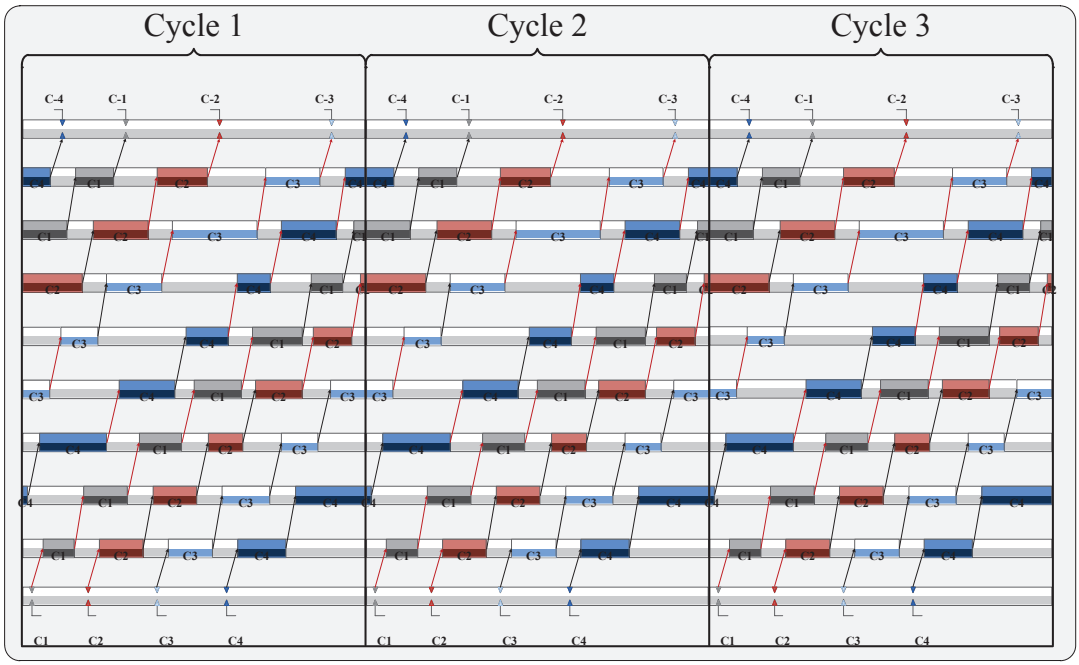



Figure 3.7 The Gantt chart of three consecutive cycles for the instance problem.

In addition, the graph model representation of the solution is illustrated in Figure 3.8. The conjunctive and disjunctive arrows demonstrate the transportation operations and empty moves of robots, respectively. The numbers on the conjunctive arrows indicate the sequence of transportation operations performed by each robot.

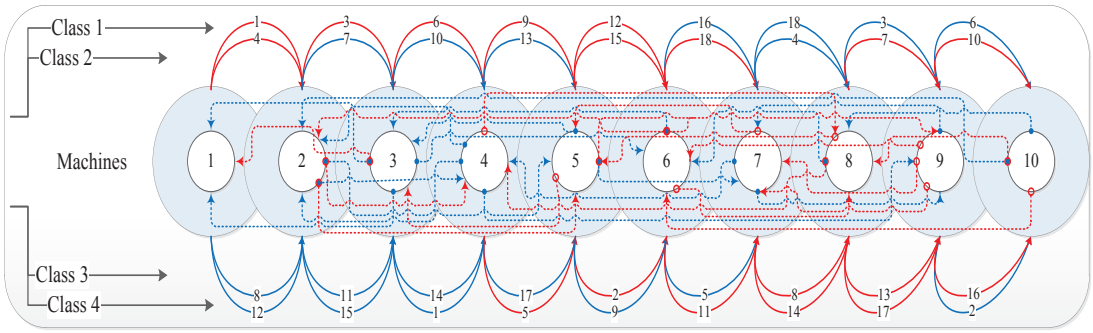



Figure 3.8 The graph model representation of processing and transportation operations for the instance problem.

### 3.6 Computational Results

In order to assess the empirical performance of the proposed MILP model, randomly generated and benchmark problem instances have been solved. The proposed model has been formulated and solved using GAMS 23.7 software on a PC including Intel(R) Core(TM)-i7 and 6 GB RAM. Table 3.7 shows the sizes of the test problems and the solution statistics of the proposed model for the two cases: In the first case, the problem instances have been solved for a fixed *k*-degree cyclic schedule; and the *k* value has been taken as 1, 2, 3, and 4, respectively. In the second case, the degree (*k*) of the cyclic schedule has been considered as a variable and an upper limit, four, has been given for the *k* value. To compare the obtained solutions with previous studies two problem sets (Copper-13 machines and Zinc-17 machines) provided by Leung, Zhang, Yang, Mak & Lam (2004) have been solved by the proposed model. The proposed model by Leung et al. Leung, et al., 2004 has solved the mentioned problems for one-degree cyclic schedules considering the hoists as part moving devices instead of robots, and the number of hoists has been taken as for 1, 2, and 3, respectively.

The different optimal cycle times obtained per produced part and throughput rates for different values of *k*-degree confirm the importance of proposed MILP model. Table 3.7 shows that different degree cyclic schedules may cause different cycle times ($\Phi$), and so indicates the importance of determining the optimal *k*-degree. For the first set of randomly generated problems ($IM = 10$), the optimal throughput rates for different number of robots ($R = 1, 2,$ and 3) are obtained with $k = 3, 1,$ and 3 degree cycles as $\mu = 0.151,\ 0.333,$ and 0.407, respectively. Also, for the second set of randomly generated problems ($IM = 10$), the optimal throughput rates for different number of robots ($R = 1, 2,$ and 3) are obtained with $k = 3,\ 1,$ and 4 degree cycles as $\mu = 0.151,\ 0.340,$ and $0.421$, respectively. It is apparent from Table 3.7 that different number of robots affects the optimal *k*-degree, cycle time, and throughput rate. In addition, the proposed MILP model found the optimal solutions to the problem sets Copper ($IM = 13$) and Zinc ($IM = 17$) as in Leung et al. Leung, et al., 2004 for one-degree cycles.

Moreover, Table 3.8 provides the MILP model size statistics for the test problems. As can be seen from the table, the number of variables and constraints increases drastically by increasing the number of *k*-degree levels. Furthermore, the run time of the last two sets of problems (Copper-13 and Zinc-17) can be compared with the smaller size data sets in Table 3.7. Here, the problems with *k=4* and *R=3* values from the Zinc ($IM = 17$) and the first problem sets ($IM = 10$) have been solved in 9421.47 and 21495.61 seconds, respectively. Table 3.7 and Table 3.8 indicate that the proposed MILP model has solved the Zinc problem ($IM = 17$) with 15946 number of constraints in much shorter time compared to the first set problem ($IM = 10$) with 4942 number of constraints.

Therefore, it can be concluded that the provided data as processing times and the difference between lower and upper bounds of the time windows for processing times influences the solution time for the *k*DCFS-RCS problem.

Table 3.7 (1) The proposed MILP model statistics for the randomly generated instance problems.

| Problems | | | *Fixed k-degree MILP Model* | | | | | *Variable k-degree MILP Model* | | |
|---|---|---|---|---|---|---|---|---|---|---|
| *IM* | *R* | *k-degree* | *Φ (sec)* | *T (sec)* | *GAP* | *μ (part/min)* | *Run Time (sec)* | *k-degree* | *μ (part/min)* | *Run Time (sec)* |
| **10***(randomly generated 1)* | **1** | **1** | 458.00 | 458 | 0 % | 0.131 | 0.03 | | | |
| | | **2** | 425.50 | 851 | 0 % | 0.141 | 0.27 | | | |
| | | **3** | **397.33** | **1192** | 0 % | **0.151** | **2.18** | **3** | **0.151** | **55.91** |
| | | **4** | 422.50 | 1690 | 0 % | 0.142 | 10.19 | | | |
| | **2** | **1** | **180.00** | **180** | 0 % | **0.333** | **0.30** | **1** | **0.333** | **12175.31** |
| | | **2** | 180.00 | 360 | 0 % | 0.333 | 14.88 | | | |
| | | **3** | 180.00 | 540 | 0 % | 0.333 | 434.06 | | | |
| | | **4** | 180.00 | 720 | 0 % | 0.333 | 10821.42 | | | |
| | **3** | **1** | 161.00 | 161 | 0 % | 0.372 | 0.56 | | | |
| | | **2** | 148.50 | 297 | 0 % | 0.404 | 391.75 | | | |
| | | **3** | **147.33** | **442** | 0 % | **0.407** | **7551.29** | **3** | **0.407** | **20732.19** |
| | | **4** | 148.50 | 594 | 0 % | 0.404 | 21495.61 | | | |
| **10***(randomly generated 2)* | **1** | **1** | 442.00 | 442 | 0 % | 0.135 | 0.05 | | | |
| | | **2** | 415.00 | 830 | 0 % | 0.144 | 0.17 | | | |
| | | **3** | **395.66** | **1187** | 0 % | **0.151** | **2.53** | **3** | **0.151** | **11.41** |
| | | **4** | 410.50 | 1642 | 0 % | 0.146 | 8.30 | | | |
| | **2** | **1** | **176.00** | **176** | 0 % | **0.340** | **0.72** | **1** | **0.340** | **10821.74** |
| | | **2** | 211.00 | 422 | 0 % | 0.284 | 8.21 | | | |
| | | **3** | 193.33 | 580 | 0 % | 0.310 | 897.41 | | | |
| | | **4** | 189.50 | 758 | 0 % | 0.316 | 12258.73 | | | |
| | **3** | **1** | 148.00 | 148 | 0 % | 0.405 | 0.30 | | | |
| | | **2** | 147.50 | 295 | 0 % | 0.406 | 17.11 | | | |
| | | **3** | 145.33 | 436 | 0 % | 0.412 | 6151.24 | | | |
| | | **4** | **142.25** | **569** | 0 % | **0.421** | **19447.12** | **4** | **0.421** | **21835.83** |
| **14***(randomly generated 3)* | **1** | **1** | 1291.00 | 1291 | 0 % | 0.046 | 0.08 | | | |
| | | **2** | **1214.50** | **2429** | 0 % | **0.049** | **2.46** | **2** | **0.049** | **45.41** |
| | | **3** | 1240.00 | 3720 | 0 % | 0.048 | 11.37 | | | |
| | | **4** | 1238.75 | 4955 | 0 % | 0.048 | 46.33 | | | |
| | **2** | **1** | 521 | 521 | 0 % | 0.115 | 4.24 | *1* | *0.115* | *20000.00* *Not Optimal* |
| | | **2** | 540 | 1080 | 0 % | 0.111 | 12457.69 | | | |
| | | **3** | *540* | *1620* | *4 %* | *0.111* | *17336.05* | | | |
| | | **4** | *540* | *2160* | *8 %* | *0.111* | *25000.00* | | | |
| | **3** | **1** | 355 | 355 | 0 % | 0.169 | 34.62 | | | |
| | | **2** | 354 | 708 | 0 % | 0.169 | 1511.77 | *2* | *0.169* | *30000.00* *Not Optimal* |
| | | **3** | *359* | *1077* | *7 %* | *0.167* | *25751.83* | | | |
| | | **4** | *361* | *1444* | *10 %* | *0.166* | *26000.00* | | | |

Table 3.7 (2) The proposed MILP model statistics for the randomly generated instance problems.

| Problems | | | *Fixed k-degree MILP Model* | | | | | *Variable k-degree MILP Model* | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | *R* | *k-degree* | *Φ (sec)* | *T (sec)* | *GAP* | *μ (part/min)* | *Run Time (sec)* | *k-degree* | *μ (part/min)* | *Run Time (sec)* |
| **13***(Copper)*Leung, et al., 2004 | **1** | **1** | **1847.20** | **1847.2** | 0 % | **0.032** | **0.03** | **1** | **0.032** | **0.21** |
| | | **2** | 1847.20 | 3694.4 | 0 % | 0.032 | 0.06 | | | |
| | | **3** | 1847.20 | 5541.6 | 0 % | 0.032 | 0.11 | | | |
| | | **4** | 1847.20 | 7388.8 | 0 % | 0.032 | 0.17 | | | |
| | **2** | **1** | **1800.00** | **1800.0** | 0 % | **0.033** | **0.17** | **1** | **0.033** | **3089.17** |
| | | **2** | 1800.00 | 3600.0 | 0 % | 0.033 | 13.27 | | | |
| | | **3** | 1800.00 | 5400.0 | 0 % | 0.033 | 145.89 | | | |
| | | **4** | 1800.00 | 7200.0 | 0 % | 0.033 | 2971.19 | | | |
| | **3** | **1** | **1800.00** | **1800.0** | 0 % | **0.033** | **0.33** | **1** | **0.033** | **1976.64** |
| | | **2** | 1800.00 | 3600.0 | 0 % | 0.033 | 68.33 | | | |
| | | **3** | 1800.00 | 5400.0 | 0 % | 0.033 | 505.04 | | | |
| | | **4** | 1800.00 | 7200.0 | 0 % | 0.033 | 1933.71 | | | |
| **17***(Zinc)*Leung, et al., 2004 | **1** | **1** | **1743.40** | **1743.4** | 0 % | **0.034** | **0.06** | **1** | **0.034** | **0.52** |
| | | **2** | 1743.40 | 3486.8 | 0 % | 0.034 | 0.14 | | | |
| | | **3** | 1743.40 | 5230.2 | 0 % | 0.034 | 0.23 | | | |
| | | **4** | 1743.40 | 6973.6 | 0 % | 0.034 | 0.48 | | | |
| | **2** | **1** | **1680.00** | **1680.0** | 0 % | **0.035** | **0.44** | **1** | **0.035** | **6285.66** |
| | | **2** | 1680.00 | 3360.0 | 0 % | 0.035 | 119.98 | | | |
| | | **3** | 1680.00 | 5040.0 | 0 % | 0.035 | 1335.87 | | | |
| | | **4** | 1680.00 | 6720.0 | 0 % | 0.035 | 6490.19 | | | |
| | **3** | **1** | **1680.00** | **1680.0** | 0 % | **0.035** | **1.68** | **1** | **0.035** | **8835.39** |
| | | **2** | 1680.00 | 3360.0 | 0 % | 0.035 | 168.61 | | | |
| | | **3** | 1680.00 | 5040.0 | 0 % | 0.035 | 5712.74 | | | |
| | | **4** | 1680.00 | 6720.0 | 0 % | 0.035 | 9421.47 | | | |

***Cycle Time: (Φ) ;***
***Total Cycle Time: (T) ;***
***Throughput: (μ);***
$GAP = \left(\frac{Best\ Found - Lower\ Bound}{Best\ Found}\right) \times 100$

***Number of machines: (IM);***
***Number of robots: (R);***

Table 3.8 (1) The proposed MILP model statistics for different problem sizes.

| Instance Problems | | | *Fixed k-degree MILP Model* | | *Variable k-degree MILP Model* | |
|---|---|---|---|---|---|---|
| ***IM*** | ***R*** | ***k-degree*** | ***Variables*** | ***Constraints*** | ***Variables*** | ***Constraints*** |
| **10***(randomly generated 1, 2)* | 1 | 1 | 66 | 229 | 687 | 2638 |
| | | 2 | 201 | 744 | | |
| | | 3 | 408 | 1547 | | |
| | | 4 | 687 | 2638 | | |
| | 2 | 1 | 75 | 301 | 723 | 3790 |
| | | 2 | 219 | 1032 | | |
| | | 3 | 435 | 2195 | | |
| | | 4 | 723 | 3790 | | |
| | 3 | 1 | 84 | 373 | 759 | 4942 |
| | | 2 | 237 | 1320 | | |
| | | 3 | 462 | 2843 | | |
| | | 4 | 759 | 4942 | | |
| **14***(randomly generated 3)* | 1 | 1 | 116 | 433 | 1391 | 5470 |
| | | 2 | 385 | 1488 | | |
| | | 3 | 810 | 3167 | | |
| | | 4 | 1391 | 5470 | | |
| | 2 | 1 | 129 | 589 | 1443 | 7966 |
| | | 2 | 411 | 2112 | | |
| | | 3 | 849 | 4571 | | |
| | | 4 | 1443 | 7966 | | |
| | 3 | 1 | 142 | 745 | 1495 | 10462 |
| | | 2 | 437 | 2736 | | |
| | | 3 | 888 | 5975 | | |
| | | 4 | 1495 | 10463 | | |

Table 3.8 (2) The proposed MILP model statistics for different problem sizes.

| Instance Problems | | | *Fixed k-degree MILP Model* | | *Variable k-degree MILP Model* | |
|---|---|---|---|---|---|---|
| ***IM*** | ***R*** | ***k-degree*** | ***Variables*** | ***Constraints*** | ***Variables*** | ***Constraints*** |
| **13***(Copper)*Leung, et al., 2004 | 1 | 1 | 102 | 377 | 1191 | 4666 |
| | | 2 | 333 | 1279 | | |
| | | 3 | 696 | 2708 | | |
| | | 4 | 1191 | 4666 | | |
| | 2 | 1 | 114 | 508 | 1239 | 6778 |
| | | 2 | 357 | 1806 | | |
| | | 3 | 732 | 3896 | | |
| | | 4 | 1239 | 6778 | | |
| | 3 | 1 | 126 | 640 | 1287 | 8890 |
| | | 2 | 381 | 2334 | | |
| | | 3 | 768 | 5084 | | |
| | | 4 | 1287 | 8890 | | |
| **17***(Zinc)*Leung, et al., 2004 | 1 | 1 | 164 | 628 | 2087 | 8266 |
| | | 2 | 565 | 2214 | | |
| | | 3 | 1206 | 4760 | | |
| | | 4 | 2087 | 8266 | | |
| | 2 | 1 | 180 | 868 | 2151 | 12106 |
| | | 2 | 597 | 3174 | | |
| | | 3 | 1254 | 6920 | | |
| | | 4 | 2151 | 12106 | | |
| | 3 | 1 | 196 | 1108 | 2215 | 15946 |
| | | 2 | 629 | 4134 | | |
| | | 3 | 1302 | 9080 | | |
| | | 4 | 2215 | 15946 | | |

## 3.7 Conclusion

This is the first research that studies the *k*-degree cyclic flow shop robotic cell scheduling (*k*DCFS-RCS) problem considering multiple single gripper robots with time window processing times. A novel mixed-integer linear programming (MILP) model is proposed for the *k*DCFS-RCS problem to minimize the cycle time for per produced part and hence to maximize the throughput rate. Besides, the special characteristics of *k*DCFS-RCS problem are stated by an instance problem given. The computational results confirm the efficiency of different *k*-degrees on cycle time for per produced part and on throughput rate. According to the computational study conducted for the proposed model, the optimal *k*-degree and cycle time can be found in reasonable computation time for realistic problem sizes, and this feature makes it applicable to real-life problems.

# CHAPTER FOUR
# MULTI-DEGREE CYCLIC FLOW SHOP ROBOTIC CELL SCHEDULING PROBLEM: ANT COLONY OPTIMIZATION

## 4.1 Introduction

In this research, we focus on the sequencing and scheduling problem of a particular type of automated material handling system in cellular manufacturing: cyclic robotic cells. A robotic flow shop usually consists of an input device, an output device, a sequence of processing machines, and a crew of identical programmable robots. Parts to be processed enter the system from the input device, they are processed successively through the machines in series, and finally leave the robotic cell from the output device. Each machine can process only one part at a time. Because there are no buffers for intermediate storage between the machines and each machine can hold only one part, a robotic cell is, in essence, a flow shop with blocking, where there are single gripper robots that transfer the parts between machines.

In this chapter, the multi-robot multi-degree cyclic flow shop robotic cell scheduling problem is considered, in which the same type of parts are produced in large quantities. In cyclic scheduling, the robots perform a series of material handling operations in a repetitive way in order to achieve the production target. In a single-degree cyclic schedule, a single part is completed when the robot or the robots complete the sequence of move operations. Whereas, in a multi-degree cyclic schedule ($k$-degree cyclic schedule), $k$ parts are completed when the robot or the robots complete the sequence of move operations. Although shorter cycle times can be achieved with multi-degree cyclic schedules, the studies have been confined to a limited framework in the relevant literature, and mostly they have been engaged in developing one-degree cyclic schedules in single robot systems. This research proposes an ant colony optimization based algorithm that simultaneously finds the optimal degree ($k$) of the cyclic schedule and the optimal sequence of robots' moves

corresponding to the *k*-degree cyclic schedule, which eventually minimize the cycle time and thus maximize the throughput rate.

Given the processing requirements, the objective that is most interested by manufacturers is the maximization of cell productivity. A natural and widely used measure of productivity is throughput; the number of finished parts produced per unit of time M. W. Dawande, et al., 2007. In the blocking flow shop robotic cells, a crew of computer-controlled robots performs the movements of parts between the machines. By optimally assigning the robots to these moves and sequencing the transportation operations at the same time, the throughput of the cell is maximized.

## 4.2 Literature Review

Sethi, et al. (1992) performed the purely analytic study of a flow shop robotic cell containing two and three machines and a single robot for one-degree cyclic schedules. Logendran & Sriskandarajah (1996) extended this work to cover different configurations of cells and more general times for the actions of the robot. Crama & Van De Klundert (1997) and M. Dawande, et al. (2002), each developed polynomial-time algorithms to find the optimal one-unit cycles considering machine numbers. More thorough reviews of the cyclic robotic flow shop cells literature can be found in surveys by Hall (2007), Crama, et al. (2000), and M. Dawande, et al. (2005), and in the book by M. W. Dawande, et al. (2007).

Agnetis (2000) studied the two- and three-machine one-degree cyclic robotic cells with a single robot and developed polynomial algorithms to solve the considered problem. Levner, et al. (1997) extended the same problem to *m*-machine using the notion of prohibited intervals. They also considered the usage of more than one robot in large robotic cells in order to eliminate material handling bottlenecks. Karzanov & Livshits (1978) studied a system with parallel tracks where robots run along their respective tracks, and they proposed an algorithm to find the minimal number of robots for a given cycle time. Furthermore, J. Liu & Jiang (2005) proposed an

efficient algorithm for one-degree cyclic scheduling of two robots in a no-wait robotic cell.

Moreover, many researchers have noted and verified that multi-degree cyclic schedules usually have larger throughput than one-degree cyclic schedules. Kats, et al. (1999) studied the *k*-degree cyclic robotic cells with a single robot and proposed a sieve method based algorithm to optimize the cycle time. On the basis of their problem, Che, et al. (2002) proposed an exact algorithm based on branch-and-bound procedure for the no-wait robotic cells. They have noted that the proposed algorithm is polynomial for a fixed *k*-degree ($k > 1$), but becomes exponential if *k* is assumed arbitrary. Lately, Che & Chu (2005) showed that the no-wait multi-degree cyclic robotic cell scheduling problem with two parallel robots can be also solved in polynomial time for a fixed *k*-degree ($k > 1$), but exponential if $k$ is arbitrary. However, the algorithm cannot be mechanically extended to the multi-robot case because it explores some properties specific to the cells with two robots.

