SCHEDULING IN FLEXIBLE ROBOTIC MANUFACTURING CELLS

A DISSERTATION

SUBMITTED TO THE DEPARTMENT OF INDUSTRIAL ENGINEERING

AND THE INSTITUTE OF ENGINEERING AND SCIENCES

OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

By

Hakan Gültekin

September, 2006

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of doctor of philosophy.

Prof. M. Selim Aktürk(Principal Advisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of doctor of philosophy.

Assoc. Prof. Oya Ekin Karaşan

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of doctor of philosophy.

Prof. Sinan Kayalıgil

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of doctor of philosophy.

Prof. Billur Barshan

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of doctor of philosophy.

Asst. Prof. Emre Alper Yıldırım

Approved for the Institute of Engineering and Sciences:

Prof. Mehmet Baray
Director of Institute of Engineering and Sciences

ABSTRACT

SCHEDULING IN FLEXIBLE ROBOTIC MANUFACTURING CELLS

Hakan Gültekin
Ph.D. in Industrial Engineering
Supervisor: Prof. M. Selim Aktürk
September, 2006

The focus of this thesis is the scheduling problems arising in robotic cells which consist of a number of machines and a material handling robot. The machines used in such systems for metal cutting industries are highly flexible CNC machines. Although flexibility is the key term that affects the performance of these systems, the current literature ignores this. As a consequence, the problems considered in the current literature are either too limiting or the provided solutions are suboptimal for the flexible systems. This thesis analyzes different robotic cell configurations with different sources of flexibility. This study is the first one to consider operation allocation problems and controllable processing times as well as some design problems and bicriteria models in the context of robotic cell scheduling. Also, a new class of robot move cycles is defined, which is overlooked in the existing literature. Optimal solutions are provided for solvable cases, whereas complexity analyses and efficient heuristic algorithms are provided for the remaining problems.

Key words: Robotic cell scheduling, Flexible manufacturing cells, CNC, Controllable processing times, Bicriteria scheduling

ÖZET

ROBOTLU ESNEK ÜRETİM HÜCRELERİNDE ÇİZELGELEME

Hakan Gültekin
Endüstri Mühendisliği Bölümü Doktora
Tez Yöneticisi: Prof. Dr. M. Selim Aktürk
Eylül, 2006

Bu tezin konusu belirli sayıda makinadan ve bunlara malzeme taşıyan bir robottan oluşan robotik hücrelerde ortaya çıkan çizelgeleme problemleridir. Metal işleme endüstrisinde bu tür hücrelerde esnekliği sağlamak için CNC makinaları kullanılmaktadır. Bu tür sistemler için esneklik, sistemin performansını etkileyen temel unsurlardan olmasına rağmen, literatürde gözardı edilmiştir. Bunun sonucunda, literatürde ele alınan problemler ya çok kısıtlı kullanım alanları içindir ya da elde edilen sonuçlar esnek sistemler için alteniyidir. Bu tez değişik hücre konfigürasyonlarını, değişik esneklik kaynaklarının varlığında incelemektedir. Operasyon atama problemleri ve kontrol edilebilir işlem zamanlarının yanında çeşitli dizayn ve iki kriterli eniyileme modelleri de robotik hücre çizelgeleme problemleri bünyesinde ilk defa bu çalışmada ele alınmıştır. Bunların yanında, literatürde gözden kaçan yeni bir robot hareket döngüsü türü de ilk defa bu çalışmada tanımlanmıştır. Çözülebilir problem türleri için eniyi çözümler sağlanırken geri kalanlar için karmaşıklık analizleri yapılmış ve etkin sezgisel algoritmalar geliştirilmiştir.

Anahtar sözcükler: Robotik hücre çizelgelemesi, Esnek üretim hücreleri, CNC, Kontrol edilebilir üretim zamanları, İki kriterli çizelgeleme

To my family...

ACKNOWLEDGEMENT

First and foremost, I would like to express my gratitude to my advisor, Prof. M. Selim Aktürk, for his invaluable helps, guidance, encouragement and support during my Ph.D. as well as my M.S. studies. He has been ready to provide help, support and advice not only in academic issues but in all aspects of life, whenever I need. Without his supervision and guidance, I would not be able to manage all those. I will always need his advices throughout my entire life.

I am grateful to Assoc. Prof. Oya Ekin Karaşan for working with us during this six years of time starting from my M.S. studies. I have made significant progress in my thesis with her invaluable guidance, remarks and recommendations. Her understanding and sympathy was a great encouragement for me.

I am indebted to Prof. Sinan Kayalıgil for reading my progress reports, listening my presentations and providing valuable suggestions as a member of my Ph.D. committee.

I would like to thank Assist. Prof. Emre Alper Yıldırım and Prof. Billur Barshan for showing keen interest to the subject matter and accepting to read and review this thesis and for their suggestions.