Furthermore, Che & Chu (2009) considered the *k*-degree cyclic robotic cell scheduling problem with multiple robots in a no-wait system. They proposed a polynomial algorithm to find the minimum number of robots for all *k*-degree ($k > 1$) cyclic schedules and consequently the optimal cycle time for any given number of robots. On the basis of this study, Che, et al. (2012) extended the proposed method to the cyclic scheduling of a no-wait re-entrant robotic flow shop with multiple robots. Zhou, et al. (2012) proposed a mixed-integer programming model for the fixed *k*-degree cyclic robotic cells with a single robot and considering time window constraints. Therefore, to determine the optimal degree the proposed model should be solved for different values of *k*, separately.

As far as the related research is concerned, there is no solution method in the literature for the multi-degree cyclic flow shop robotic cell scheduling (*k*DCFS-RCS) problem with multiple robots and free processing times. The proposed metaheuristic algorithm also assumes the degree of the cyclic schedule as a variable, and

determines the optimal degree of the cyclic schedule and the optimal sequencing of the robots moves simultaneously such that the throughput rate is maximized.

Table 4.1 Literature of cyclic flow shop robotic cell.

| Num | Research | Robot Number | Degree of Cyclic Schedule | Machine Number | Pickup Criterion | Main contribution |
|---|---|---|---|---|---|---|
| **1** | H. Chen, et al. (1998) | 1 | 1 | M | Time Window | A branch-and-bound algorithm |
| **2** | Che, et al. (2002) | 1 | K | M | No-Wait | A branch-and-bound algorithm |
| **3** | Che & Chu (2005) | 2 | K | M | No-Wait | A polynomial-time algorithm |
| **4** | Neil Geismar, et al. (2005) | 1 | K | M | Free | A polynomial-time algorithm for the three most common classes: additive, constant, and Euclidean travel-time. |
| **5** | Leung & Levner (2006) | R | 1 | M | No-Wait | A polynomial-time algorithm to find the minimum number of robots for all feasible cycle times, and to find the optimal cycle time for any given number of robots. |
| **6** | Alcaide, et al. (2007) | R | K | M | Time Window | Graph model and PERT–CPM solution techniques |
| **7** | Jiang & Liu (2007) | R | 1 | M | No-Wait | A graph model of the production process is developed, making it possible to apply PERT–CPM solution techniques. |
| **8** | Kats & Levner (2009) | 1 | 2 | M | No-Wait | A polynomial-time algorithm |
| **9** | Che & Chu (2009) | R | K | M | No-Wait | A polynomial-time algorithm |
| **10** | Che, et al. (2011) | R | 2 | M | No-Wait | A mathematical model and a polynomial algorithm for the multi-robot 2-cyclic scheduling problem |
| **11** | Zhou, et al. (2012) | 1 | K | M | Time Window | A mixed-integer linear programming model |
| **12** | Che, et al. (2012) | R | K | M | No-Wait | A polynomial-time algorithm |
| **13** | Jiang & Liu (2014) | R | 1 | M | Time Window | A polynomial algorithm for scheduling multiple robots in re-entrant robotic flow shop. |
| **14** | This research | R | K (would be determined) | M | Free | Ant colony optimization based algorithm. |

### 4.3 Problem Definition

The robotic flow shop under study consists of multiple computer-controlled robots for material handling,$M$ machines assigned in a row, and the input and output stations. The machines are indexed as $m \in \{m_{In}, m_1, m_2, \dots, m_M, m_{M+1}\}$, where $m_{In}$ and $m_{M+1}$ are the input and the output stations, respectively. All the parts are of the single type and processed in the robotic flow shop by successively visiting the machines. The part flow can be described as follows: After a part is moved into the robotic flow shop from the input station $m_{In}$, it is processed sequentially through machines $m \in \{m_1, m_2, \dots, m_M\}$, and finally leaves the flow shop from the output station $m_{M+1}$.

In the cyclic flow shop, all the parts are categorized to classes and the parts of the same class have the same conditions in consecutive cycles. In other words, if there are $J$ parts that would be processed in a cycle, the parts are indexed as classes $\{1, 2, \dots, J\}$ according to their sequence of entering the system during the cycle. There may be parts from the same class in each cycle. Thereupon, some of the consecutive processing operations of parts may continue in the upcoming cycles. This means that only one part from the same class is completed per cycle. Therefore, if there are $p$ parts from the same class, the processing operations of this class are divided to $p$ sets of consecutive operations in a cycle.

With the problem description and notation given above, minimizing the mean cycle time needed for per part, requires finding an optimal *k*-degree cyclic schedule. Note that the number of part classes that enter the system during the cycle influences the throughput rate. As the number of part classes in a cycle increases, the total cycle time ($T$) including all the part classes also increases; however, although the total cycle time increases, it is likely to find a solution that minimizes the mean cycle time ($\Phi$) and thus maximizes the throughput rate. Here the objective is to find the optimal *k*-degree cyclic schedule that sequences the transportation operations and assigns the appropriate robots to perform each transportation operation such that the mean cycle time per part is minimized. Figure 4.1 illustrates an example of a robotic cell

containing eight machines (excluding input and output devices) with two single-gripper robots.

Before presenting the proposed mathematical models in detail, the following characteristics about the flow shop robotic cell are stated to make the description of the models easier.

- There are $(M + 2)$ machines including input and output devices assigned in a row.
- The parts are processed by successively visiting the machines.
- All the parts are of identical part type.
- Each machine can process only one part at any time.
- The parts cannot be processed on more than one machine at the same time.
- There is no intermediate buffer between the machines.
- Each part is transported to the next machine any time after completing its processing on a machine.
- Each robot can transport only one part at the same time (single-gripper).
- Each transportation operation is performed by only one robot.
- The required time for the robots to transport the parts between machines and for their empty moves are deterministic constants.

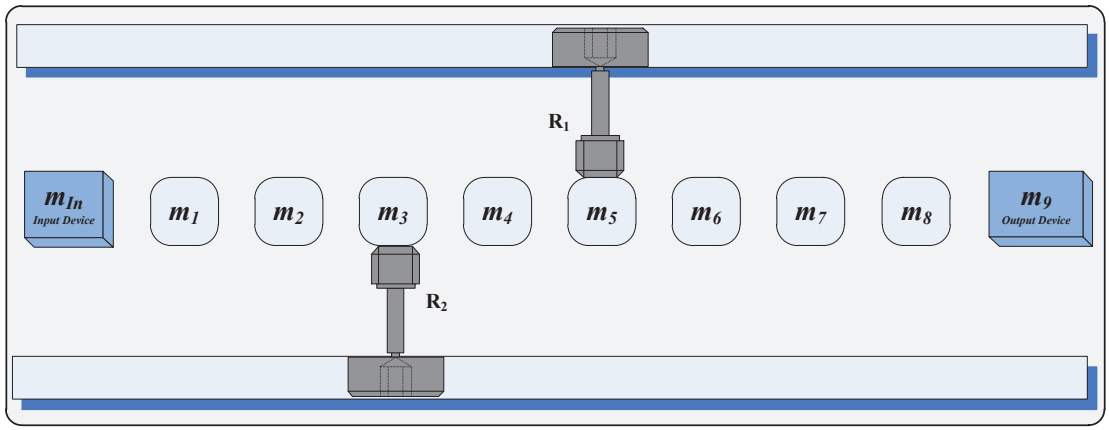



Figure 4.1 An example of the cyclic flow shop robotic cell with multiple single-gripper robots.

For simplicity, the robot move of transporting a part of class $j$ from machine $m$ to machine $m+1$ is called transportation operation $[j,m]$. A transportation operation $[j,m]$ is made up of three simple robot sub-operations: (1) unloading a part of class $j$ from machine $m$; (2) transporting the part of class $j$ to the next machine $m+1$; and (3) loading it onto machine $m+1$. It is understandable that there are totally $k \times (M+1)$ transportation operations in a $k$-cyclic schedule. Under this definition, we define the time required for any robot to perform transportation operation $[j,m]$ as $\theta_m$, and the time required for any robot without transporting parts to travel from machine $m$ to machine $n$ (such a travel is also called a empty move $[\![m,n]\!]$) as $\delta_{[\![m,n]\!]}$.

### 4.4 Ant Colony Optimization (ACO)

ACO algorithms are inspired by the foraging behavior of natural ant colonies in which individual ants deposit a substance called pheromone on the path while moving from the nest to the food sources and vice versa. So, the other ants pursue deposited pheromones to find the paths. Therefore, stronger pheromone in a path increases the probability of selecting that path. Consequently, after some time all the ants would select the shortest path from the nest to the food source. Each ant is a kind of computational agent where an ant colony based approach uses these agents to lead the search mechanism in the solution space for optimality (Dorigo & Gambardella, 1997).

Considering aforementioned concept, each artificial ant starts with an empty position and chooses one of the nodes. Then, the ant iteratively appends an unselected node to the partial tour constructed so far until a complete tour is built. At each step, a node is chosen by applying a transition rule based on the pheromone trails. The performance quality of the constructed tour is then used to modify the pheromone trails to lead the other ants as local updating rule. Once all the ants in the colony have built their tours, to make the search more directed, the pheromone trails are modified by applying a global updating rule. The general structure of the proposed ACO based algorithm is then represented as follows.

### *4.4.1 Initialization of the Pheromone Trails*

The pheromone trails are updated during the search procedure in order to form a kind of adaptive memory of previously found solutions to guide the searching process. The pheromone amount specifies how good the previous edges and nodes are for the following ants. The pheromone trails are updated according to the objective function after a colony of ants has completed their tours and formed a kind of adaptive memory of previously found tours to guide the searching process. In this research, the value of pheromone trails has been initialized as one (1.0) for all the edges.

### *4.4.2 Transition Rule*

At each iteration of the proposed ACO based algorithm, each ant constructs a solution for the problem at hand. While building a complete tour as a solution, the $f^{th}$ ant moves from the start node to the end node. The state transition rule given by the following equation is applied to determine the selected node $b$ for the $a^{th}$ position of the $f^{th}$ ant tour (Dorigo & Gambardella, 1997).

$$b \Leftarrow \begin{cases} arg \underset{u \in Cn(a)}{max} \{\tau(a,u)\} ; & q \leq q^0 \\ B_{Selection} ; & otherwise \end{cases} \tag{4.1}$$

Where, $\tau(a,u)$ is the pheromone trail of edge $(a,u)$, $Cn(a)$ is the set of candidate nodes between the remaining nodes that should be visited by the $f^{th}$ ant. Also, $q$ is a random number uniformly distributed in$[0,1]$, and $q^0$ is a parameter ($0 \leq q^0 \leq 1$), which determines the relative importance of exploitation versus exploration. In addition, $B_{Selection}$ chooses the node $b$ from the set of candidate nodes based on a probabilistic rule. The selection of nodes to produce a tour plays an extremely important role in ACO. The node selections are normally done by a probabilistic way, which is based on the pheromone trail of edges such that the edges with higher pheromone values have an increased chance of selection.

A node in the current candidate nodes set can be selected by applying different methods. Holland (1975) developed the roulette wheel mechanism and there are some other selection mechanisms in the literature of which, scaling techniques, tournaments, elitist models and ranking methods (Goldberg & Holland (1988), Michalewicz (1992)) can be mentioned. In this research, the extended roulette wheel given by Atashpaz-Gargari & Lucas (2007) is used to select the candidate nodes. The extended roulette wheel eliminates the cumulative distribution function and uses the probability density function. Therefore, the mentioned extended mechanism needs much less computational time compared to the roulette wheel mechanism (Atashpaz-Gargari & Lucas, 2007).

Initially, the pheromone trail of each candidate node related edge is normalized based on the total pheromone trail value of all candidate nodes. The normalized pheromone trail of edge$(a,u)$ is calculated by:

$$\tau^{n}_{(a,u)} = \left(\frac{\tau(a,u)}{\sum_{u \in Cn(a)} \tau(a,u)}\right) \tag{4.2}$$

The vector $PH$ is then formed based on the normalized pheromone trail of candidate nodes. A vector $PR$ with the same size as $PH$ is then generated whose elements are uniformly distributed random numbers between $[0,1]$. The selection of nodes is done with respect to vector $PD$ obtained by subtracting $PR$ from $PH$.

$$|Cn(a)| = |PH| = |PR| = |PD| \tag{4.3}$$

$$PH = \left[\tau^{n}_{(a,u^1)}, \tau^{n}_{(a,u^2)}, \dots, \tau^{n}_{(a,u^{|PH|})}\right] \tag{4.4}$$

$$PR = \left[pr^{u^1}, pr^{u^2}, \dots, pr^{u^{|PH|}}\right] \tag{4.5}$$

$$PD = \left[\tau^{n}_{(a,u^1)} - pr^{u^1}, \tau^{n}_{(a,u^2)} - pr^{u^2}, \dots, \tau^{n}_{(a,u^{|PH|})} - pr^{u^{|PH|}}\right] = \left[pd^{u^1}, pd^{u^2}, \dots, pr^{u^{|PH|}}\right] \tag{4.6}$$

The $B_{Selection}$ used in the proposed transition rule chooses the node associated with the highest value in vector $PD$. The derivation is discussed in the following

(based on Solimanpur & Foroughi (2011)), which proves that the $B_{Selection}$ provides more selection chance for the nodes with higher value edge.

Suppose $(a, u^x)$ is an edge with higher objective function value than edge$(a, u^y)$, where $\tau(a, u^y) < \tau(a, u^x)$. Since the proposed selection mechanism chooses the node associated with the highest value in vector $PD$:

$$\begin{cases} P\left(pd^{u^y} \le pd^{u^x}\right) > P\left(pd^{u^y} \ge pd^{u^x}\right) \\ P\left(pd^{u^y} \ge pd^{u^x}\right) = 1 - P\left(pd^{u^y} \le pd^{u^x}\right) \end{cases} \quad \Rightarrow \quad P\left(pd^{u^y} \le pd^{u^x}\right) > \frac{1}{2} \tag{4.7}$$

Therefore, it is mathematically required to prove that:

$$P\left(pd^{u^y} \le pd^{u^x}\right) > \frac{1}{2} \tag{4.8}$$

Since $pr^{u^x}$ and $pr^{u^y}$ are random variables with uniform distribution in $[0,1]$, it is obvious that $pr^{u^x} - pr^{u^y}$ will be a random variable with uniform distribution in $[-1,1]$. Therefore, thedistribution functions of $pr^{u^x} - pr^{u^y}$ can be written as follows.

$$f_{pr^{u^x} - pr^{u^y}}(t) = \frac{1}{2} \;\longrightarrow\; -1 \le t \le 1 \tag{4.9}$$

$$P\left(pr^{u^x} - pr^{u^y} \le t\right) = \begin{cases} 0 & ;\ t < -1 \\ \frac{t+1}{2} & ;\ -1 \le t \le 1 \\ 0 & ;\ 1 < t \end{cases} \tag{4.10}$$

Thus, it is concluded that:

$$P\left(pd^{u^y} \le pd^{u^x}\right) = P\left(\tau(a, u^y) - pr^{u^y} \le \tau(a, u^x) - pr^{u^x}\right) \tag{4.11}$$

$$P\left(pd^{u^y} \le pd^{u^x}\right) = P\left(pr^{u^x} - pr^{u^y} \le \tau(a, u^x) - \tau(a, u^y)\right) \tag{4.12}$$

$$P\left(pd^{u^y} \le pd^{u^x}\right) = \frac{\tau(a,u^x) - \tau(a,u^y) + 1}{2} = \frac{\tau(a,u^x) - \tau(a,u^y)}{2} + \frac{1}{2} \tag{4.13}$$

$$\tau(a, u^x) \ge \tau(a, u^y) \Longrightarrow \frac{\tau(a,u^x) - \tau(a,u^y)}{2} \ge 0 \Longrightarrow P\left(pd^{u^y} \le pd^{u^x}\right) \ge \frac{1}{2} \tag{4.14}$$

Thus, we have:

$$\tau(a,u^y) < \tau(a,u^x) \Longrightarrow P\left(pd^{u^y} \leq pd^{u^x}\right) \geq \frac{1}{2} \tag{4.15}$$

### *4.4.3 Updating of the Pheromone Trails*

Each ant builds a tour by repeatedly applying the state transition rule and due to the special characteristics of the *k*DCFS-RCS problem the local and global updating rules are utilized to update the pheromone trails.

#### *4.4.3.1 Local Updating*

At each iteration, before constructing any tour by the ants the pheromone trails of all edges are saved in $\tau_s(a,b)$ as a short term memory to apply the local evaporation rule, where,

$$\tau_s(a,b) \longleftarrow \tau(a,b)\ ; for\ all\ the\ (a,b)\ edges \tag{4.16}$$

During the construction of a tour by an ant, the amount of pheromone on the visited edges is updated by applying the local updating rule in (4.17) Every time an ant constructs a path, the local updating rule will decrease the amount of pheromone on the visited edges and make them less attractive for the next ants (local evaporation). By this way, the desirability of edges changes dynamically in order to shuffle the tour.

$$\begin{cases} \tau_s(a,b) \longleftarrow (1-\omega) \times \tau_s(a,b) \\ if\ (a,b)\ \in tour\ of\ ant\ v \end{cases} \tag{4.17}$$

Where, $\omega(0 < \omega < 1)$ is the local pheromone evaporating parameter.

*4.4.3.2 Global Updating*

Furthermore, once all ants have terminated their tours, pheromone trails on the edges are modified again by applying the global updating rule. Firstly, all the pheromone trails are restored to the values at the start of each iteration as:
$\tau(a,b)\ ; for\ all\ the\ (a,b)\ edges$

Then the global updating is performed as follows:

$$\begin{cases} \tau(a,b) \longleftarrow (1-\theta) \times \tau(a,b) + \left(\theta \times \Delta\tau(a,b)\right) \\ \Delta\tau(a,b) = \begin{cases} \mu^{v}\ ; if\ (a,b)\ \in tour\ of\ the\ v^{best}\ ant \\ 0\ \ ; otherwise \end{cases} \end{cases} \tag{4.18}$$

Where, the parameter $\theta$ ( $0 < \theta < 1$ ) has been considered as the global pheromone updating parameter of the edges. The deposition value of those edges being the part of the best ant ($v^{best}$) tour from the previous iterations is $\left(\theta \times \Delta\tau(a,b)\right)$.

## 4.5 The Multi Bridge Ant Colony Optimization (MB-ACO)

The considered *k*DCFS-RCS problem contains a number of sub-problems with network structures, which any proposed metaheuristic algorithm should consider solving in parallel to reach an optimum solution. The constructive type of search algorithms like ACO are much more effective if the problem has an inherently network structure. Thus, a multi-bridge ant colony optimization (MB-ACO) approach has been proposed for the *k*DCFS-RCS problem to handle the simultaneous solution of the sub-problems metaheuristically. In this approach, there are bridges connecting the different regions that represent the sets of nodes for each sub-problem. The detailed explanations of the sub-problems and their related nodes sets are presented in the following sections. In addition, the instance problem given in Table 4.2 is used to briefly explain the sub-problems and the nodes sets.

Table 4.2 The related data of the considered instance problem.

| Data | | Machines | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | In | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| **Processing time** | | 0 | 42 | 59 | 47 | 63 | 50 | 43 | 74 | 51 | 0 |
| **Transportation time** | | 15 | 13 | 16 | 17 | 15 | 12 | 15 | 12 | 16 | - |
| | | In | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| | In | 0 | 3 | 6 | 9 | 12 | 15 | 18 | 21 | 24 | 27 |
| | 1 | 3 | 0 | 3 | 6 | 9 | 12 | 15 | 18 | 21 | 24 |
| | 2 | 6 | 3 | 0 | 3 | 6 | 9 | 12 | 15 | 18 | 21 |
| | 3 | 9 | 6 | 3 | 0 | 3 | 6 | 9 | 12 | 15 | 18 |
| **Empty move** | 4 | 12 | 9 | 6 | 3 | 0 | 3 | 6 | 9 | 12 | 15 |
| | 5 | 15 | 12 | 9 | 6 | 3 | 0 | 3 | 6 | 9 | 12 |
| | 6 | 18 | 15 | 12 | 9 | 6 | 3 | 0 | 3 | 6 | 9 |
| | 7 | 21 | 18 | 15 | 12 | 9 | 6 | 3 | 0 | 3 | 6 |
| | 8 | 24 | 21 | 18 | 15 | 12 | 9 | 6 | 3 | 0 | 3 |
| | 9 | 27 | 24 | 21 | 18 | 15 | 12 | 9 | 6 | 3 | 0 |
| ***Number of Robots:** 2* | | | | | | | | | | | |
| ***Maximum number of parts processed during a cycle:** 4* | | | | | | | | | | | |

### *4.5.1 Determining the Optimum k-Degree of the Cycle*

As mentioned previously, the number of parts that enter the system during a cycle (*k-degree*) influences the throughput. Considering $J$ as the maximum number of parts that can be processed during a cycle, our proposed metaheuristic approach determines the optimum *k-degree* for the *k*DCFS-RCS while optimizing the other sub-problems, where $k \in \{1,2,\dots,J\}$.

For the first sub-problem and for the first set of nodes of the proposed MB-ACO, a pheromone trail based approach has been proposed to optimize the $k$-degree value of the cyclic schedule. Considering $(J)$ maximum number of parts that can be processed during the cycle, there are $J$ nodes as paths between the nest and the food for the artificial ants. Figure 4.2 illustrates the graph representation of the first nodes set of the proposed MB-ACO. The pheromone trails of ants' selected paths would be updated due to the obtained cycle time value$(\Phi)$ at each iteration.

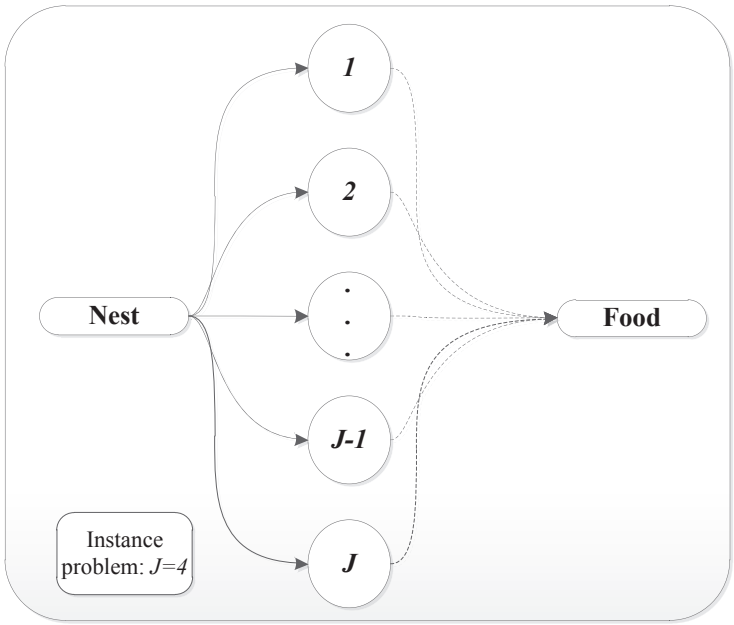



Figure 4.2 The first set of nodes that determines the degree of cycle.

### *4.5.2 Determining the Occupied Machines at the Start of the Cycle*

Due to the special characteristics of the *k*DCFS-RCS problem, some of processing operations will occupy the related machines at the beginning of the cycle. Figure 4.3 indicates the importance of determining the occupied machines and occupying classes, where five machines are occupied in a 4-degree cycle for the example problem given in Table 4.2.

Since the *k*DCFS-RCS problem requires that the precedence constraints between the transportation operations of each class be considered, the solution procedure should determine the number of occupied machines at the start of the cycle. The graph representation of the second nodes set of the proposed MB-ACO together with the first nodes set has been given in Figure 4.4, where both the *k*-degree and the number of occupied machines ($g$) at the start of the cycle are determined.

Moreover, the third nodes set has to be considered, which the ants would path to determine each one of the occupied machine. The graph representation of the third nodes set of the MB-ACO is illustrated in Figure 4.5.

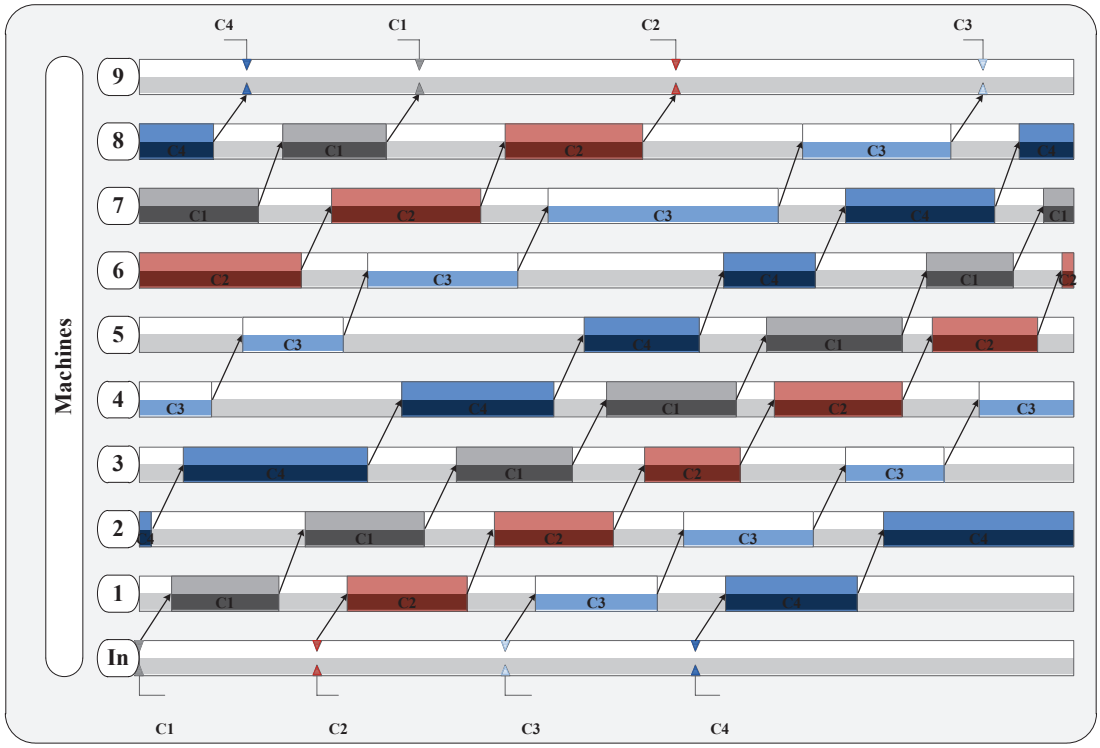



Figure 4.3 The Gantt chart of the instance problem indicating the sequence of all transportation operations.

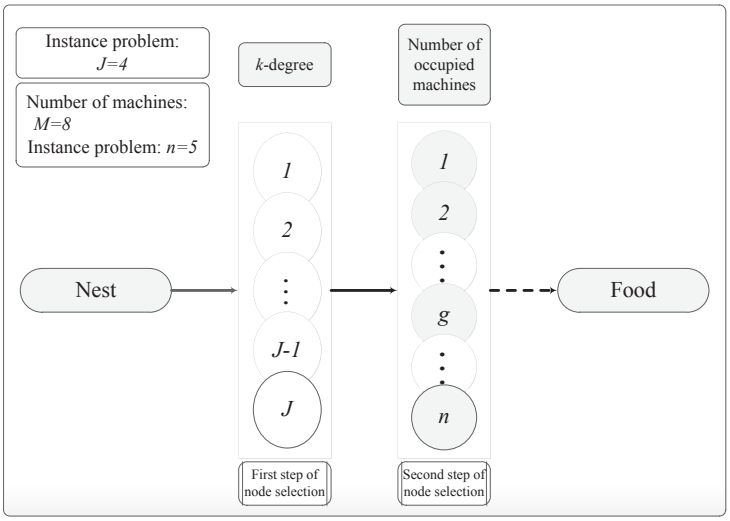



Figure 4.4 The second set of nodes that determines the number of occupied machines at the start time of the cycle.

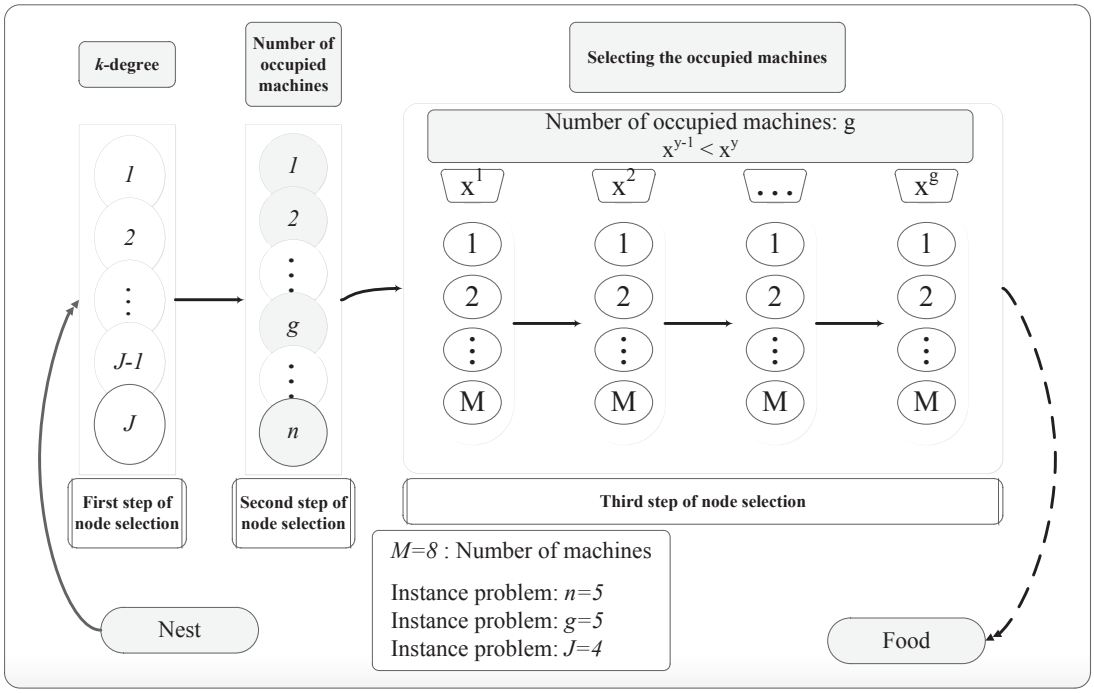



Figure 4.5 The third set of nodes that determines each one of the occupied machines at the start time of the cycle.

Consider a $k$-degree cycle and a set that indicates the occupied machines ($SOM$) at the beginning of the cycle in the increasing order. Due to the cyclic behavior of $k$DCFS-RCS problem, the related $a^{th}$ class that occupies each selected machine ($m^x \in SOM$) is determined as follows:

$$SOM = \{m^1, m^2, \dots, m^{x-1}, m^x, m^{x+1}, \dots, m^g\};\ m^x < m^{x+1} \tag{4.19}$$

$$\forall \begin{cases} m^x \in SOM \\ a = \begin{cases} k - (x\ mod\ k) + 1\ ;\ (x\ mod\ k) \neq 0 \\ 1\ ;\ (x\ mod\ k) = 0 \end{cases} \end{cases} \Rightarrow \left(m^x \propto a_{class}^{th}\right) \tag{4.20}$$

Accordingly, the set of occupied machines for the given instance problem in Table 4.2 is $SOM = \{2,4,6,7,8\}$, where $(|SOM| = g = 5)$ and the part classes that occupy the indicated machines in $SOM$ are $\{4,3,2,1,4\}$, respectively considering $k = 4$.

Now, the transportation operations for the part classes should be identified. Consider that the $a^{th}$ class occupies the $m^{th}$ machine at the beginning of the cycle, then the transportation operations of the $a^{th}$ class are divided to sub-sets as follows:

$$\begin{cases} \{[a,n] | n \in \{In, 1, \dots, M\}, n < m\} \\ \{[a,n] | n \in \{In, 1, \dots, M\}, m \leq n\} \end{cases} \tag{4.21}$$

In addition, if the $a^{th}$ class occupies $(y)$ number of machines at the beginning of the cycle then:

$$\begin{cases} occupied\ by\ the\ a^{th}\ class: \{m^1, m^2, \dots, m^y\} \\ n \in \{In, 1, \dots, M\} \end{cases} \Rightarrow \begin{cases} \{[a,n] |\ n < m^1\} \\ \{[a,n] |\ m^1 \leq n < m^2\} \\ \vdots \\ \{[a,n] |\ m^{y-1} \leq n < m^y\} \\ \{[a,n] |\ m^y \leq n\} \end{cases} \tag{4.22}$$

As a result, the number of sub-sets for the consequent transportation operations of each class can be determined using the above partitioning. By the same token, the part classes 1, 2, 3, and 4 for the instance problem given in Figure 4.3 are divided to 2, 2, 2, and 3 numbers of sub-sets, respectively (Table 4.3).

Table 4.3 The sub-sets for the consequent transportation operations of each class in Figure 4.3.

| Part Classes | Sub-sets | Transportation operations |
|---|---|---|
| 1 | 1 | $\{In, 1, 2, 3, 4, 5, 6\}$ |
| | 2 | $\{7, 8\}$ |
| 2 | 1 | $\{In, 1, 2, 3, 4, 5\}$ |
| | 2 | $\{6, 7, 8\}$ |
| 3 | 1 | $\{In, 1, 2, 3\}$ |
| | 2 | $\{4, 5, 6, 7, 8\}$ |
| 4 | 1 | $\{In, 1\}$ |
| | 2 | $\{2, 3, 4, 5, 6, 7\}$ |
| | 3 | $\{8\}$ |

### *4.5.3 Sequencing the Transportation Operations*

Maximizing the throughput as an objective function in the $k$DCFS-RCS problem requires sequencing the transportation operations. In the $k$-degree cycle there are $\big(k \times (M+1)\big)$ number of transportation operations as $\big\{[a,m]|m \in \{In,1,\dots,M\}, a \in \{1,2,\dots,k\}\big\}$, where $[a,In]$ indicates transporting of the $a^{th}$ class from the input device to the first machine. The fourth nodes set provides the paths to generate the sequence of transportation operations for the artificial ants (Figure 4.6).

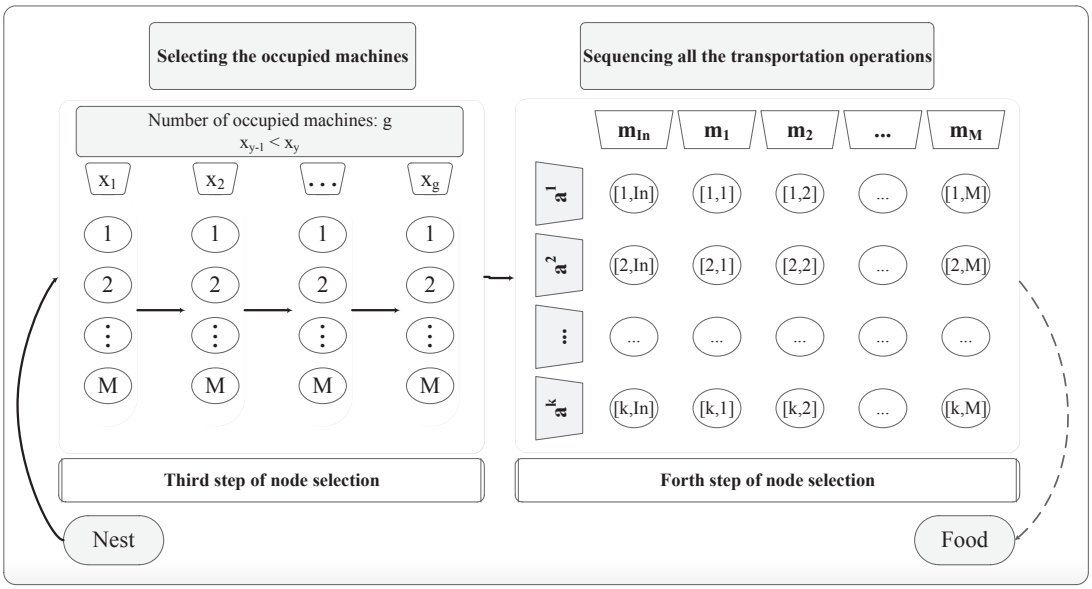



Figure 4.6 Determining the sequence of $Q = k \times (M+1)$ number of transportation operations for the $k$-degree cycle (fourth step).

As an example, Figure 4.7 illustrates the precedence relations for the instance problem in Figure 4.3 considering the sub-sets provided in Table 4.3. In addition, Figure 4.7 provides the precedence relations between the transportation operations when the number of occupied machines at the start time of the cycle is equal to its degree $(g = k)$. Each artificial ant selects the nodes considering the precedence relations from the fourth nodes set to generate a feasible sequence of transportation operations.

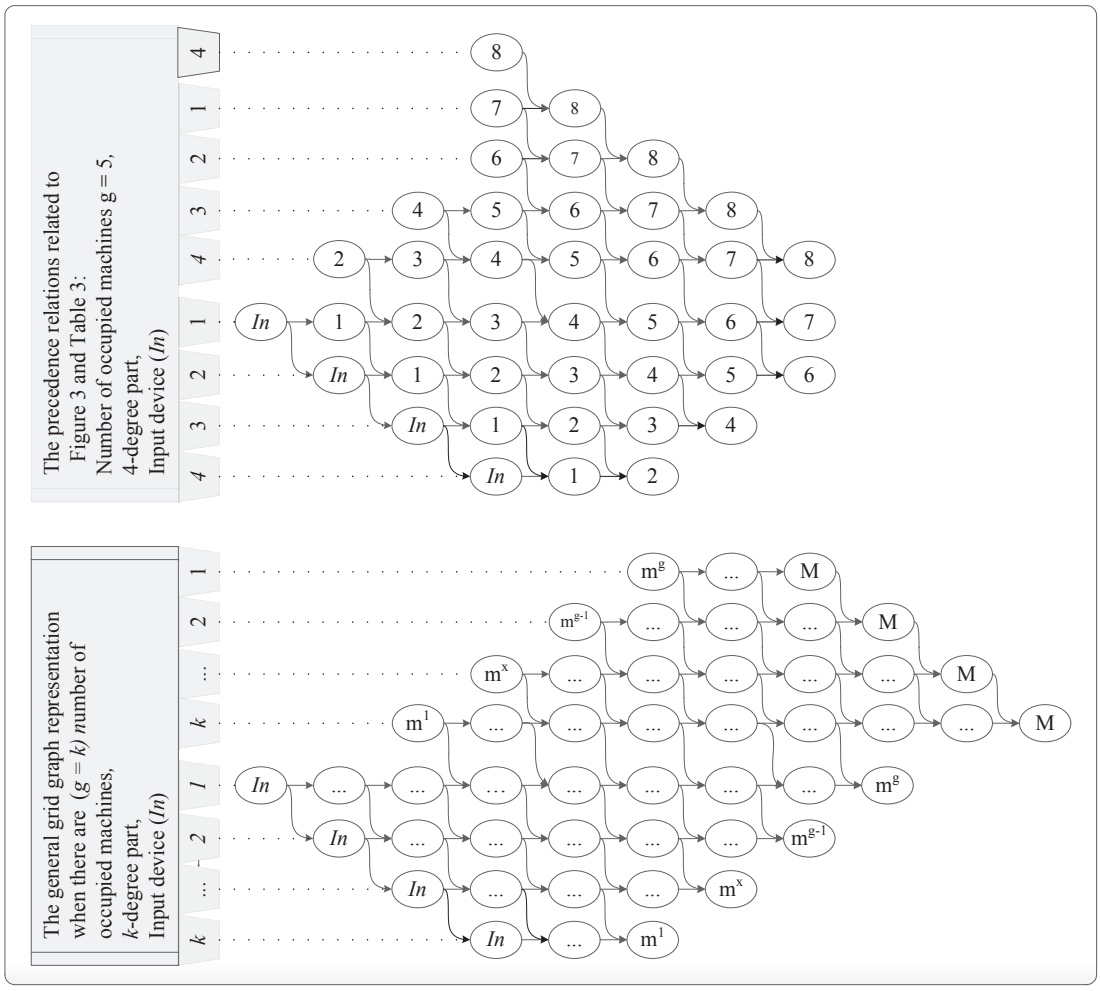



Figure 4.7 The grid graph representation for the precedence relations of transportation operations.

### *4.5.4 Determining the Robot Assignments for the Transportation Operations*

The *k*DCFS-RCS problem considers multiple single gripper robots to perform the transportations between the machines. The limited transportation resources affect the sequence of transportation operations, cycle time and throughput rate. The fifth nodes set of the MB-ACO is used to determine which robot to perform each transportation operation such that the throughput rate is maximized by the appropriate robot assignments. The assigned robot to perform each transportation operation is selected in regard to the pheromone trails of different robots for the considered operation. Figure 4.8 provides the graph representation of the fifth nodes set of the proposed MB-ACO.

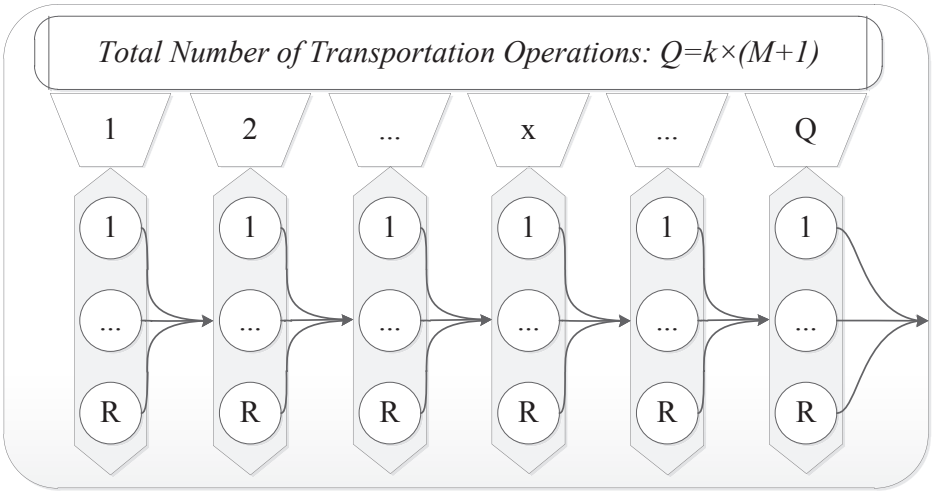



Figure 4.8 Determining the assigned robot to perform the $x^{th}$ transportation operation.

The assigned robots for all the $Q = k \times (M + 1) = 4 \times (8 + 1)$ number of transportation operations in the considered instance problem given in Table 4.2 for the generated feasible solution is provided in Table 4.3. In addition, Figure 4.9 illustrates the Gantt chart of all transportation operations and their assigned robots.

Table 4.4 The assigned robot to perform each transportation operation of instance problem.

| **Degree 1** | **Robots** | | **Degree 2** | **Robots** | | **Degree 3** | **Robots** | | **Degree 4** | **Robots** | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | **1** | **2** | | **1** | **2** | | **1** | **2** | | **1** | **2** |
| [1,*In*] | | ✓ | [2,*In*] | | ✓ | [3,*In*] | ✓ | | [4,*In*] | ✓ | |
| [1,1] | | ✓ | [2,1] | ✓ | | [3,1] | ✓ | | [4,1] | ✓ | |
| [1,2] | | ✓ | [2,2] | ✓ | | [3,2] | ✓ | | [4,2] | ✓ | |
| [1,3] | | ✓ | [2,3] | ✓ | | [3,3] | ✓ | | [4,3] | | ✓ |
| [1,4] | | ✓ | [2,4] | | ✓ | [3,4] | | ✓ | [4,4] | ✓ | |
| [1,5] | ✓ | | [2,5] | | ✓ | [3,5] | ✓ | | [4,5] | | ✓ |
| [1,6] | ✓ | | [2,6] | ✓ | | [3,6] | | ✓ | [4,6] | | ✓ |
| [1,7] | ✓ | | [2,7] | | ✓ | [3,7] | | ✓ | [4,7] | | ✓ |
| [1,8] | ✓ | | [2,8] | | ✓ | [3,8] | | ✓ | [4,8] | ✓ | |

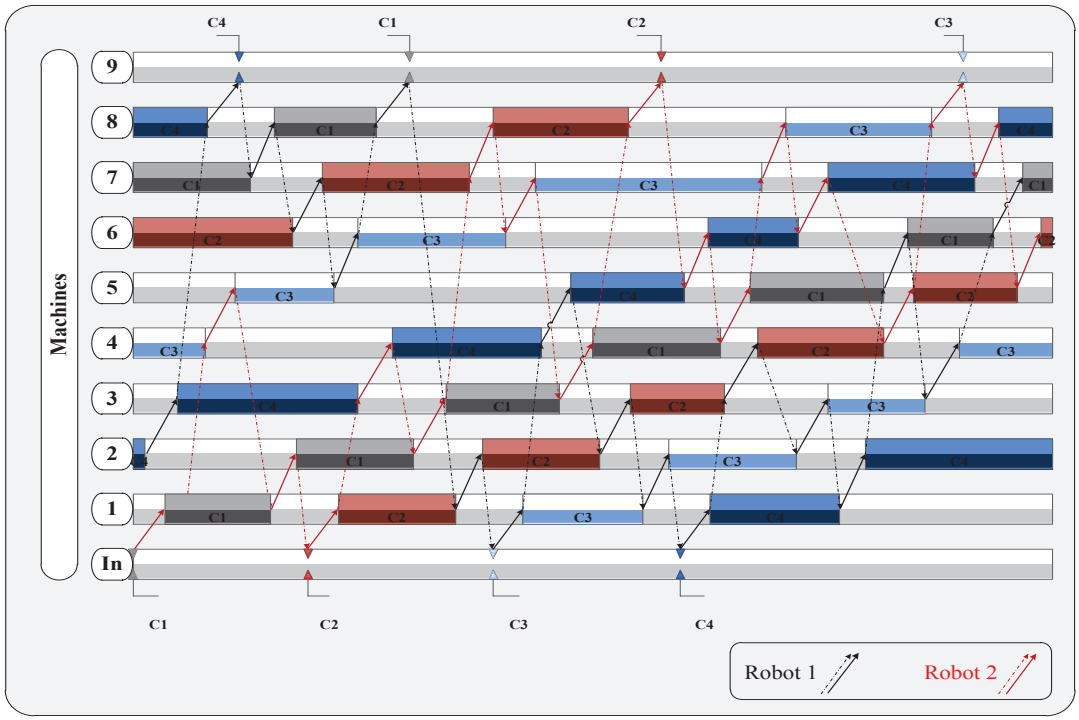



Figure 4.9 The Gantt chart of the instance problem indicating the sequence of all transportation operations and their assigned robots.