I would like to express my appreciation to all my friends who have contributed directly or indirectly to this dissertation. I am grateful to Selçuk Gören and Sinan Gürel for their friendship, helps and academic and most importantly morale support. I would like to thank Fatih Safa Erenay and Mehmet Mustafa Tanrıkulu (memuta) for their friendship. I am grateful for their support.

I would like to thank my family for their endless love and support.

Last but not the least, I would like to express my deepest gratitude and love to my wife, Feyza Gültekin and my children, Ayşe Melek Gültekin and Abdulkadir Gültekin for everything they brought to my life.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The search for better ways to manufacture components has increased the level of automation in manufacturing industries. This trend involves the use of computer controlled machines and automated material handling devices. One of the widespread applications of automation is the installation and use of robotic cells. A manufacturing cell consisting of a number of machines and a material handling robot is called a robotic cell. These kinds of robots are used extensively in chemical, electronic and metal cutting industries. Robots are installed in order to reduce labor cost, to increase output, to provide a more flexible production system and to replace people working in dangerous or hazardous conditions [13]. However, in order to use such systems efficiently some important problems must be tackled. Among these, the design of the cells and the scheduling of robot moves are eminent.

In this thesis we will consider a flexible robotic manufacturing cell which consists of a number of Computer Numerically Controlled (CNC) machines and a material handling robot. "Flexibility" plays a crucial role in such cells. There are many different types of flexibilities such as operational flexibility,

process flexibility, routing flexibility, material handling flexibility, and machine flexibility [18]. In this thesis we will consider those types of flexibilities that affect the processing times of the parts on the machines. More specifically, we will consider the operational and process flexibilities. *Operational flexibility* is defined as the capability of changing the ordering of several operations where *process flexibility* is defined as the capability of performing several operations at the same machine. Such flexibilities are achieved by considering alternative tool types for operations and loading multiple tools to the tool magazines of the machines.

We will investigate the productivity gain attained by the additional flexibility introduced by the CNCs. The aim is the maximization of the throughput of the cell or equivalently minimization of the cycle time which is defined as the long run average time required by the robotic cell to complete one part. More formally, we assume that we have infinite number of parts and if C_n denotes the completion time of the n^{th} part then the long run average cycle time is $\limsup_{n\to\infty} C_n/n$ [21]. Cyclic production in a robotic cell refers to the production of finished parts by repeating a fixed sequence of robot moves. As discussed in Geismar et al. [30], the main motivation for studying cyclic production comes from practice: cyclic schedules are easy to implement and control and are the primary ways of specifying the operation of a robotic cell in industry. Furthermore, Dawande et al. [23] show that for the problem of scheduling operations in bufferless robotic cells that produce identical parts (similar to our problem), it is sufficient to consider cyclic schedules in order to maximize throughput. They prove that there is at least one cyclic schedule in the set of all schedules that optimizes the throughput of the cell.

After reviewing the relevant literature in Chapter 2 we formulate our problem and present the necessary definitions and notation in Chapter 3. The focus of Chapter 4 is an m-machine robotic cell in which the machines are

assumed to be capable of performing all the required operations of each part. As a consequence of this assumption, we relax the flowshop assumption which is used in the current literature and which unnecessarily limits the number of alternatives. The flexibility of the CNC machines leads to the definition of a new class of robot move cycles. We select and focus on one of the cycles among this class which is widely used in industry not because it is proved to be optimal but because it is simple and practical. The regions of optimality for this cycle for the m-machine case is determined. We analyze the 2- and 3-machine cases further in detail. For the regions where the proposed cycle may not be optimal, we present a worst case performance bound of using this cycle.

Till now the research on robotic cell scheduling problems concentrated on the operational aspects such as finding the part input sequence and the robot move sequence. However, the design of the cells also affects the performance of such cells. In Chapter 5, we study some design problems arising in robotic manufacturing cells. We first consider the layout of the machines and show that the efficiency of the cells can be increased by changing the layout of the machines. As a second design problem we consider the number of machines as a decision variable. We determine the optimal number of machines that minimizes the cycle time for given parameters such as the robot transportation time, load/unload time and the processing times of the operations.

Assuming that the CNC machines are capable of performing all the required operations may be unrealistic at times since the tool magazines have limited capacity and in many practical applications the required number of tools exceeds this capacity; ultimately, duplicating all the tools may not be economically justifiable. In this respect, in Chapter 6 we consider a 2-machine robotic cell and assume that some operations can only be processed on the first machine while some others can only be processed on the second machine

due to tooling constraints. As a consequence, the system is assumed to be a flowshop in which each part passes through all machines in the same sequence; $1, 2, \ldots, m$. The remaining operations can be processed on either machine. The problem is to find the allocation of the remaining operations to the machines and the optimal robot move cycle that jointly minimize the cycle time. We prove that the optimal solution is either a 1-unit or a 2-unit robot move cycle, where an n-unit cycle is defined to be a cycle in which all machines are loaded and unloaded exactly n times and the initial and the final states of the system (position of the robot and the status of each machine) are the same. We present the regions of optimality for all 1-unit and 2-unit robot move cycles. Finally, a sensitivity analysis on the results is conducted.