The general structure of the proposed MB-ACO for the *k*DCFS-RCS problem and the different regions representing the sub-problems are illustrated in Figure 4.10. The artificial ants have to generate a path at each different region before crossing the bridge to the next region. In addition, Figure 4.11 provides the flow chart of the proposed MB-ACO algorithm for the *k*DCFS-RCS problem.

Figure 4.10 The general structure of the proposed MB-ACO algorithm and the regions that the artificial ants have to visit.

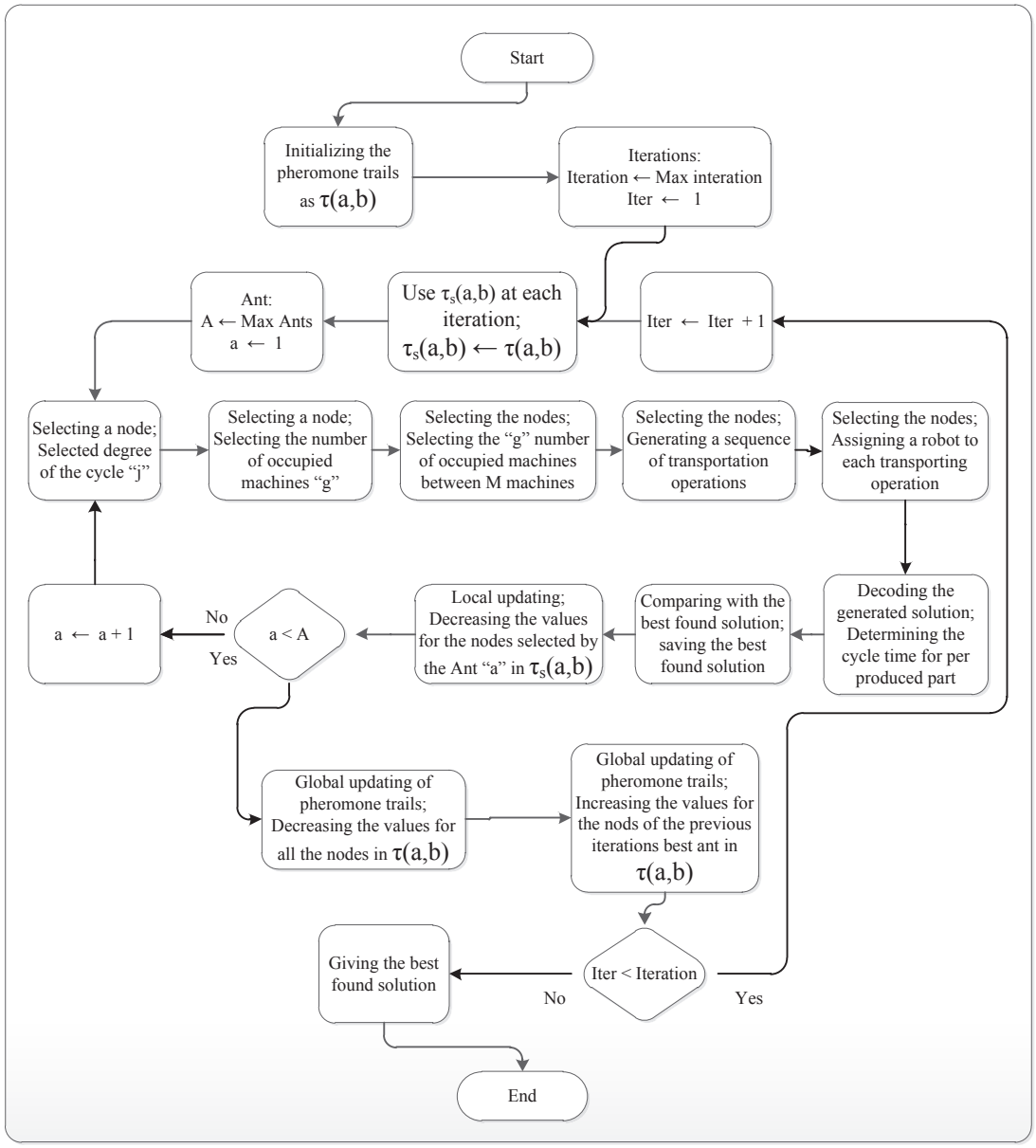



Figure 4.11 The flow chart of the proposed MB-ACO algorithm for the considered kDCFS-RCS problem.

### *4.5.5 Encoding Algorithm*

A feasible solution for the *k*DCFS-RCS problem is generated considering the selected *k*-degree, number of occupied machines and the machines, which are occupied at the start of the cycle. The feasible solution is determined as the permutation of all $k \times (M+1)$ transportation operations. This kind of representation uses a permutation in which unload operations are sequenced to specify in which

order they are performed. The length of the representation is the total number of unload operations performed by the robots. It is assumed that each unload operation will also include transportation to the next related machine and loading on it. It should be noted that the permutation of transportation operations is generated considering the precedence relations which were discussed previously. Thereupon, the operations sequence (illustrated by $\delta(f)$ in Figure 4.12) that will be unloaded by the robots is generated in the fourth region of the proposed MB-ACO algorithm. Moreover, the assigned robot to perform each transportation operation is determined during the fifth region of the algorithm and saved in the matrix $RoTO_{[j,m]}$ for each $[j,m]$ (Figure 4.13).

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\delta(f)^{class}$ | *C1* | *C4* | *C3* | *C4* | *C1* | *C1* | *C2* | *C3* | ... | ... | ... | ... |
| $\delta(f)^{machine}$ | *In* | *2* | *4* | *8* | *7* | *1* | *6* | *5* | ... | ... | ... | ... |
| *f* | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | ... | ... | ... | $k \times (M+1)$ |

The sequence of transportation operations indicated in Figure 9.

Figure 4.12 Encoding the feasible solution; the sequence of transportation operations.

| $[j,m]$ | | Part class *j* | | | | |
|---|---|---|---|---|---|---|
| | | **1** | **2** | … | … | ***J*** |
| Unloading from machine *m* | ***In*** | $RoTO_{[1,In]}$ | $RoTO_{[2,In]}$ | … | … | $RoTO_{[J,In]}$ |
| | **1** | $RoTO_{[1,1]}$ | $RoTO_{[2,1]}$ | … | … | $RoTO_{[J,1]}$ |
| | **2** | $RoTO_{[1,2]}$ | $RoTO_{[2,2]}$ | … | … | $RoTO_{[J,2]}$ |
| | … | … | … | … | … | … |
| | … | … | … | … | … | … |
| | ***M*** | $RoTO_{[1,M]}$ | $RoTO_{[2,M]}$ | … | … | $RoTO_{[J,M]}$ |

Figure 4.13 Encoding the feasible solution; the assigned robot to perform each transportation operation $[j,m]$.

### *4.5.6 Decoding Algorithm*

Each artificial ant generates a feasible solution for the *k*DCFS-RCS problem. The sequence of transportation operations is generated according to the previously determined *k*-degree and occupied machines. The sequence of transportation operations and the assigned robots to perform each one are needed to obtain the cycle time and to decode the generated feasible solution. The specifications of the encoded feasible solution and the pseudo code of the decoding algorithm are provided in the following.

*Indices and sets:*

$J$ maximum number of identical parts that can enter during a cycle, where $j, k \in \{1,2, \dots, J\}; j \leq k$.

$f$ the index for the transportation operations, $f \in \{1,2, \dots, k \times (M+1)\}$.

$d$ the index for the occupied machines, where $d \in \{1,2, \dots, g\}$ and $g$ is the determined number of occupied machines at the start time of the cycle.

$r$ the index for the single gripper robots, where $r \in \{1,2, \dots, R\}$.

*Parameters:*

$P_m$ standard processing time on machine $m$, where the processing times of all parts at input and output stages are equal to zero.

$\theta_m$ standard handling time of a part from machine $m$ to the next machine $(m+1)$.

$\delta_{[\![m,n]\!]}$ the time required for any robot to perform void move $[\![m,n]\!]$.

$LgNum$ sufficiently large number.

*Specifications of the feasible solution:*

$k$ the number of part classes considered for the feasible solution.

$Q$ the total number of transportation operations.

$\delta(f)$ the sequence of transportation operations $[j,m]$ that saves the related class $j$ and machine $m$.

$Occ_Mch_d$ the machine that corresponds to the $d^{th}$ occupied machines.

$Cl_OccMch_d$ the part class that is occupied by machine $Occ_Mch_d$.

$RoTO_{[j,m]}$ the matrix that provides the assigned robot for each transportation operation.

*Determined by decoding algorithm:*

$StartTO_{[j,m]}$ the start time of each transportation operation.

$T$ the total cycle time .

$\Phi$ the cycle time for per produced part.

$\mu$ throughput rate.

*Decoding algorithm*

**Begin**

Initializing

$f \leftarrow Q \leftarrow k \times (M+1)$

$f \leftarrow Q + 1$

**Repeat**

$f \leftarrow f - 1$

$j \leftarrow \delta(f)^{class}$

$m \leftarrow \delta(f)^{machine}$

**Repeat** *// determining the precedence relations of transportation operations //*

$f' \leftarrow f + 1$

$j' \leftarrow \delta(f')^{class}$

$m' \leftarrow \delta(f')^{machine}$

$temp_1, temp_2, temp_3 \leftarrow 0$

**If** $(j' = j)$ **&&** $(m' = m+1)$ **then:**

$temp_1 \leftarrow StartTO_{[j,m+1]} - P_{m+1} - \theta_m$ *// precedence relations of two successive transportation operations of class j//*

**If** $(m' = m-1)$ **then:**

$temp_2 \leftarrow StartTO_{[j',m']} + \theta_{m'}$ *// precedence relations of transportation operations related to the same machine m//*

**If** $\left(RoTO_{[j',m']} = RoTO_{[j,m]}\right)$ **then:**

$temp_3 \leftarrow StartTO_{[j',m']} - \delta_{[\![m+1,m']\!]} - \theta_m$

*// precedence relations of transportation operations related to the same robot r//*

**Until** $\left(f^{'} = Q\right)$

$StartTO_{[j,m]} \leftarrow \boldsymbol{min}\{LgNum,\ temp_1,\ temp_2,\ temp_3\}$ *// determining the start time of transportations due to the precedence relations //*

**Until** $(f = 1)$

$TEMP \leftarrow \boldsymbol{min}\{StartTO_{[j,m]},\ \boldsymbol{forall}[j,m]\}$

$StartTO_{[j,m]} \leftarrow \left(StartTO_{[j,m]} - TEMP\right);\ \boldsymbol{forall}[j,m]$

$d \leftarrow 0,$

**Repeat**

$d \leftarrow d + 1$

$m \leftarrow Occ_Mch_d$

$j \leftarrow Cl_OccMch_d$

$temp_1 \leftarrow \boldsymbol{max}\{StartTO_{[j,m-1]} + \theta_{m-1} + P_m - StartTO_{[j,m]}\}$

**Until**$(d = |Occ_Mch|)$

$temp_2 \leftarrow \left(StartTO_{[j,m]} + \theta_m\right);\ \boldsymbol{forall}[j,m]$

$T \leftarrow \boldsymbol{max}\{temp_1,\quad temp_1\}$

$\Phi \leftarrow (T/k)$

**End**

### 4.6 Computational Analysis

The computational experiments have been carried out on a set of randomly generated instances. The problem sizes are determined by varying the levels of two parameters, which are respectively the number of processing machines and the number of single gripper robots. There are four groups of problems with respect to the machine numbers in the robotic cell, which are 8, 12, 16, and 20, respectively. The maximum *k*-degree varies according to the machine number for each problem.

In order to assess the empirical performance of the proposed MB-ACO algorithm, the following data sets are generated from the different domains of the uniform distribution $[a, b]$ for each problem, separately.

- The standard processing times of the machines.
  The uniform distributions of $U[150, 250]$, $U[150, 300]$, $U[150, 300]$ and $U[150, 400]$ are used for the problems with machine numbers 8, 12, 16, and 20, respectively.
- The time needed for the loaded robot moves between the successive machines, which are considered as transportation operations.
  The uniform distributions of $U[20, 40]$, $U[40, 70]$, $U[40, 80]$ and $U[40, 100]$ are used for the problems with machine numbers 8, 12, 16, and 20, respectively.
- The time needed for the unloaded robot moves between the machines, which are considered as empty moves.
  The uniform distributions of $U[5, 20]$, $U[10, 20]$, $U[10, 20]$ and $U[10, 30]$ are used for the problems with machine numbers 8, 12, 16, and 20, respectively.

Moreover, the number of single gripper robots is considered at four levels; 4, 5, 8, and 10 for the problems with machine numbers 8, 12, 16, and 20, respectively. Table 4.4 provides the problems and the characteristics of obtained solutions for each problem (a total of 27 problems).

Table 4.5 The generated problem sizes and the proposed MB-ACO algorithm's obtained solutions.

| **Problems** | | | **MB-ACO algorithm** | | | | | **No** |
|---|---|---|---|---|---|---|---|---|
| ***M*** *(#machines)* | ***Max k-degree*** | ***R*** *(#robots)* | ***T*** *(sec)* | ***Φ*** *(sec)* | ***Determined k-degree*** | ***μ*** *(part/min)* | ***Run Time*** *(sec)* | |
| 8 | 3 | 1 | 512 | 512.00 | 1 | 0.117 | 52.45 | 1 |
| | | 2 | 437 | 437.00 | 1 | 0.137 | 60.94 | 2 |
| | | 3 | 838 | 419.00 | 2 | 0.143 | 64.37 | 3 |
| | | 4 | 1245 | 415.00 | 3 | 0.144 | 64.81 | 4 |
| 12 | 4 | 1 | 1049 | 1049.00 | 1 | 0.057 | 79.43 | 5 |
| | | 2 | 872 | 872.00 | 1 | 0.068 | 98.61 | 6 |
| | | 3 | 1606 | 803.00 | 2 | 0.074 | 105.11 | 7 |
| | | 4 | 2354 | 784.66 | 3 | 0.076 | 107.87 | 8 |
| | | 5 | 1569 | 785.50 | 2 | 0.076 | 118.52 | 9 |
| 16 | 5 | 1 | 1568 | 1568.00 | 1 | 0.038 | 97.15 | 10 |
| | | 2 | 1163 | 1163.00 | 1 | 0.051 | 116.09 | 11 |
| | | 3 | 1079 | 1079.00 | 1 | 0.055 | 138.30 | 12 |
| | | 4 | 1075 | 1075.00 | 1 | 0.055 | 152.07 | 13 |
| | | 5 | 2150 | 1075.00 | 2 | 0.055 | 154.22 | 14 |
| | | 6 | 2150 | 1075.00 | 2 | 0.055 | 166.09 | 15 |
| | | 7 | 2150 | 1075.00 | 2 | 0.055 | 183.51 | 16 |
| | | 8 | 4237 | 1059.25 | 4 | 0.056 | 205.74 | 17 |
| 20 | 6 | 1 | 1843 | 1843.00 | 1 | 0.032 | 219.38 | 18 |
| | | 2 | 1595 | 1595.00 | 1 | 0.037 | 241.25 | 19 |
| | | 3 | 1284 | 1284.00 | 1 | 0.046 | 271.53 | 20 |
| | | 4 | 1197 | 1197.00 | 1 | 0.050 | 291.47 | 21 |
| | | 5 | 1197 | 1197.00 | 1 | 0.050 | 311.25 | 22 |
| | | 6 | 2367 | 1183.50 | 2 | 0.050 | 314.77 | 23 |
| | | 7 | 2367 | 1183.50 | 2 | 0.050 | 337.45 | 24 |
| | | 8 | 3474 | 1158.33 | 3 | 0.051 | 336.92 | 25 |
| | | 9 | 4634 | 1158.50 | 4 | 0.051 | 351.08 | 26 |
| | | 10 | 3475 | 1158.33 | 3 | 0.051 | 372.26 | 27 |

Furthermore, Figure 4.13 illustrates the impact of the number of robots on the throughput rate for each problem group. Accordingly, the influence of the number of robots on the throughput rate decreases as the robot number increases. In addition, Figure 4.14 represents the obtained throughput rate and determined *k*-degree separately for each machine group with different number of single gripper robots.

Accordingly, the increase in the *k*-degree is observed at larger number of robots usage for cells with larger number of machines. For example, the appropriate *k*-degree increases from 1-degree to 2-degree as the number of robots is increased from 2, 2, 4, and 5 robots use in the 8, 12, 16, and 20 machine-contained cells, respectively. Therefore, it can be said that the appropriate *k*-degree change involves a larger number of robots support when the number of machines in the robotic cells increases. Moreover, studying the larger *k*-degree schedules to find the appropriate degree needs more robots. Therefore, investigating the larger *k*-degree schedules considering less number of robots is inefficient.

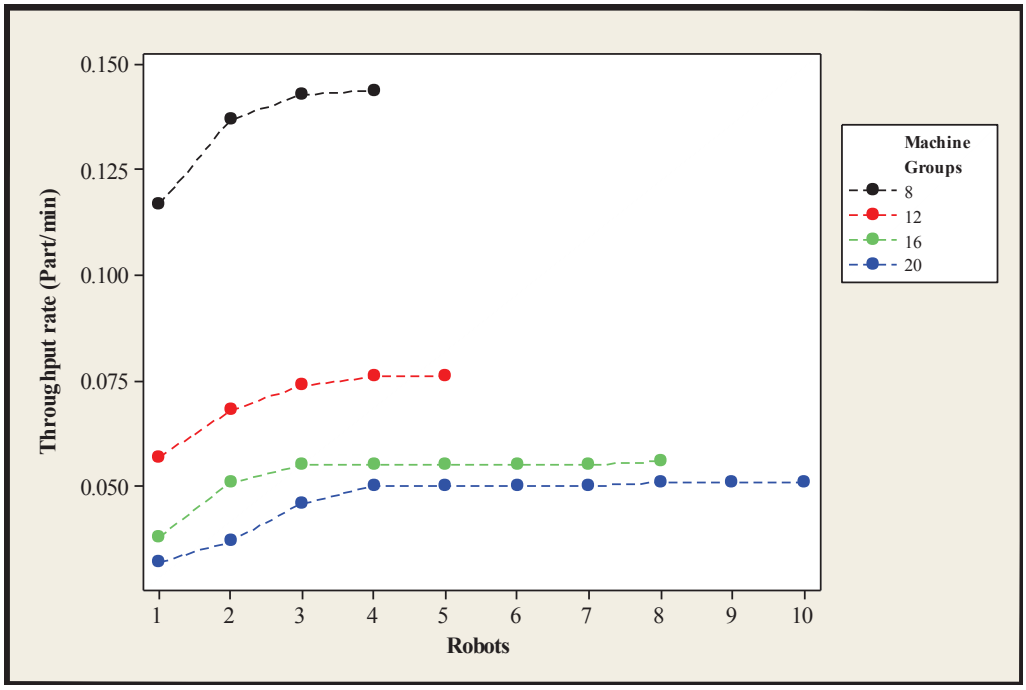



Figure 4.14 The impact of the number of robots on the throughput rate for the *k*DCFS-RCS problem.

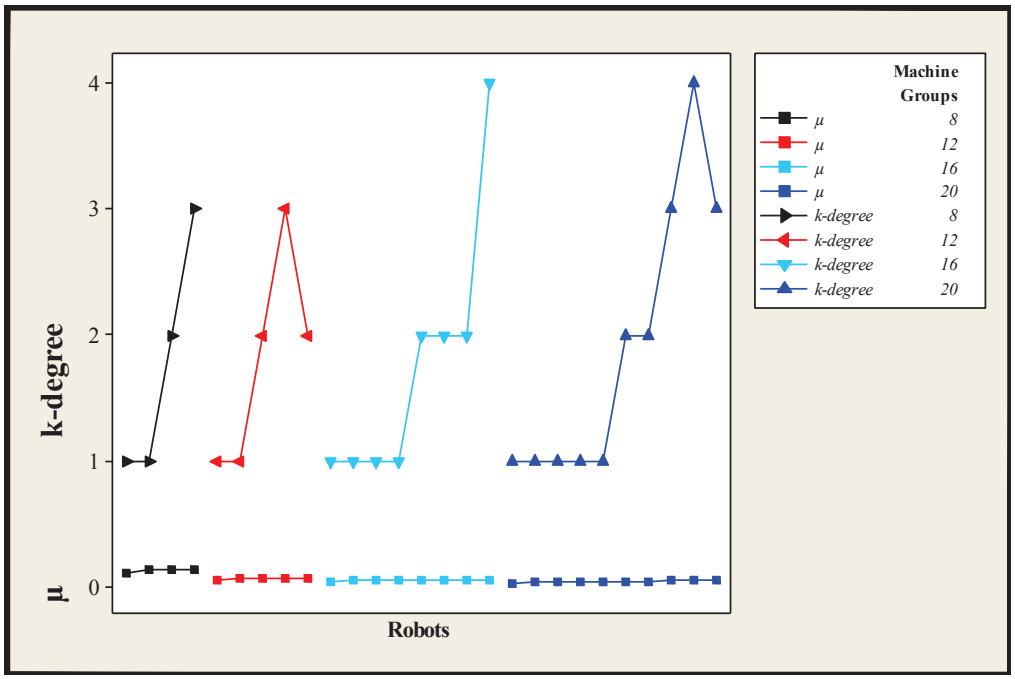



Figure 4.15 The obtained throughput rate (μ) and determined *k*-degree for the generated problems with different single gripper robot numbers.

## 4.7 Conclusion

This research studies the *k*-degree cyclic flow shop robotic cell scheduling (*k*DCFS-RCS) problem considering multiple single gripper robots. Since the investigated problem is hard to solve with exact methods for real-size problems, an ant colony optimization based algorithm (MB-ACO) is proposed for its solution in order to minimize the cycle time for per produced part, and hence to maximize the throughput rate.

To evaluate the performance of the proposed algorithm, four problem categories are considered to be analyzed under different numbers of robots usage (27 problems in total), and the generated problems are solved by the proposed MB-ACO algorithm. Besides, the special characteristics of *k*DCFS-RCS problem are stated by an instance problem given. The computational results confirm the efficiency of different *k*-degrees on cycle time for per produced part and on throughput rate. According to the conducted computational study, the proposed MB-ACO algorithm has solved the investigated *k*DCFS-RCS problem efficiently and in reasonable

computation time for realistic problem sizes, and this feature makes it applicable to real-life problems.

# CHAPTER FIVE
# SOLVING THE CYCLIC JOB SHOP ROBOTIC CELL SCHEDULING PROBLEM USING ANT COLONY OPTIMIZATION

## 5.1 Introduction

The classical job shop problem is known as a standard problem in scheduling and has been widely investigated over the last few decades. There are many situations where it happens that a job is not going to be produced only once but several times. That means, many instances of the same job have to be processed. In large scale production, for instance, there usually is a fixed set of jobs that have to be processed indefinitely often. If we model this as a classical job shop problem we have to include many repetitions of the same job in our schedule. This could be a huge number of individual jobs and, therefore, very hard to solve. Thus, the mentioned sets of jobs should be scheduled in the cyclic manner.

Here we study the cyclic job shop problem with multiple single gripper robots. Thereupon, at the time a job has finished its processing on a machine, it moves automatically directly to the related machine of the next operation if it is free. In many industrial production lines, the multiple transporting robots do these moves. This means that after a job has finished its processing on a machine, a robot unloads the job, transports it to the next machine and loads the job onto that machine. After a job has been transported to, and loaded onto the machine, the robot stays at this machine either, until the job is finished and unloads it, or moves empty to another machine. A cyclic production environment where the robots are in charge of transporting the jobs between the machines are often called robotic cells in the literature. Robots have increased the production capabilities in manufacturing by making the assembly or machinery processing faster, more efficient and reliable than ever before. They also save workers from doing monotonous work on an assembly line. In cyclic scheduling problems, the height is one of the most important parameters. It is defined as the number of cycles that a specific instance of a job will stay in the system from start to finish. Furthermore, it distinguishes such problems

from non-cyclic problems when the height is equal to one for all jobs. However, the literature does not provide many interpretations of the height for the cyclic job shop systems. On contrary to the past study in cyclic job shop scheduling, this study assumes the height of the jobs as a variable, and determines the appropriate height of each job to maximize the throughput rate.

## 5.2 Literature Review

The area we are studying in this work can be characterized as production scheduling where a set of products has to pass several machines in specific orders and where the corresponding production plan has to be repeated over and over again as it is the case in large scale productions.

The relevant cyclic job shop scheduling literature can be given as follows. Song and Lee Song & Lee (1998) investigated the scheduling problem for general cyclic job shops with blocking and developed Petri-net models for the shops. Cavory, Dupas & Goncalves (2005) considered the cyclic job shop scheduling problem with linear constraints. They presented an approach based on the coupling of a genetic algorithm and a Petri-net modeling for the linear precedence constraints. Brucker & Kampmeyer (2008a) introduced the cyclic job shop problem with job repetition model. For the general cyclic job shop problem with blocking, Brucker & Kampmeyer (2005) developed a tabu search algorithm and presented computational results. Jamili, Shafia & Tavakkoli-Moghaddam (2011) considered the cyclic job shop scheduling problem and proposed a hybrid algorithm based on particle swarm optimization and simulated annealing algorithms. Smutnicki (2012) investigated the cyclic job shop scheduling problem and proposed a new lower bound. Jalilvand-Nejad & Fattahi (2013) studied the flexible job shop scheduling problem with cyclic jobs and presented a mixed-integer linear programming model. Also, they developed two solution approaches based on genetic and simulated annealing algorithms to solve real size problems. Kechadi, Low & Goncalves (2013) studied the cyclic job shop problem and they developed an efficient neural network approach to minimize the cycle time of the schedule.

The general cyclic job shop problem with transportation is rarely discussed in the literature. Hanen (1994) studied the cyclic job shop scheduling problem with disjunctive resources. The problem proved to be *NP*-hard and a mixed linear program has been presented. Hurink & Knust (2002) considered a single-machine scheduling problem as a sub-problem in a job shop environment and presented a local search algorithm. In addition, they have generalized the developed local search algorithm to the generalized job shop scheduling problem with a single transport robot Hurink & Knust (2005). Brucker, Burke & Groenemeyer (2012b) presented an integer programming formulation for the cyclic job shop problems with a single transportation device and blocking. They have compared the proposed model with the formulated model based on the work of Hanen (1994), and Brucker & Kampmeyer (2008b). Furthermore, they have developed a branch and bound algorithm to solve the mentioned problem Brucker, Burke & Groenemeyer (2012a). Zhang, Manier & Manier (2012) proposed a mathematical model for flexible job shop scheduling problem with transportation constraints and bounded processing times. In addition, a genetic algorithm with tabu search procedure has been proposed to solve the assignment of resources and sequencing problems on resources.

As far as the related research is concerned, there is no solution method in the literature for the multi-cyclic job shop robotic cell scheduling (CJS-RCS) problem with multiple robots. An ant colony optimization-based metaheuristic approach is proposed. The proposed metaheuristic approach assumes the height of the jobs as a variable, and determines the appropriate height of each job to maximize the throughput rate. Simultaneously, it assigns the proper robot to perform each transportation operation and sequences the robots moves such that the throughput rate is maximized.

### 5.3 Problem Definition

The problem studied in this chapter is a cyclic job-shop problem with multiple single gripper computer-controlled robots for material handling and $M$ machines. These machines are indexed as $m \in \{m_1, m_2, \dots, m_M\}$ including the loading ($m_1$) and

unloading ($m_M$) devices. Each job $j$ is indicated by $J_j$,, and there are a set of $N$ jobs, where $j \in \{J_1, J_2, \dots, J_N\}$ consists of a set of $n_j$ operations that have to be processed in a prescribed order (considering precedence constraints). The mentioned operations of job $J_j$ are indicated by $O_{i,j}$, in an increasing order of $i \in \{1,2, \dots, n_j\}$., where $O_{i,j}$ provides the machine $m$ that processes the $i^{th}$ operation of job $J_j$. All jobs visit the loading and unloading devices at the start and end of their process, respectively. Therefore, the first and last operation of all jobs are considered to be done on the devices $m_1$ and $m_M$. The job flow can be described as follows: After moving the job $J_j$ into the robotic job shop from the loading station $m_1$ (identified with the operation $i = 1$), it is processed sequentially through operations $i \in \{2,3, \dots, n_j - 1\}$, and finally leaves the job shop from the unloading station $m_M$ (identified as the operation $i = n_j$). A single gripper robot $r \in \{1,2, \dots, R\}$ would performs each one of the transportation operations and the robots can perform just one operation at the same time.

### 5.4 The Multi Bridge Ant Colony Optimization (MB-ACO)

The studied CJS-RCS problem consists of such sub-problems with network structures that any proposed metaheuristic algorithm should consider them in parallel. The constructive type of search algorithms like ACO are much more effective if the problem has an inherently network structure. For this reason, a multi-bridge ant colony optimization (MB-ACO) proposed in chapter four (Section 4.4) has been developed to solve the CJS-RCS problem where each bridge connects the different regions representing the sub-problems as different sets of nodes. The detailed explanations of the sub-problems and their related nodes sets are presented in the following sections. In addition, the following instance problem (Table 5.1, Table 5.2, and Table 5.3) is used to briefly explain the sub-problems and the nodes sets.

Table 5.1 The jobs operations related data (processing time and processing machine).

| **Operation** $(i)$ | **1** | | **2** | | **3** | | **4** | | **5** | | **6** | | **7** | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Job** | ***P*** | ***m*** | ***P*** | ***m*** | ***P*** | ***m*** | ***P*** | ***m*** | ***P*** | ***m*** | ***P*** | ***m*** | ***P*** | ***m*** |
| **1** | 0 | 1 | 27 | 6 | 29 | 5 | 45 | 4 | 51 | 3 | 77 | 2 | 0 | 7 |
| **2** | 0 | 1 | 31 | 6 | 50 | 5 | 18 | 2 | 62 | 3 | 45 | 4 | 0 | 7 |
| **3** | 0 | 1 | 39 | 3 | 40 | 2 | 61 | 4 | 39 | 6 | 36 | 5 | 0 | 7 |
| **4** | 0 | 1 | 41 | 4 | 40 | 2 | 84 | 6 | 48 | 5 | 69 | 3 | 0 | 7 |

***P***: Processing time of the $i^{th}$ operation of $j^{th}$ job,
***m***: The machine that processes the $i^{th}$ operation of $j^{th}$ job.

Table 5.2 The needed transportation time after the completion of each processing operation.

| $\boldsymbol{\theta_{j,i}}$ | **Operation** $(i)$ | | | | | |
|---|---|---|---|---|---|---|
| **Job** $(j)$ | **1** | **2** | **3** | **4** | **5** | **6** |
| **1** | 7 | 10 | 11 | 9 | 9 | 13 |
| **2** | 16 | 7 | 24 | 9 | 7 | 6 |
| **3** | 10 | 11 | 14 | 12 | 6 | 12 |
| **4** | 7 | 11 | 15 | 11 | 13 | 10 |

$\theta_{j,i}$: the handling time of the $j^{th}$ job between the $i^{th}$ and $i+1^{th}$ operations.

Table 5.3 The empty move times of robots between the machines.

| $\boldsymbol{\delta_{m,m'}}$ | **Machine** $(m)$ | | | | | | |
|---|---|---|---|---|---|---|---|
| **Machine** $(m')$ | **1** | **2** | **3** | **4** | **5** | **6** | **7** |
| **1** | - | 6 | 7 | 5 | 7 | 5 | 7 |
| **2** | 6 | - | 5 | 6 | 8 | 7 | 7 |
| **3** | 7 | 5 | - | 6 | 5 | 8 | 5 |
| **4** | 4 | 6 | 6 | - | 7 | 6 | 5 |
| **5** | 7 | 8 | 5 | 7 | - | 4 | 6 |
| **6** | 5 | 7 | 8 | 6 | 4 | - | 3 |
| **7** | 6 | 7 | 5 | 5 | 6 | 3 | - |

$\delta_{m,m'}$: the handling time of the $j^{th}$ job between the $i^{th}$ and $i+1^{th}$ operations.

### *5.4.1 Determining the Occupied Machines at the Start of the Cycle*

Due to the special conditions of the CJS-RCS problem, some of the machines may be occupied by the processing operations of related jobs at the beginning of the cycle. Figure 5.1 indicates the occupied two machines and the occupying jobs for a feasible solution of the instance problem.

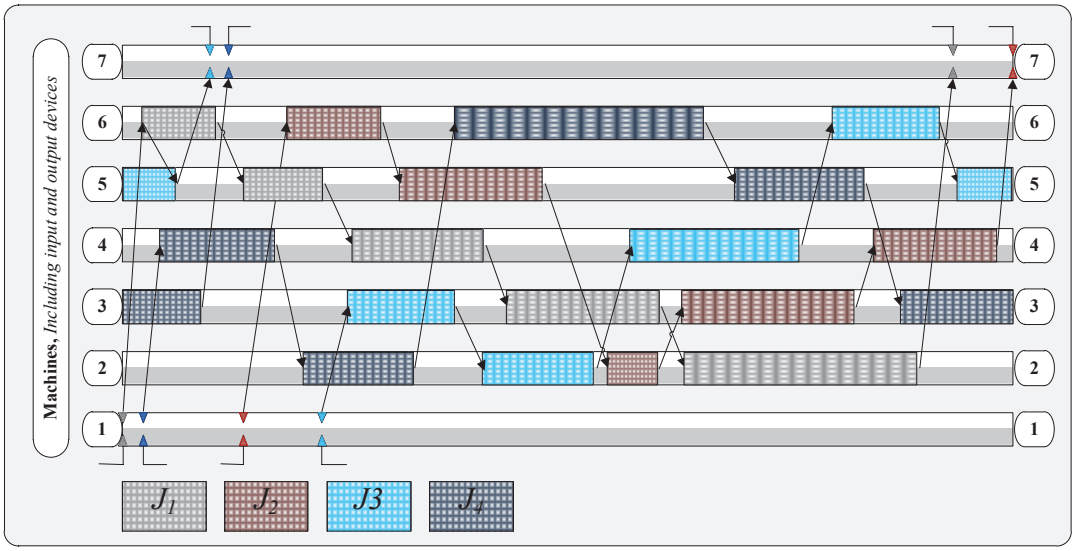



Figure 5.1 The Gantt chart of the feasible solution for the instance problem (4 jobs and 7 machines including input and output devices).

Table 5.4 The start time of transportation operations for the feasible solution of the instance problem (Figure 5.1).

| **Job** | **Start time of transportation operations $O_{j,i}$** | | | | | |
|---|---|---|---|---|---|---|
| | **1** | **2** | **3** | **4** | **5** | **6** |
| **1** | 0 | 34 | 73 | 131 | 196 | 290 |
| **2** | 44 | 94 | 153 | 195 | 267 | 319 |
| **3** | 72 | 121 | 172 | 247 | 298 | 19 |
| **4** | 7 | 55 | 106 | 212 | 271 | 28 |

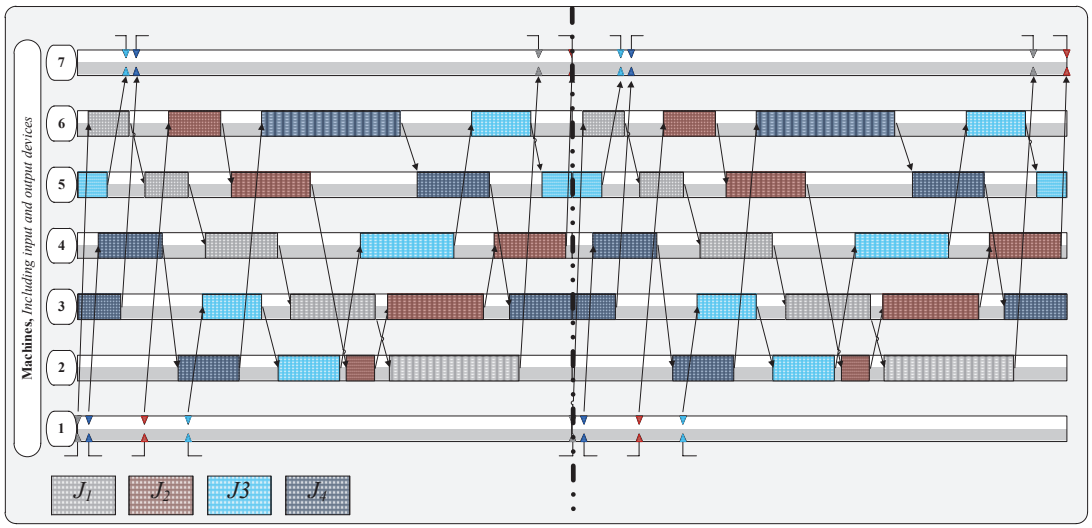



Figure 5.2 The Gantt chart of two consecutive cycles for the instance problem.

While solving the CJS-RCS problem, the precedence constraints between the transportation operations of each job should be taken into account. In order to identify these precedence constraints, the solution procedure has to determine the number of occupied machines at the start of the cycle, which is the first sub-problem of the CJS-RCS problem. The graph representation of first nodes set of proposed MB-ACO for this sub-problem has been indicated in Figure 5.3, by which the number of occupied machines is determined.

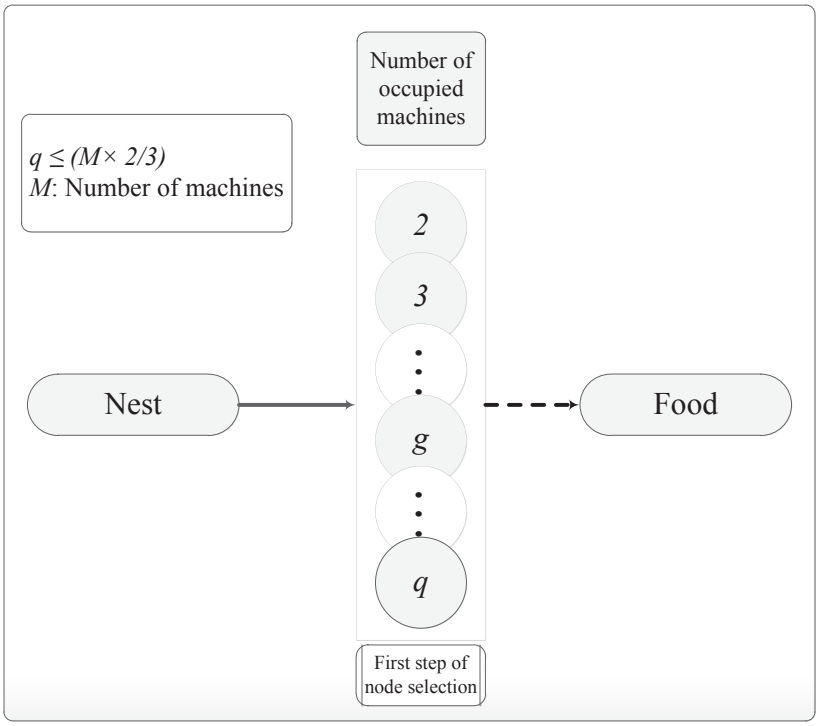



Figure 5.3 The first set of nodes that determines the number of occupied machines at the start time of the cycle.

As the second sub-problem, the second nodes set are considered through which the ants path to determine each one of the occupied machines. The graph representation of second nodes set of the MB-ACO is illustrated in Figure 5.4.

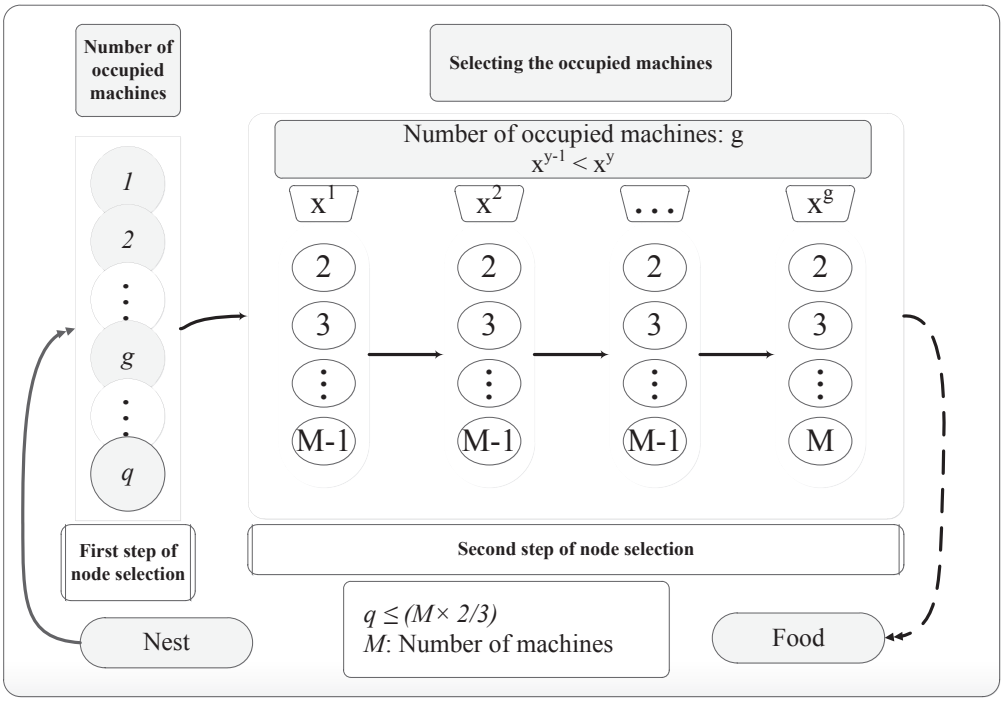



Figure 5.4 Determining each one of the occupied machines at the start time of the cycle.

Let us consider the set of occupied machines ($SOM$) at the beginning of the cycle in the increasing order. The third sub-problem is to determine the related $a^{th}$ job that occupies each selected machine ($m^x \in SOM$) and the third set of nodes representing this sub-problem is shown in Figure 5.5.

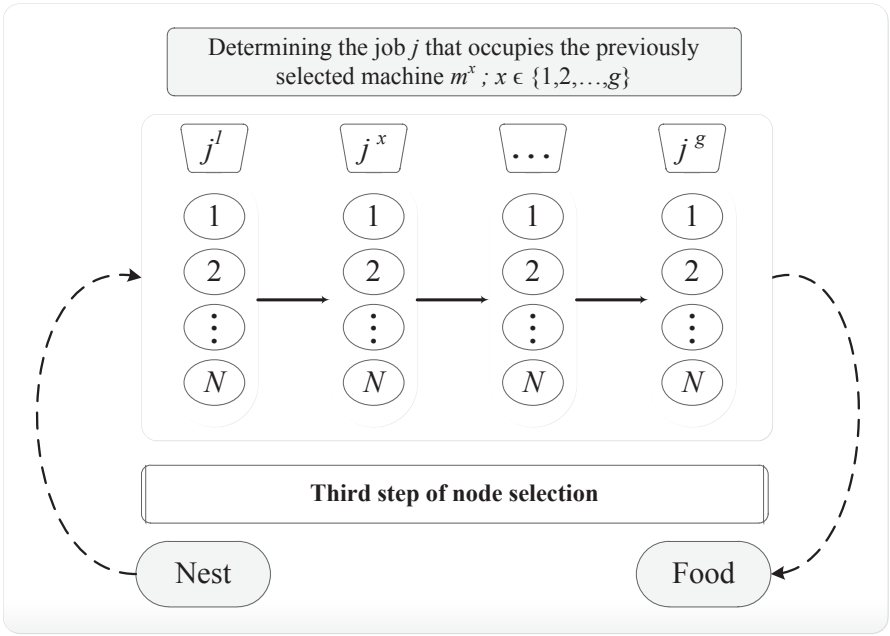



Figure 5.5 Determining the jobs that occupy the machines at the start time of the cycle.

The set of occupied machines for the feasible solution of the instance problem in Figure 5.1 is $SOM = \{3, 5\}$ in, where $(|SOM| = g = 2)$, and the jobs that occupy these machines are $\{4, 3\}$, respectively.

At this point, the transportation operations of the jobs should be considered. Assume that the $a^{th}$ job occupies only the $m^{th}$ machine at the beginning of the cycle. Then the transportation operations of the $a^{th}$ job are divided to two sub-sets as follows:

$$m \propto i' \mid i' \in \{1, \dots, n_a - 1\} \Longrightarrow \begin{cases} \{[a, i] | i \in \{1, \dots, n_a - 1\}, i < i'\} \\ \{[a, i] | i \in \{1, \dots, n_a - 1\}, i' \leq i\} \end{cases} \tag{5.20}$$

Where the $m^{th}$ machine $(m \in \{m_1, m_2, \dots, m_{M-1}\})$ performs the $i'^{th}$ operation of the $a^{th}$ job.

In addition, if the $a^{th}$ job occupies $(y)$ number of machines at the beginning of the cycle then:

$$\begin{cases} occupied\ by\ the\ a^{th} job: \\ \{m_1, m_2, \dots, m_y\} \propto \{i'_1, i'_2, \dots, i'_y\} \\ \forall\ i \in \{1, \dots, n_a - 1\} \end{cases} \Longrightarrow \begin{cases} \{[a, i] |\ i < i'_1\} \\ \{[a, i] |\ i'_1 \leq i < i'_2\} \\ \vdots \\ \{[a, i] |\ i'_{y-1} \leq i < i'_y\} \\ \{[a, i] |\ i'_y \leq i\} \end{cases} \tag{5.21}$$

Thereupon, the number of sub-sets for the consequent transportation operations of each job can be determined. Also, if the $a^{th}$ job occupies $y$ number of machines at the start of the cycle, then it stays during $y$ number of consecutive cycles in the system. Therefore, the mentioned $y$ number here can be defined as the height of the $a^{th}$ job. It can be concluded that the third step of the proposed MB-ACO algorithm provides the facility of considering different height values for the jobs independently. The jobs 3 and 4 of the instance problem in Figure 5.1 are divided to two sub-sets for their transportation operations.

Table 5.5 The sub-sets for the consequent transportation operations of each job in Figure 5.1.

| Job | Sub-sets | Transportation operation $[a, i]$ |
|---|---|---|
| 1 | 1 | $i \in \{1,2,3,4,5,6\}$ |
| 2 | 1 | $i \in \{1,2,3,4,5,6\}$ |
| 3 | 1 | $i \in \{1,2,3,4,5\}$ |
| | 2 | $i \in \{6\}$ |
| 4 | 1 | $i \in \{1,2,3,4,5\}$ |
| | 2 | $i \in \{6\}$ |

### *5.4.2 Sequencing the Transportation Operations*

Maximizing the throughput as an objective function in CJS-RCS problem requires sequencing the transportation operations optimally. Considering that the *N*-job cycle has $Q = \sum_{j=1}^{N}(n_j - 1)$ number of transportation operations as $\{[j,i]|j \in \{1,2,\dots,N\}, i \in \{1,\dots,n_j - 1\}\}$, the forth nodes set provides the paths to generate the sequence of transportation operations for the artificial ants as given in Figure 5.6.

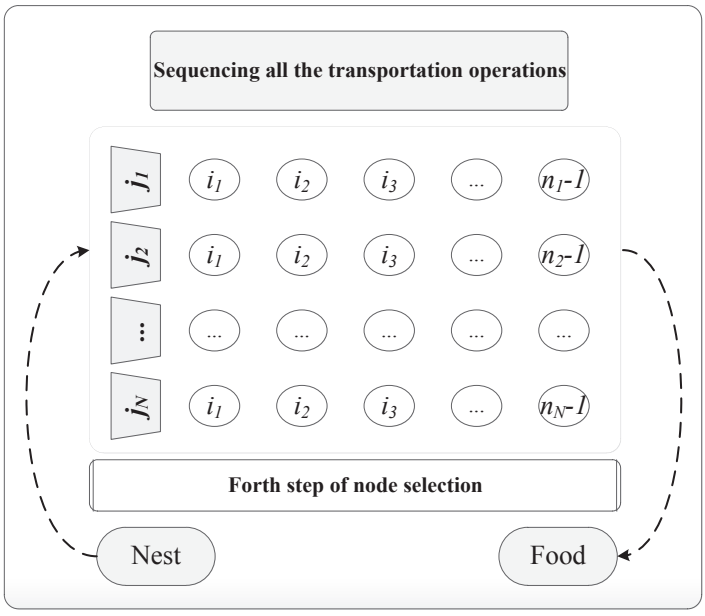



Figure 5.6 Determining the sequence of $\sum_{j=1}^{N}(n_j - 1)$ number of transportation operations for the *N*-job cycle (fourth step).

As an example, Figure 5.7 illustrates the precedence relations for the instance problem in Figure 5.1 considering the sub-sets provided in Table 5.5.

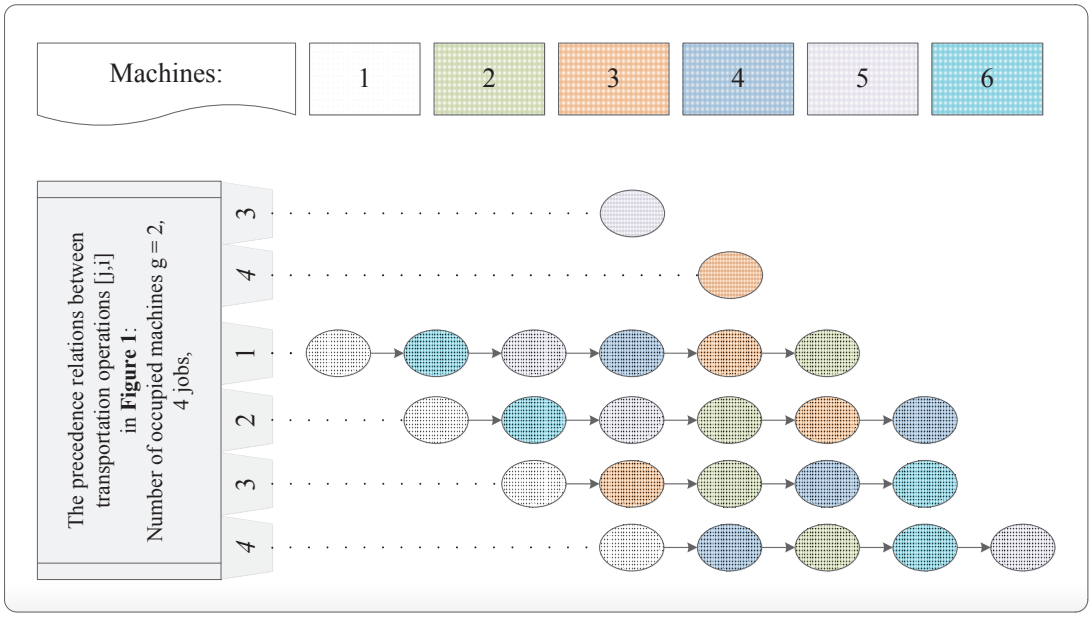



Figure 5.7 The precedence relations for the provided instance problem (Figure 5.1).

### *5.4.3 Robot Assignments for the Transportation Operations*

The CJS-RCS problem considers multiple single gripper robots to perform transportations between the machines. The limited transportation resources affect the sequence of transportation operations, cycle time and throughput rate. The fifth nodes set of the MB-ACO is used to determine which robot to assign to perform the transportation operations, such that the throughput rate is maximized by the appropriate assignments.

The assigned robot to perform each transportation operation would be selected due to the pheromone trails of different robots for the considered operation. Figure 5.8 provides the graph representation of the fifth nodes set of the proposed MB-ACO.

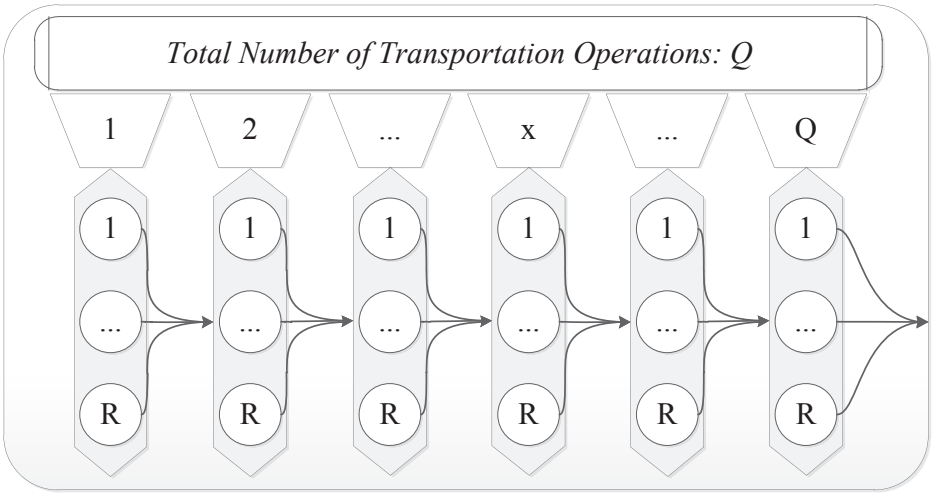



Figure 5.8 Determining the assigned robot to perform the $x^{th}$ transportation operation.

The assigned robots for all the $Q = \sum_{j=1}^{N}(n_j - 1)$ number of transportation operations in the considered instance problem for the generated feasible solution is provided in Table 5.6. In addition, Figure 5.9 illustrates the Gantt chart of all transportation operations and their assigned robots.

Table 5.6 The assigned robot to perform each transportation operation of the instance problem.

| **Job 1** | **Robots** | | **Job 2** | **Robots** | | **Job 3** | **Robots** | | **Job 4** | **Robots** | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | **1** | **2** | | **1** | **2** | | **1** | **2** | | **1** | **2** |
| [1,1] | | ✓ | [2,1] | ✓ | | [3,1] | | ✓ | [4,1] | ✓ | |
| [1,2] | | ✓ | [2,2] | ✓ | | [3,2] | ✓ | | [4,2] | | ✓ |
| [1,3] | ✓ | | [2,3] | | ✓ | [3,3] | ✓ | | [4,3] | | ✓ |
| [1,4] | | ✓ | [2,4] | | ✓ | [3,4] | ✓ | | [4,4] | ✓ | |
| [1,5] | ✓ | | [2,5] | | ✓ | [3,5] | ✓ | | [4,5] | ✓ | |
| [1,6] | | ✓ | [2,6] | ✓ | | [3,6] | | ✓ | [4,6] | ✓ | |

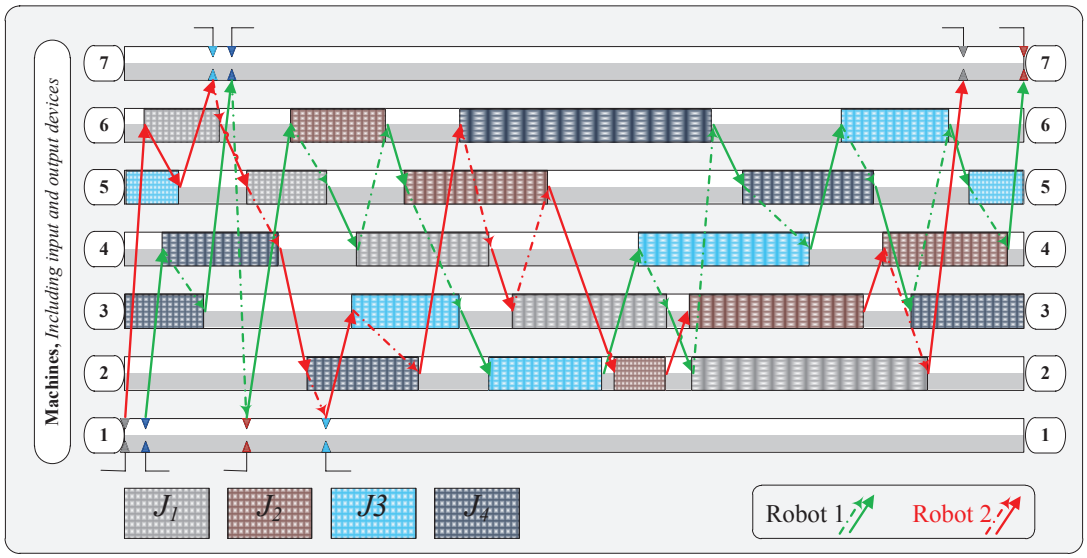



Figure 5.9 The Gantt chart of the instance problem indicating the sequence of all transportation operations and their assigned robots.

The general structure of the proposed MB-ACO for the CJS-RCS problem and the different regions as sub-problems are illustrated in Figure 5.10. The artificial ants have to generate a path at each different region before crossing the bridge to the next region. In addition, Figure 5.11 provides the flow chart of the proposed MB-ACO algorithm for the CJS-RCS problem.

Figure 5.10 The general structure of the proposed MB-ACO algorithm and the regions that the artificial ants have to visit.

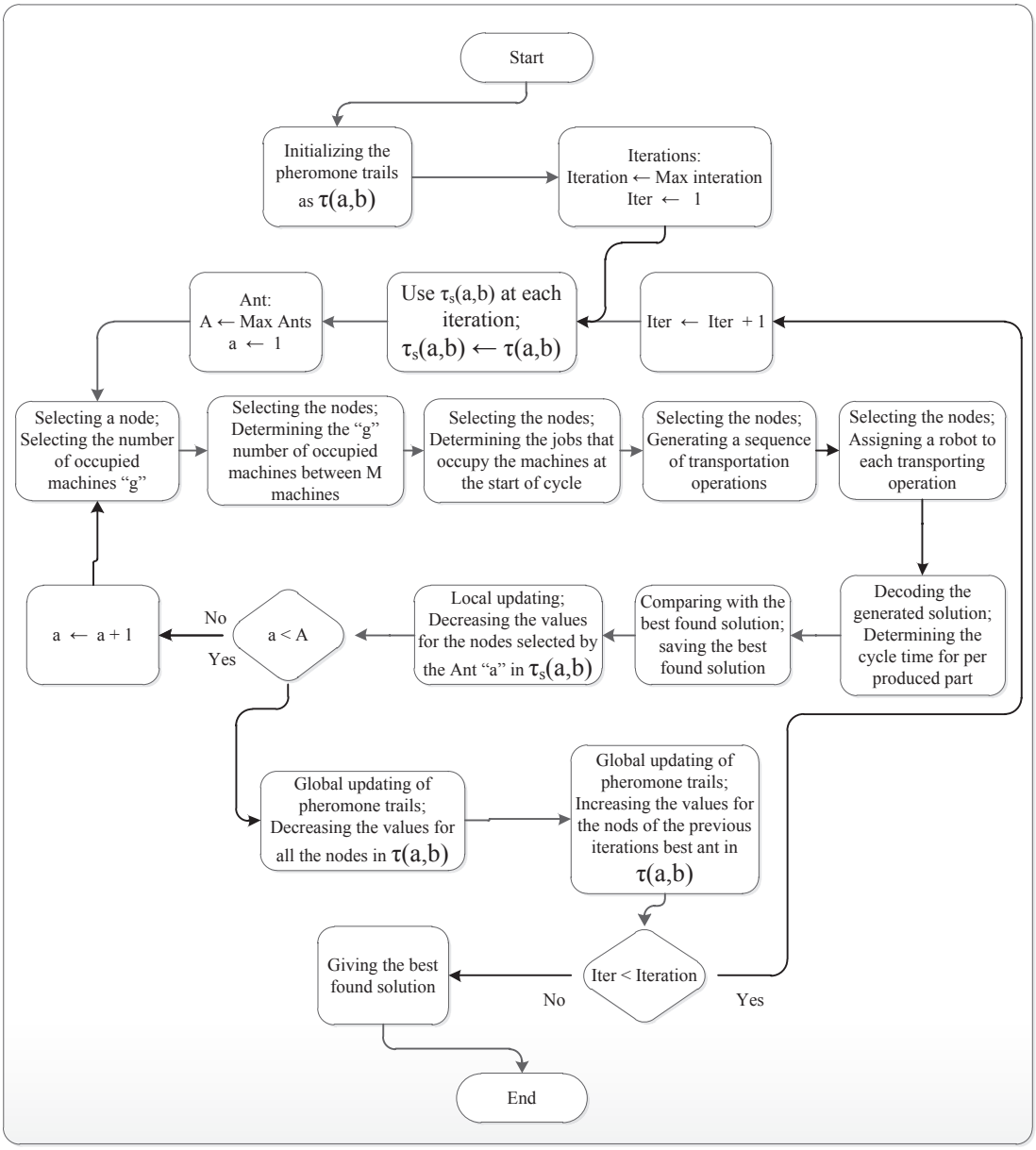



Figure 5.11 The flow chart of the proposed MB-ACO algorithm for the considered CJS-RCS problem.

### *5.4.4 Encoding Feasible Solution*

A feasible solution for the CJS-RCS problem is generated considering the selected number of occupied machines and the machines which are occupied at the start of the cycle. The feasible solution is determined as the permutation of all $Q = \sum_{j=1}^{N}(n_j - 1)$ transportation operations. This kind of representation uses a permutation in which unload operations are sequenced to specify in which order they

are performed. The length of the representation is the total number of unload operations performed by the robots. It is assumed that each unload operation will also include transportation to the next related machine and loading on it. It should also be noted that the permutation of transportation operations is generated considering the precedence relations which were discussed previously. Thereupon, the operations sequence (illustrated by $\delta(f)$ in Figure 5.12) that will be unloaded by the robots is generated in the fourth region of the proposed MB-ACO algorithm. Moreover, the assigned robot to perform each transportation operation is determined during the fifth region of the algorithm and saved in the matrix $RoTO_{[j,i]}$ for each $[j, i]$ (Figure 5.13).

| $\delta(f)^{job}$ | $j_1$ | $j_4$ | $j_3$ | $j_4$ | $j_1$ | $j_2$ | $j_4$ | ... | ... |
|---|---|---|---|---|---|---|---|---|---|
| $\delta(f)^{operation}$ | 1 | 1 | 6 | 6 | 2 | 1 | 2 | ... | ... |
| $f$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | ... | $4 \times 6$ |

Figure 5.12 Encoding the feasible solution; the sequence of transportation operations.

| $[j, i]$ | | Job $j$; $j \in \{1, 2, \dots, N\}$ | | | | |
|---|---|---|---|---|---|---|
| | | **1** | **2** | .. | .. | ***N*** |
| **Unloading the $i^{th}$ operation** | **1** | $RoTO_{[1,1]}$ | $RoTO_{[2,1]}$ | .. | .. | $RoTO_{[N,1]}$ |
| | **2** | $RoTO_{[1,2]}$ | $RoTO_{[2,2]}$ | .. | .. | $RoTO_{[N,2]}$ |
| | .. | .. | .. | .. | .. | .. |
| | .. | .. | .. | .. | .. | .. |
| | $n_j - 1$ | $RoTO_{[1,n_1-1]}$ | $RoTO_{[2,n_2-1]}$ | .. | .. | $RoTO_{[N,n_N-1]}$ |

Figure 5.13 Encoding the feasible solution; the assigned robot to perform each transportation operation $[j, i]$.

### *5.4.5 Decoding Algorithm*

Each artificial ant generates a feasible solution for the CJS-RCS problem. The sequence of transportation operations is generated according to the previously determined occupied machines and the height of the jobs. Therefore, to obtain the cycle time and to decode the generated feasible solution, the sequence of transportation operations and the assigned robots to perform each one are needed. The specifications of encoded feasible solution and the pseudo code of the decoding algorithm are provided in the following.

*Indices and sets:*

$N$ the number of parts that enter during a cycle, where $j \in \{J_1, J_2, \dots, J_N\}$.

$M$ number of machines including input device, $m \in \{m_1, m_2, \dots, m_{M-1}\}$.

$Q$ the total number of transportation operations, $Q = \sum_{j=1}^{N} (n_j - 1)$.

$f$ the index for the transportation operations, $f \in \{1,2, \dots, Q\}$.

$d$ the index for the occupied machines, where $d \in \{1,2, \dots, g\}$ and $g$ is the determined number of occupied machines at the start time of the cycle.

*Parameters:*

$P_{j,i}$ the processing time of the $i^{th}$ operation of the $j^{th}$ job, where the processing times of all jobs at input and output devices are equal to zero.

$\theta_{j,i}$ the handling time of the $j^{th}$ job between the $i^{th}$ and $i + 1^{th}$ operations.

$\delta_{m,m'}$ the time required for the robots to perform void move $[\![m, m']\!]$.

$LgNum$ sufficiently large number.

*Specifications of the feasible solution:*

$\delta(f)$ the transportation operation in the $f^{th}$ position of generated sequence that saves the related job $j$ and operation $i$, $[j, i]$.

$Occ_Mch_d$ the set of occupied machines (the $d^{th}$ occupied machine).

$Cl_OccMch_d$ the job that occupies the occupied machine $Occ_Mch_d$ (the $d^{th}$ occupied machine).

$RoTO_{j,i}$ the matrix that provides the assigned robot for each transportation operation.

*Determined by decoding algorithm:*

$StartTO_{j,i}$ the start time of each transportation operation.

$T$ the cycle time for all the jobs.

**Begin**

Initializing

$Q \leftarrow \sum_{j=1}^{N} (n_j - 1)$

$f \leftarrow Q + 1$

**Repeat**

$f \leftarrow f - 1$

$j \leftarrow \delta(f)^{job}$

$i \leftarrow \delta(f)^{operation}$

$m \leftarrow (j,i)^{machine}$

**Repeat** *// determining the precedence relations of transportation operations.*

$f' \leftarrow f + 1$

$j' \leftarrow \delta(f')^{job}$

$i' \leftarrow \delta(f')^{operation}$

$m' \leftarrow (j',i')^{machine}$

$temp_1, temp_2, temp_3 \leftarrow 0$

**If** $(j' = j)$ **&&** $(i' = i + 1)$ **then:**

$temp_1 \leftarrow StartTO_{j,i+1} - P_{j,i+1} - \theta_{j,i}$ *// precedence relations of two successive transportation operations of job* $j$.

**If** $(m' = m)$ **then:**

$temp_2 \leftarrow StartTO_{j',i'-1} + \theta_{j',i'-1}$ *// precedence relations of transportation operations related to the same machine* $m$.

**If** $(RoTO_{j',i'} = RoTO_{j,i})$ **then:**

$m'' \leftarrow (j, i+1)^{machine}$

$temp_3 \leftarrow StartTO_{j',i'} - \delta_{[\![m'',m']\!]} - \theta_{j,i}$ // *precedence relations of transportation operations related to the same robot* $r$.

**Until** $(f' = Q)$

$StartTO_{j,i} \leftarrow \boldsymbol{min}\ \{LgNum,\ temp_1,\ temp_2,\ temp_3\}$ // *determining the start time of transportations due to the precedence relations.*

**Until** $(f = 1)$

$TEMP \leftarrow \boldsymbol{min}\{\ StartTO_{j,i},\ \ \boldsymbol{for\ all}\ [j,i]\}$

$StartTO_{j,i} \leftarrow (StartTO_{j,i} - TEMP);\ \boldsymbol{for\ all}\ [j,i]$

$d \leftarrow 0$

**Repeat**

$d \leftarrow d + 1$

$m \leftarrow Occ_Mch_d$

$j \leftarrow Cl_OccMch_d$

$\boldsymbol{if}\ (m == (j,i)^{machine})$

$temp_1 \leftarrow \boldsymbol{max}\{\ StartTO_{j,i-1} + \theta_{j,i-1} + P_{j,i} - StartTO_{j,i}\}$

**Until** $(d = |Occ_Mch|)$

$temp_2 \leftarrow (StartTO_{j,i} + \theta_{j,i})\ ;\ \boldsymbol{for\ all}\ [j,i];$

$T \leftarrow \boldsymbol{max}\{\ temp_1,\ \ \ \ temp_1\}$

**End**

### 5.5 Computational Analysis

The computational experiments have been carried out on a set of randomly generated instances. The problem sizes are determined by varying the levels of three parameters, which are respectively the number of jobs, the processing machines and the number of single gripper robots. There are three groups of problems with respect to the number of jobs in the robotic cell, which are 10, 15, and 20, respectively. The maximum height of the cycles are considered to be 5 for all three groups of problems.

In order to assess the empirical performance of the proposed MB-ACO algorithm, the following data sets are generated from the different domains of the uniform distribution $[a, b]$ for each problem, separately.

- The standard processing times of the machines are generated from the uniform distribution $U[100, 200]$.
- The time needed for the loaded robot moves between the successive machines, which are considered as transportation operations, are generated using the uniform distribution $U[25, 50]$.
- The time needed for the unloaded robot moves between the machines, which are considered as empty moves, are generated from the uniform distribution $U[10, 25]$.

Moreover, the number of single gripper robots is considered at different levels; min-1 and max-10. Table 5.7, Table 5.8, and Table 5.9 provide the problems and the characteristics of obtained solutions for each problem (a total of 63 problems). The proposed MB-ACO algorithm for the CJS-RCS problem was coded in C#.NET and run on a Pentium 4, 100 GHz PC.

Table 5.7 The generated 10-job problem and the proposed MB-ACO algorithm's obtained solutions.

| **Problems** | | | **MB-ACO algorithm** | | | **No** |
|---|---|---|---|---|---|---|
| ***N*** ***#jobs*** | ***M*** ***#machines*** | ***R#robots*** | ***T*** ***(sec)*** | ***Height of the cycle*** | ***Run Time*** ***(sec)*** | |
| 10 | 5 | 1 | 912 | 2 | 58.47 | 1 |
| | | 2 | 885 | 2 | 63.71 | 2 |
| | | 3 | 881 | 2 | 65.88 | 3 |
| | | 4 | 881 | 2 | 69.35 | 4 |
| | 10 | 1 | 1528 | 2 | 91.79 | 5 |
| | | 2 | 1429 | 2 | 94.31 | 6 |
| | | 3 | 1403 | 2 | 95.84 | 7 |
| | | 4 | 1403 | 2 | 105.74 | 8 |
| | | 5 | 1389 | 3 | 110.18 | 9 |
| | 15 | 1 | 2473 | 2 | 117.64 | 10 |
| | | 2 | 2193 | 3 | 128.04 | 11 |
| | | 3 | 2075 | 3 | 128.61 | 12 |
| | | 4 | 1894 | 3 | 133.90 | 13 |
| | | 5 | 1894 | 3 | 135.58 | 14 |
| | | 6 | 1865 | 3 | 139.48 | 15 |

Table 5.8 The generated 15-job problem and the proposed MB-ACO algorithm's obtained solutions.

| Problems | | | MB-ACO algorithm | | | No |
|---|---|---|---|---|---|---|
| *N* | *M* | *R* | *T (sec)* | *Height of the cycle* | *Run Time (sec)* | |
| 15 | 10 | 1 | 2741 | 2 | 119.31 | 16 |
| | | 2 | 2458 | 2 | 122.57 | 17 |
| | | 3 | 2397 | 2 | 129.13 | 18 |
| | | 4 | 2315 | 2 | 130.90 | 19 |
| | | 5 | 2154 | 3 | 136.11 | 20 |
| | | 6 | 2154 | 3 | 141.84 | 21 |
| | 15 | 1 | 3461 | 2 | 191.53 | 22 |
| | | 2 | 3315 | 3 | 230.36 | 23 |
| | | 3 | 3274 | 2 | 238.61 | 24 |
| | | 4 | 3209 | 2 | 257.74 | 25 |
| | | 5 | 3041 | 3 | 250.93 | 26 |
| | | 6 | 2861 | 4 | 269.27 | 27 |
| | | 7 | 2861 | 4 | 269.89 | 28 |
| | 20 | 1 | 4619 | 2 | 351.11 | 29 |
| | | 2 | 4185 | 2 | 367.83 | 30 |
| | | 3 | 4053 | 2 | 391.02 | 31 |
| | | 4 | 3924 | 3 | 404.15 | 32 |
| | | 5 | 3848 | 3 | 419.58 | 33 |
| | | 6 | 3764 | 4 | 421.73 | 34 |
| | | 7 | 3721 | 3 | 439.33 | 35 |
| | | 8 | 3692 | 4 | 444.72 | 36 |
| | | 9 | 3692 | 4 | 447.25 | 37 |

Table 5.9 The generated 20-job problem and the proposed MB-ACO algorithm's obtained solutions.

| Problems | | | MB-ACO algorithm | | | No |
|---|---|---|---|---|---|---|
| *N* | *M* | *R* | *T (sec)* | *Height of the cycle* | *Run Time (sec)* | |
| 20 | 15 | 1 | 4725 | 2 | 373.59 | 38 |
| | | 2 | 4561 | 3 | 385.62 | 39 |
| | | 3 | 4188 | 3 | 407.25 | 40 |
| | | 4 | 4067 | 3 | 421.67 | 41 |
| | | 5 | 3966 | 3 | 423.58 | 42 |
| | | 6 | 3941 | 3 | 431.63 | 43 |
| | | 7 | 3780 | 3 | 449.01 | 44 |
| | | 8 | 3771 | 3 | 460.72 | 45 |
| | 20 | 1 | 5658 | 3 | 527.55 | 46 |
| | | 2 | 5391 | 3 | 530.31 | 47 |
| | | 3 | 5261 | 4 | 542.64 | 48 |
| | | 4 | 5158 | 3 | 566.11 | 49 |
| | | 5 | 5086 | 3 | 579.83 | 50 |
| | | 6 | 4841 | 4 | 581.03 | 51 |
| | | 7 | 4632 | 4 | 579.64 | 52 |
| | | 8 | 4604 | 4 | 587.77 | 53 |
| | 25 | 1 | 6194 | 3 | 637.51 | 54 |
| | | 2 | 5793 | 3 | 639.84 | 55 |
| | | 3 | 5722 | 4 | 648.60 | 56 |
| | | 4 | 5609 | 3 | 656.97 | 57 |
| | | 5 | 5596 | 4 | 671.02 | 58 |
| | | 6 | 5370 | 4 | 675.24 | 59 |
| | | 7 | 5307 | 4 | 688.39 | 60 |
| | | 8 | 5184 | 4 | 695.41 | 61 |
| | | 9 | 5177 | 5 | 710.85 | 62 |
| | | 10 | 5169 | 4 | 719.06 | 63 |

Figures 5.14, 5.15, and 5.16 illustrate the impact of the number of robots on the cycle time for each group of jobs 10, 15, and 20, respectively. Accordingly, the influence of the number of robots on the cycle time decreases as the robot number increases. Figure 5.17 represents the determined maximum height value separately

for each problem with different number of jobs and machines. Accordingly, the increase in the maximum height value of cycle is observed at large size problems due to the number of machines and jobs (Figure 5.18).

It can be also said that the appropriate height of the cycle increases as the number of robots is increased. Larger height values involve a larger number of robots support when the number of machines in the robotic cells increases.

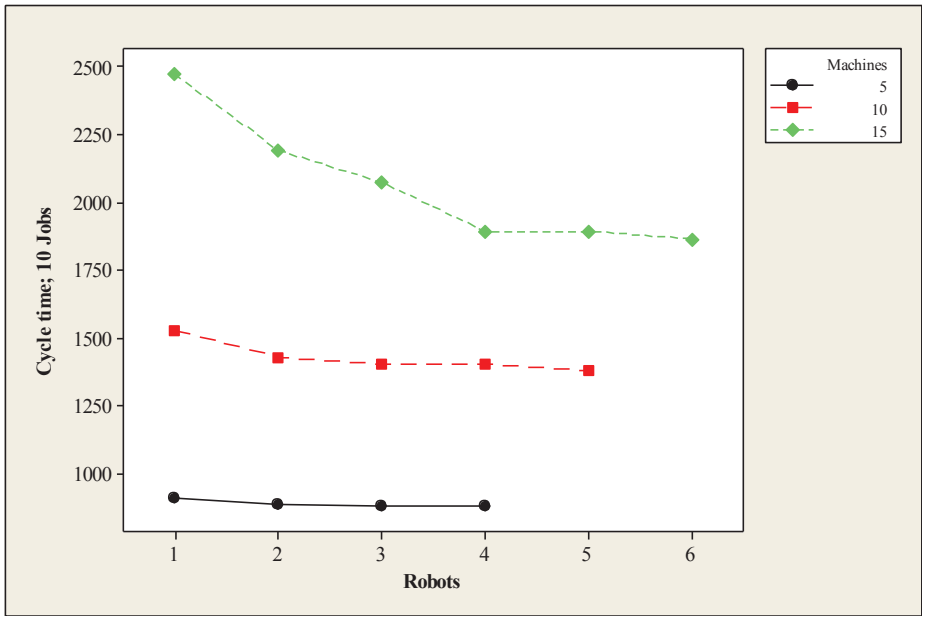



Figure 5.14 The obtained cycle time by the proposed MB-ACO for 10 jobs.

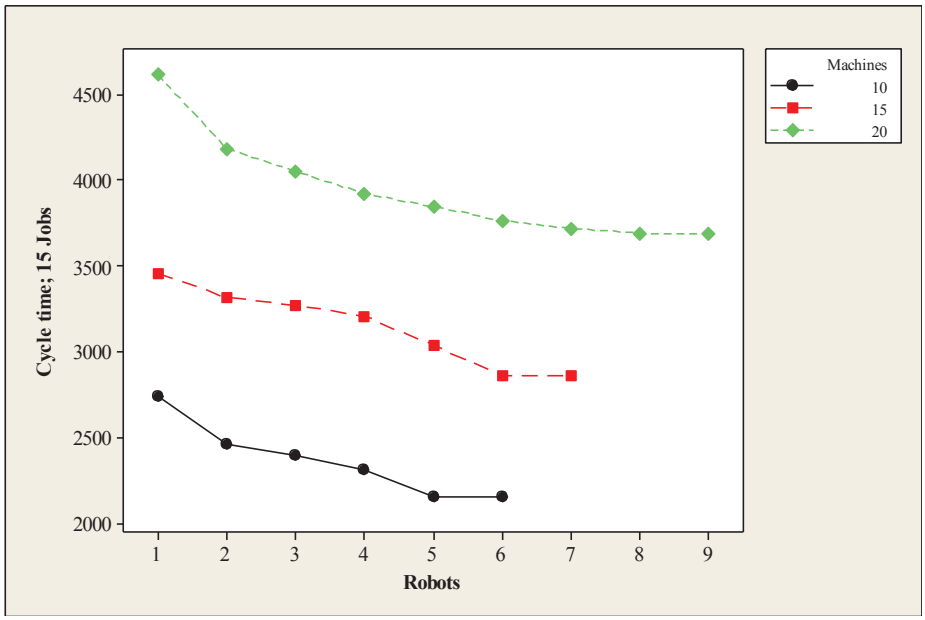



Figure 5.15 The obtained cycle time by the proposed MB-ACO for 15 jobs.

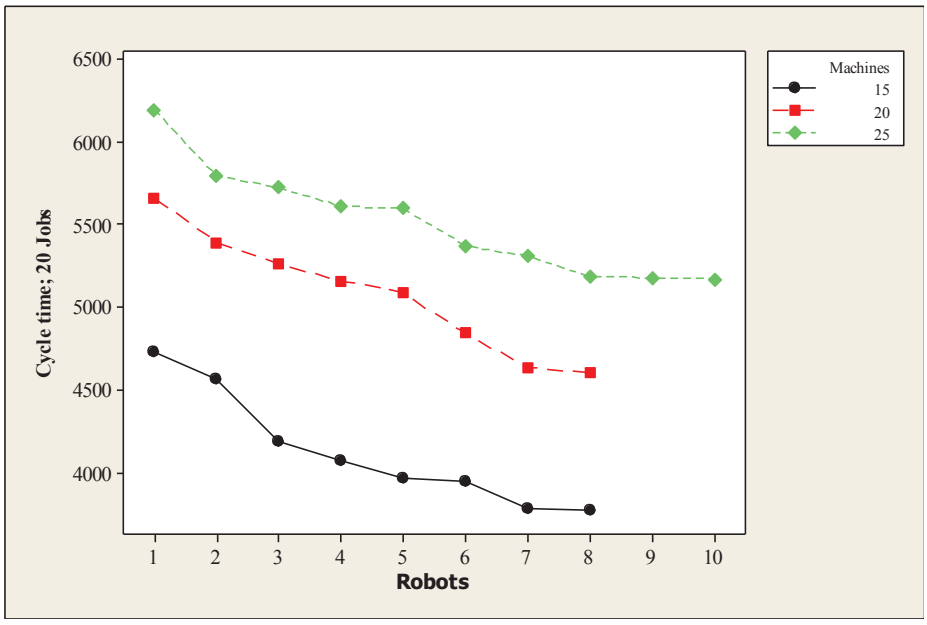



Figure 5.16 The obtained cycle time by the proposed MB-ACO for 20 jobs.

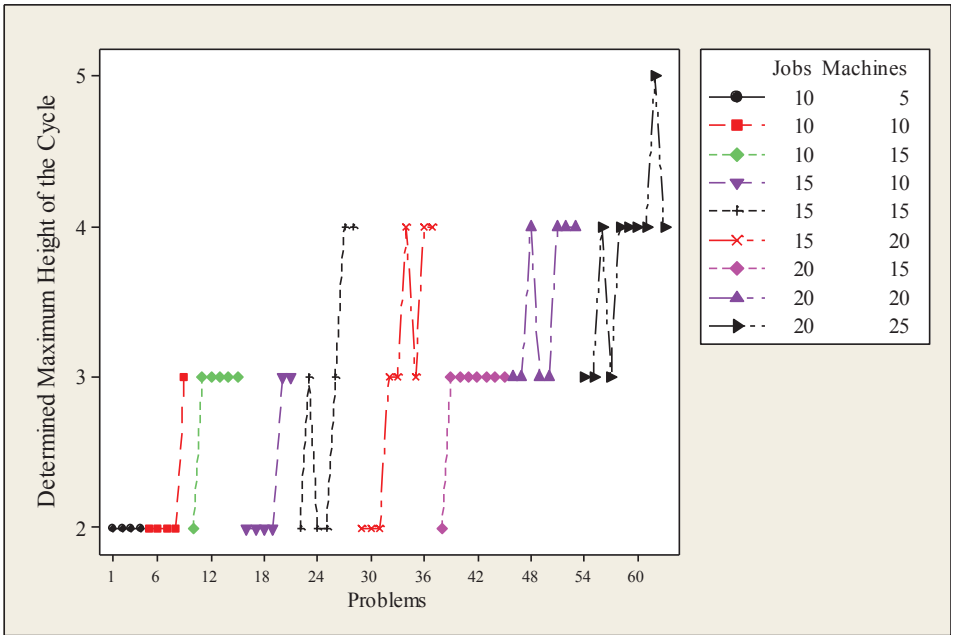



Figure 5.17 The determined maximum height value of the cycle for different number of jobs and machines by the proposed MB-ACO algorithm.

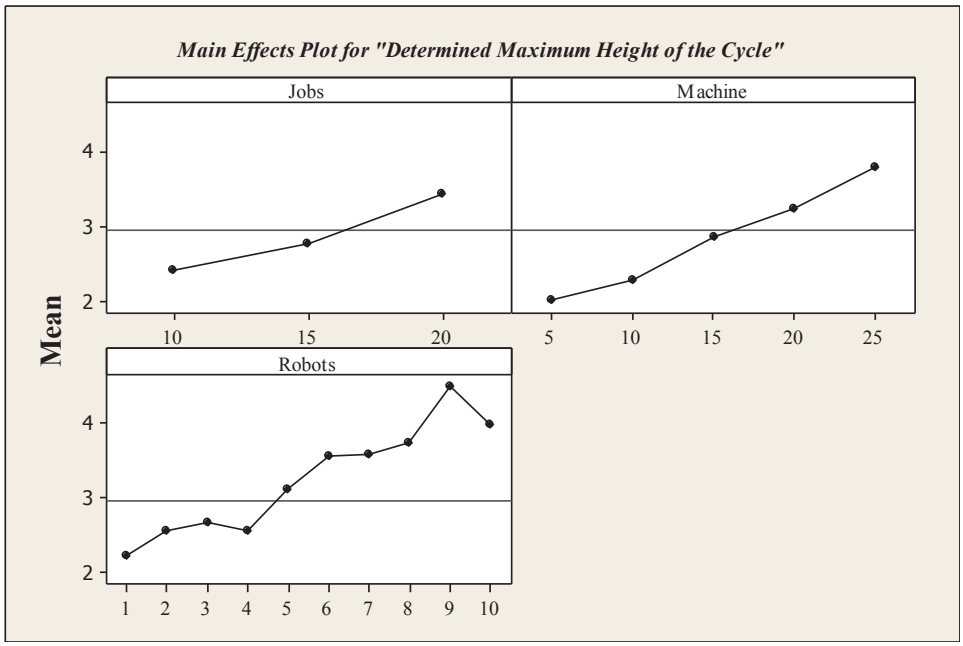



Figure 5.18 The Main Effects Plot for determined maximum height value of the cycle.

## 5.6 Conclusion

This research studies the cyclic job shop robotic cell scheduling (CJS-RCS) problem considering multiple single gripper robots. Since the investigated problem is hard to solve with exact methods for real-size problems, an ant colony optimization-based algorithm (MB-ACO) is proposed for its solution in order to minimize the cycle time, and hence to maximize the throughput rate.

To evaluate the performance of the proposed algorithm, nine problem categories are considered to be analyzed under different numbers of robots usage (63 problems in total), and the generated problems are solved by the proposed MB-ACO algorithm. Besides, the special characteristics of CJS-RCS problem are stated by an instance problem given. The computational results confirm the efficiency of different height values on the cycle time and on throughput rate. According to the conducted computational study, the proposed MB-ACO algorithm has solved the investigated CJS-RCS problem efficiently and in reasonable computation time for realistic problem sizes, and this feature makes it applicable to real-life problems.

# CHAPTER SIX
# CONCLUSION

In this thesis, we have studied the throughput optimization in robotic cells. As far as the robotic cell scheduling literature is concerned, , this thesis covers those problems which have not been studied yet by other researchers. The importance of these problems and their applications in different manufacturing systems is one reason why they have been chosen for this thesis.

Chapter 2 introduces the Blocking Hybrid Flow Shop Robotic Cell Scheduling (BHFS-RCS) problem with unrelated parallel machines and machines eligibility constraints. The problem incorporates additional real world constraints such as machines with different processing speeds at each stage, machine eligibility constraints and transportation constraints.

$RF_s(m_1, \dots, m_s)|(\text{free}, \text{A}, \text{MP}, \text{non} - \text{cyclic})|Cmax$

Section 2.3 addresses the robotic scheduling problem in BHFS-RC that considers a single transportation robot to move the parts between stages. Initially, a mixed integer linear programming (MILP) model is proposed to minimize the makespan. Due to the complexity of the model, a simulated annealing (SA) based solution approach is developed to solve the problem. This approach uses both simple insertion method ($SA^{IN}$) and a new neighborhood structure based on block properties ($SA^{BP}$) while generating neighbor solutions, which yields two different SA algorithms respectively. The performance of the proposed SA approach is assessed over a set of randomly generated instances. The computational results demonstrate that the SA algorithm is effective with the employed neighborhood structure for this problem.

$RF_{s,r}(m_1, \dots, m_s)\big|(\text{free}, \text{A}, \text{MP}, \text{non} - \text{cyclic})|Cmax$

In addition, the proposed MILP model and $SA^{IN}$ and $SA^{BP}$ algorithms for one single-gripper robot have been extended to multiple single-gripper robots in Section 2.4. To evaluate the performance of the proposed algorithms, randomly generated

problems have been analyzed by different numbers of robots and solved by both proposed $SA^{IN}$ and $SA^{BP}$ algorithms. Also, small size problems are used to compare the performance of $SA^{BP}$ with the optimal solutions. The concluding remarks of Chapter 2 can be outlined as follows.

- As Encoding and Decoding algorithms are relatively dependant, there may be infeasible solutions while generating neighbor solutions. Therefore, the Rectifying algorithm has been proposed to rectify infeasible solutions.
- The proposed Decoding algorithm is also a constructive algorithm that determines the best machine at each stage to perform the needed operation on each particular part.
- Two kinds of neighborhood structures are used in this paper for $SA^{IN}$ and $SA^{BP}$ algorithms based on simple insertion method and block properties respectively. The efficient neighborhood generation, insertion method based on block properties, has been used for the first time for the hybrid flow shop scheduling problem. This method determines both critical processing and transportation operations in the critical path.
- The proposed $SA^{IN}$ and $SA^{BP}$ algorithms have been tested on 100 problem instances. $SA^{BP}$ algorithm has shown to be a successful heuristic in solving the problem. According to the computational results, $SA^{BP}$ algorithm reduced the makespan by 8.2 % over $SA^{IN}$ algorithm. As a result, using block properties made the search process more efficient for the BHFS-RCS problem.
- Additional number of robots results in significant reduction in the makespan values when the number of stages and machines increases in the cell. It means that increasing the number of robots in cells that contain larger number of stages and machines will be more effective in comparison to those with smaller number of stages and machines.
- The maximum number of robots needed would not be larger than the maximum number of machines utilized in the cell and additional robots used would not be operative any more.

- The sequence of processing operations and unload operations influence the makespan simultaneously. Hence, the proposed SA algorithm sequences both of them in parallel to optimize the BHFS-RCS problem in an efficient way.
- According to the obtained solutions and the sequence of operations, it has been found that the appropriate number of robots for the BHFS-RCS problem depends on the sequence of processing operations of parts performed at each stage.

$RF_{s,r}|(\text{interval}, \text{C}, \text{Cyclic})|\mu$

Chapter 3 deals with the cyclic flow shop robotic cell scheduling problem with multiple robots, where parts are processed successively on multiple machines with lower and upper bounds on processing times and the robots execute the transportation of parts among the machines. A novel MILP programming model for the multi-degree cyclic flow shop robotic cell scheduling (*k*DCFS-RCS) problem with multiple robots has been proposed. The proposed model simultaneously determines the optimal degree of the cyclic schedule and the optimal sequencing of the robots moves, which in return maximizes the throughput rate. The validity of the proposed model is examined by a computational study on a set of randomly generated problem instances and solved using commercial optimization software GAMS. The computational experiments indicate the efficiency of proposed model. The concluding remarks of Chapter two can be outlined as follows.

- The proposed model solves the multi-degree cyclic flow shop robotic cells scheduling problem with multiple single gripper robots optimally.
- The proposed model finds the optimum degree of *k*DCFS-RCS problem while maximizing the throughput rate, which differentiates it from past research involved in cyclic robotic cell scheduling problems.
- Different number of robots affects the optimal *k*-degree and the cycle time for the *k*DCFS-RCS problem with time window processing times.
- The study shows that the processing operations of a part might be continued throughout three or more consecutive cycles, which is also different from the similar studies conducted.

- The provided data as processing times and the difference between lower and upper bounds of the time windows for processing times remarkably influence the solution time for the *k*DCFS-RCS problem.

$RF_{s,r}|(\text{free}, \text{C}, \text{Cyclic})|\mu$

Chapter 4 studies the multi-degree cyclic flow shop robotic cell (*k*DCFS-RCS) scheduling problem with multiple robots and free-pickup criterion. All the parts are processed successively through the machines with standard processing times and single gripper robots perform the transportation operations of parts between the machines. Due to the special characteristics of the considered problem, a metaheuristic algorithm based on ant colony optimization (MB-ACO) has been proposed. The proposed algorithm simultaneously determines the optimal degree of the cyclic schedule, the robot assignments for the transportation operations, and the optimal sequencing of the robots' moves, which in return maximize the throughput rate. The efficiency of the proposed metaheuristic algorithm is examined by a computational study on a set of randomly generated problem instances. The concluding remarks of this study can be outlined as follows.

- The proposed MB-ACO algorithm solves the multi-degree cyclic flow shop robotic cells scheduling problem with multiple single gripper robots efficiently.
- The proposed MB-ACO algorithm finds the appropriate cycle degree of the *k*DCFS-RCS problem while maximizing the throughput rate, which differentiates it from the past research involved in cyclic robotic cell scheduling problems.
- Different number of robots affects the optimal *k*-degree and the cycle time for the *k*DCFS-RCS problem.
- The study shows that the processing operations of a part might be continued throughout three or more consecutive cycles, which is also different from most of the similar studies conducted for the *k*-degree parameter.
- The study indicates the impact of occupied machines at the start time of the cycle and their related occupying part classes. The number of occupied machines determines the number of cycles that a specific instance of a part class will stay in

the system from start to finish, and therefore influences the generated solutions for the *k*DCFS-RCS problem.

$RJ_{s,r}|(\text{free}, \text{C}, \text{MP}, \text{Cyclic})|\mu$

Chapter 5 studies the cyclic job shop robotic cell scheduling (CJS-RCS) problem with multiple robots. All the jobs are processed in order of their operations on multiple machines with standard processing times, and the single gripper robots perform the transportation operations of jobs between the machines. Due to the special characteristics of the considered problem and NP-hard computational complexity, a metaheuristic algorithm based on ant colony optimization (MB-ACO) has been proposed. The proposed algorithm simultaneously determines the optimal height of jobs in the cyclic schedule, the robot assignments for transportation operations, and the optimal sequencing of the robots moves, which in return maximize the throughput rate. The efficiency of the proposed model is examined by a computational study on a set of randomly generated problem instances. The concluding remarks of this study can be outlined as follows:

- The proposed MB-ACO algorithm solves the CJS-RCS problem with multiple single gripper robots efficiently.
- The proposed MB-ACO algorithm finds the appropriate cycle height of the CJS-RCS problem while maximizing the throughput rate, which differentiates it from the past researches that considered the cycle height as constant.
- The number of robots affects the optimal height value of the cycle and the cycle time for the CJS-RCS problem.
- The study shows that the processing operations of jobs might continue throughout three or more consecutive cycles, which is also different from most of the similar studies that consider the height parameter as either one or two.
- The study indicates the impact of occupied machines at the start time of the cycle and the occupying jobs. The number of occupied machines determines the number of cycles that a specific instance of a job stays in the system from start to finish, and therefore it influences the generated solutions for the CJS-RCS problem.

As a scope for future researches, stochastic processing times can be assumed for the investigated problems in order to increase the applicability of the proposed approaches. Also, instead of using single-gripper robots, dual-gripper robots can be considered to perform the transportation operations. Furthermore, the no-wait pickup criterion can be addressed and the travel time metric might also be extended to *Euclidean* for the considered problems. The remarks of future researches can be outlined as follows:

- $RF_s^2(m_1, \dots, m_s)|(\text{Euclidean}, \text{A}, \text{MP}, \text{non} - \text{cyclic})|Cmax$
- $RF_{s,r}^2(m_1, \dots, m_s)\big|(\text{free}, \text{A}, \text{MP}, \text{non} - \text{cyclic})|Cmax$
- $RF_{s,r}^2\big|(\text{interval}, \text{C}, \text{Cyclic})|\mu$
- $\left\{RF_{s,r}^2\big|(\text{free}, \text{C}, \text{Cyclic})|\mu\right\}$; Stochastic processing times
- $\left\{RJ_{s,r}^2\big|(\text{free}, \text{C}, \text{MP}, \text{Cyclic})|\mu\right\}$; Stochastic processing times

In addition, the sequence-dependant setup times of parts or jobs on processing machines can be included in the transportation operations which are performed by the robots. Thereupon, the transportation robots can be considered as transporting servers between the machines in manufacturing cells.

## REFERENCES


Agnetis, A. (2000). Scheduling no-wait robotic cells with two and three machines. *European Journal of Operational Research, 123*, 303-314.

Agnetis, A. & Pacciarelli, D. (2000). Part sequencing in three-machine no-wait robotic cells. *Operations Research Letters, 27*, 185-192.

Alaykýran, K., Engin, O. & Döyen, A. (2007). Using ant colony optimization to solve hybrid flow shop scheduling problems. *The International Journal of Advanced Manufacturing Technology, 35*, 541-550.

Alcaide, D., Chu, C., Kats, V., Levner, E. & Sierksma, G. (2007). Cyclic multiple-robot scheduling with time-window constraints using a critical path approach. *European Journal of Operational Research, 177*, 147-162.

Aneja, Y. P. & Kamoun, H. (1999). Scheduling of parts and robot activities in a two machine robotic cell. *Computers & Operations Research, 26*, 297-312.

Atashpaz-Gargari, E. & Lucas, C. (2007). Imperialist competitive algorithm: an algorithm for optimization inspired by imperialistic competition. In *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on* (4661-4667): IEEE.

Batur, G. D., Karasan, O. E. & Akturk, M. S. (2012). Multiple part-type scheduling in flexible robotic cells. *International Journal of Production Economics, 135*, 726-740.

Bilge, Ü. & Ulusoy, G. (1995). A time window approach to simultaneous scheduling of machines and material handling system in an FMS. *Operations Research, 43*, 1058-1070.

Bostel, N. & Dejax, P. (2006). An integrated approach for modeling and solving the scheduling problem of container handling systems. *Journal of Zhejiang University Science A, 7*, 234-239.

Brucker, P., Burke, E. K. & Groenemeyer, S. (2012a). A branch and bound algorithm for the cyclic job-shop problem with transportation. *Computers & Operations Research, 39*, 3200-3214.

Brucker, P., Burke, E. K. & Groenemeyer, S. (2012b). A mixed integer programming model for the cyclic job-shop problem with transportation. *Discrete Applied Mathematics, 160*, 1924-1935.

Brucker, P. & Kampmeyer, T. (2005). Tabu search algorithms for cyclic machine scheduling problems. *Journal of Scheduling, 8*, 303-322.

Brucker, P. & Kampmeyer, T. (2008a). Cyclic job shop scheduling problems with blocking. *Annals of Operations Research, 159*, 161-181.

Brucker, P. & Kampmeyer, T. (2008b). A general model for cyclic machine scheduling problems. *Discrete Applied Mathematics, 156*, 2561-2572.

Carlier, J., Haouari, M., Kharbeche, M. & Moukrim, A. (2010). An optimization-based heuristic for the robotic cell problem. *European Journal of Operational Research, 202*, 636-645.

Cavory, G., Dupas, R. & Goncalves, G. (2005). A genetic approach to solving the problem of cyclic job shop scheduling with linear constraints. *European Journal of Operational Research, 161*, 73-85.

Che, A., Chabrol, M., Gourgand, M. & Wang, Y. (2012). Scheduling multiple robots in a no-wait re-entrant robotic flowshop. *International Journal of Production Economics, 135*, 199-208.

Che, A. & Chu, C. (2005). Multi-degree cyclic scheduling of two robots in a no-wait flowshop. *Automation Science and Engineering, IEEE Transactions on, 2*, 173-183.

Che, A. & Chu, C. (2009). Multi-degree cyclic scheduling of a no-wait robotic cell with multiple robots. *European Journal of Operational Research, 199*, 77-88.

Che, A., Chu, C. & Chu, F. (2002). Multicyclic hoist scheduling with constant processing times. *Robotics and Automation, IEEE Transactions on, 18*, 69-80.

Che, A., Hu, H., Chabrol, M. & Gourgand, M. (2011). A polynomial algorithm for multi-robot 2-cyclic scheduling in a no-wait robotic cell. *Computers & Operations Research, 38*, 1275-1285.

Chen, H., Chu, C. & Proth, J.-M. (1997). Sequencing of parts in robotic cells. *International Journal of Flexible Manufacturing Systems, 9*, 81-104.

Chen, H., Chu, C. & Proth, J.-M. (1998). Cyclic scheduling of a hoist with time window constraints. *Robotics and Automation, IEEE Transactions on, 14*, 144-152.

Chen, L., Bostel, N., Dejax, P., Cai, J. & Xi, L. (2007). A tabu search algorithm for the integrated scheduling problem of container handling systems in a maritime terminal. *European Journal of Operational Research, 181*, 40-58.

Crama, Y., Kats, V., Van de Klundert, J. & Levner, E. (2000). Cyclic scheduling in robotic flowshops. *Annals of Operations Research, 96*, 97-124.

Crama, Y. & Van De Klundert, J. (1997). Cyclic scheduling of identical parts in a robotic cell. *Operations Research, 45*, 952-965.

Dawande, M., Geismar, H. N., Sethi, S. P. & Sriskandarajah, C. (2005). Sequencing and scheduling in robotic cells: Recent developments. *Journal of Scheduling, 8*, 387-426.

Dawande, M., Sriskandarajah, C. & Sethi, S. (2002). On throughput maximization in constant travel-time robotic cells. *Manufacturing and Service Operations Management, 4*, 296-312.

Dawande, M. W., Geismar, H. N., Sethi, S. P. & Sriskandarajah, C. (2007). *Throughput optimization in robotic cells* (Volume 101): Springer.

Dorigo, M. & Gambardella, L. M. (1997). Ant colony system: a cooperative learning approach to the traveling salesman problem. *Evolutionary Computation, IEEE Transactions on, 1*, 53-66.

Elmi, A., Solimanpur, M., Topaloglu, S. & Elmi, A. (2011). A simulated annealing algorithm for the job shop cell scheduling problem with intercellular moves and reentrant parts. *Computers & Industrial Engineering, 61*, 171-178.

Fazel Zarandi, M., Mosadegh, H. & Fattahi, M. (2013). Two-machine robotic cell scheduling problem with sequence-dependent setup times. *Computers & Operations Research, 40*, 1420-1434.

Foumani, M. & Jenab, K. (2013). Analysis of flexible robotic cells with improved pure cycle. *International Journal of Computer Integrated Manufacturing, 26*, 201-215.

Geismar, H. N., Dawande, M. & Sriskandarajah, C. (2004). Robotic cells with parallel machines: Throughput maximization in constant travel-time cells. *Journal of Scheduling, 7*, 375-395.

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*, 311-328.

Geismar, H. N., Pinedo, M. & Sriskandarajah, C. (2008). Robotic cells with parallel machines and multiple dual gripper robots: a comparative overview. *IIE Transactions, 40*, 1211-1227.

Glover, F. & Laguna, M. (1999). *Tabu search*: Springer.

Goldberg, D. E. & Holland, J. H. (1988). Genetic algorithms and machine learning. *Machine Learning, 3*, 95-99.

Grabowski, J. & Wodecki, M. (2004). A very fast tabu search algorithm for the permutation flow shop problem with makespan criterion. *Computers & Operations Research, 31*, 1891-1909.

Hall, N. G. (2007). *Operations research techniques for robotics systems*: Handbook of Industrial Robotics (2nd Edition).

Hall, N. G., Kamoun, H. & Sriskandarajah, C. (1998). Scheduling in robotic cells: Complexity and steady state analysis. *European Journal of Operational Research, 109*, 43-65.

Hall, N. G. & Sriskandarajah, C. (1996). A survey of machine scheduling problems with blocking and no-wait in process. *Operations Research, 44*, 510-525.

Hanen, C. (1994). Study of a NP-hard cyclic scheduling problem: The recurrent job-shop. *European Journal of Operational Research, 72*, 82-101.

Holland, J. H. (1975). *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence*: U Michigan Press.

Hurink, J. & Knust, S. (2001). Makespan minimization for flow-shop problems with transportation times and a single robot. *Discrete Applied Mathematics, 112*, 199-216.

Hurink, J. & Knust, S. (2002). A tabu search algorithm for scheduling a single robot in a job-shop environment. *Discrete Applied Mathematics, 119*, 181-203.

Hurink, J. & Knust, S. (2005). Tabu search algorithms for job-shop problems with a single transport robot. *European Journal of Operational Research, 162*, 99-111.

Jalilvand-Nejad, A. & Fattahi, P. (2013). A mathematical model and genetic algorithm to cyclic flexible job shop scheduling problem. *Journal of Intelligent Manufacturing*, 1-14.

Jamili, A., Shafia, M. A. & Tavakkoli-Moghaddam, R. (2011). A hybrid algorithm based on particle swarm optimization and simulated annealing for a periodic job shop scheduling problem. *The International Journal of Advanced Manufacturing Technology, 54*, 309-322.

Jiang, Y. & Liu, J. (2007). Multihoist cyclic scheduling with fixed processing and transfer times. *Automation Science and Engineering, IEEE Transactions on, 4*, 435-450.

Jiang, Y. & Liu, J. (2014). A new model and an efficient branch-and-bound solution for cyclic multi-hoist scheduling. *IIE Transactions, 46*, 249-262.

Jin, Z., Ohno, K., Ito, T. & Elmaghraby, S. (2002). Scheduling hybrid flow shops in printed circuit board assembly lines. *Production and Operations Management, 11*, 216-230.

Jin, Z., Yang, Z. & Ito, T. (2006). Metaheuristic algorithms for the multistage hybrid flowshop scheduling problem. *International Journal of Production Economics, 100*, 322-334.

Karzanov, A. & Livshits, E. (1978). Minimal quantity of operators for serving a homogeneous linear technological process. *Automation and Remote Control, 39*, 445-450.

Kats, V. & Levner, E. (2009). A polynomial algorithm for 2-cyclic robotic scheduling: A non-Euclidean case. *Discrete Applied Mathematics, 157*, 339-355.

Kats, V., Levner, E. & Meyzin, L. (1999). Multiple-part cyclic hoist scheduling using a sieve method. *Robotics and Automation, IEEE Transactions on, 15*, 704-713.

Kechadi, M., Low, K. S. & Goncalves, G. (2013). Recurrent neural network approach for cyclic job shop scheduling problem. *Journal of Manufacturing Systems, 32*, 689-699.

Kharbeche, M., Carlier, J., Haouari, M. & Moukrim, A. (2010). Exact method for robotic cell problem. *Electronic Notes in Discrete Mathematics, 36*, 859-866.

Kirkpatrick, S. & Vecchi, M. (1983). Optimization by simmulated annealing. *Science, 220*, 671-680.

Leung, J. M. & Levner, E. (2006). An efficient algorithm for multi-hoist cyclic scheduling with fixed processing times. *Operations Research Letters, 34*, 465-472.

Leung, J. M., Zhang, G., Yang, X., Mak, R. & Lam, K. (2004). Optimal cyclic multi-hoist scheduling: a mixed integer programming approach. *Operations Research, 52*, 965-976.

Levner, E., Kats, V. & Levit, V. E. (1997). An improved algorithm for cyclic flowshop scheduling in a robotic cell. *European Journal of Operational Research, 97*, 500-508.

Liu, B., Wang, L. & Jin, Y.-H. (2008). An effective hybrid PSO-based algorithm for flow shop scheduling with limited buffers. *Computers & Operations Research, 35*, 2791-2806.

Liu, J. & Jiang, Y. (2005). An efficient optimal solution to the two-hoist no-wait cyclic scheduling problem. *Operations Research, 53*, 313-327.

Logendran, R. & Sriskandarajah, C. (1996). Sequencing of robot activities and parts in two-machine robotic cells. *International Journal of Production Research, 34*, 3447-3463.

Lundy, M. & Mees, A. (1986). Convergence of an annealing algorithm. *Mathematical Programming, 34*, 111-124.

Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A. & Teller, E. (1953). Simulated annealing. *Journal of Chemical Physics, 21*, 1087-1092.

Michalewicz, Z. (1992). *Genetic Algorithm*. In: Springer-Verlarg.

Nawaz, M., Enscore Jr, E. E. & Ham, I. (1983). A heuristic algorithm for the< i> m</i>-machine,< i> n</i>-job flow-shop sequencing problem. *Omega, 11*, 91-95.

Neil Geismar, H., Dawande, M. & Sriskandarajah, C. (2005). Approximation algorithms for< i> k</i>-unit cyclic solutions in robotic cells. *European Journal of Operational Research, 162*, 291-309.

Nowicki, E. & Smutnicki, C. (2005). An advanced tabu search algorithm for the job shop problem. *Journal of Scheduling, 8*, 145-159.

Phadke, M. S. (1989). *Quality engineering using robust design*, New Jersey: Practice Hall Englewood Cliffs.

Pinedo, M. (2002). *Scheduling: theory, algorithms, and systems*: Springer.

Ribas, I., Leisten, R. & Framiñan, J. M. (2010). Review and classification of hybrid flow shop scheduling problems from a production system and a solutions procedure perspective. *Computers & Operations Research, 37*, 1439-1454.

Ruiz, R. & Vázquez-Rodríguez, J. A. (2010). The hybrid flow shop scheduling problem. *European Journal of Operational Research, 205*, 1-18.

Sarkheil, N. & Jenab, K. (2012). On the optimality of 1-unit cycles in flow-shop networks. *International Journal of Production Research, 50*, 2705-2709.

Sethi, S. P., Sriskandarajah, C., Sorger, G., Blazewicz, J. & Kubiak, W. (1992). Sequencing of parts and robot moves in a robotic cell. *International Journal of Flexible Manufacturing Systems, 4*, 331-358.

Shafiei-Monfared, S., Salehi-Gilani, K. & Jenab, K. (2009). Productivity analysis in a robotic cell. *International Journal of Production Research, 47*, 6651-6662.

Smutnicki, C. (2012). An efficient algorithm for finding minimal cycle time in cyclic job shop scheduling problem. *Intelligent Engineering Systems (INES), 2012 IEEE 16th International Conference on* (381-386): IEEE.

Solimanpur, M. & Foroughi, A. (2011). A new approach to the cell formation problem with alternative processing routes and operation sequence. *International Journal of Production Research, 49*, 5833-5849.

Solimanpur, M., Vrat, P. & Shankar, R. (2004). A neuro-tabu search heuristic for the flow shop scheduling problem. *Computers & Operations Research, 31*, 2151-2164.

Song, J.-S. & Lee, T.-E. (1998). Petri net modeling and scheduling for cyclic job shops with blocking. *Computers & Industrial Engineering, 34*, 281-295.

Soukhal, A. & Martineau, P. (2005). Resolution of a scheduling problem in a flowshop robotic cell. *European Journal of Operational Research, 161*, 62-72.

Soukhal, A., Oulamara, A. & Martineau, P. (2005). Complexity of flow shop scheduling problems with transportation constraints. *European Journal of Operational Research, 161*, 32-41.

Sriskandarajah, C., Hall, N. G. & Kamoun, H. (1998). Scheduling large robotic cells without buffers. *Annals of Operations Research, 76*, 287-321.

Tsubone, H., Ohba, M., Takamuki, H. & Miyake, Y. (1993). A production scheduling system for a hybrid flow shop-A case study. *Omega, 21*, 205-214.

Zahrouni, W. & Kamoun, H. (2012). Sequencing and scheduling in a three-machine robotic cell. *International Journal of Production Research, 50*, 2823-2835.

Zhang, Q., Manier, H. & Manier, M.-A. (2012). A genetic algorithm with tabu search procedure for flexible job shop scheduling with transportation constraints and bounded processing times. *Computers & Operations Research, 39*, 1713-1723.

Zhou, Z., Che, A. & Yan, P. (2012). A mixed integer programming approach for multi-cyclic robotic flowshop scheduling with time window constraints. *Applied Mathematical Modelling, 36*, 3621-3629.