Processing times of the parts on the machines can be changed by altering the machining conditions such as the speed and the feed rate for highly flexible CNC machines and this affects the cycle time. On the other hand, altering the machining conditions also affects the manufacturing cost. As a result, in Chapters 7 and 8 we develop and solve a bicriteria problem formulation for the robotic cell scheduling problem. In Chapter 7, we consider 2- and 3-machine robotic cells. The cell is assumed to be a flowshop in which each part has one specific operation on each machine and follows the same sequence of machines. The aim is to find the robot move sequence as well as the processing times of the parts on each machine that not only minimizes the cycle time but, for the first time in robotic cell scheduling literature, also minimizes the manufacturing cost. For each 1-unit cycle in 2- and 3-machine cells, we determine the efficient set of processing time vectors such that no other processing time vector gives both a smaller cycle time and a smaller cost value. We also compare these cycles with each other to determine the sufficient conditions under which each of the cycles dominates the rest. Finally, we show how different assumptions on cost structures affect the results. On the other hand, in Chapter 8, besides

determining the robot move sequence and the processing times of the operations on the machines, we determine the allocation of the operations to the machines. Since finding the allocation of the operations is NP-Hard itself, we develop a heuristic algorithm which approximates a set of points on the efficient frontier. An experimental framework is designed in order to evaluate the efficiency of the algorithm and the results are compared with a commercial nonlinear mixed integer program solver software GAMS-DICOPT2x-C.

Chapter 2

Literature Review

In this section we will review the relevant literature pertinent to this study. However, let us first give some necessary notation and definitions that will be used throughout this study. The following definitions are borrowed from [19].

Definition 2.1 A_i is the robot activity defined as; robot unloads machine i, transfers part from machine i to machine $i+1$, loads machine $i+1$.

Definition 2.2 An n-unit robot move cycle is the robot move cycle in which starting with an initial state of the system, the robot performs each activity exactly n times and ends up with the initial state of the system. Note that, in an n-unit robot move cycle exactly n parts are produced.

In an m-machine robotic cell we have exactly $m+1$ robot activities: A_0, A_1, ..., A_m, where the machines are numbered as $1, 2, \ldots, m$, the input buffer is numbered as 0 and the output buffer is numbered as $m+1$. Since in an optimal cycle we require that the robot move path is as short as possible, any two consecutive activities uniquely determine the robot moves between them.

Therefore, any robot move cycle can be uniquely described by a permutation of the above activities. Additionally, Crama et al. [21] make the following basic feasibility assumptions which we shall incorporate in our study as well:

1- Robot cannot load an already loaded machine.

2- Robot cannot unload an already unloaded machine.

These assumptions restrict the ordering of the activities. For example, let Si^m represent a specific robot move cycle in an m-machine robotic cell. Then for two machines we have only two feasible 1-unit robot move cycles:

$$S_1^2 = A_0A_1A_2, \quad S_2^2 = A_0A_2A_1.$$

In a 3-machine cell there are six feasible 1-unit cycles which can be listed as follows:

$$S_1^3 = (A_0A_1A_2A_3), \quad S_2^3 = (A_0A_2A_1A_3), \quad S_3^3 = (A_0A_1A_3A_2),$$
$$S_4^3 = (A_0A_3A_1A_2), \quad S_5^3 = (A_0A_2A_3A_1), \quad S_6^3 = (A_0A_3A_2A_1).$$

The animated views of these robot move cycles can be found at the web site http://www.ie.bilkent.edu.tr/∼robot. Now, let us calculate the cycle time of S_2^2 for a 2-machine cell producing identical parts as an example. Let P_i represent the processing time of each of the identical parts on machine i and w_i represent the waiting time of the robot in front of machine i. Let T_S represent the cycle time of the robot move cycle S, i.e., the long run average time to produce one part under robot move cycle S. Furthermore, let δ represent the robot transportation time between any two consecutive machines and ϵ represent the loading/unloading time of the machines. In this cycle, initially

the first machine is idle and the second machine is loaded. The robot is in front of the input buffer just before taking a part. The robot takes a part from the input buffer (ϵ), transports it to the first machine (δ), loads it (ϵ), travels to the second machine (δ), waits in front of the machine to finish processing of the part (w_2), unloads it (ϵ), transports the part to output buffer (δ), drops the part (ϵ), travels back to the first machine (2δ), waits in front of the machine (w_1), unloads it (ϵ), transports the part to the second machine (δ), loads it (ϵ), travels back to input buffer (2δ). The initial and the final states are the same thus the cycle is completed. Then the cycle time is the following:

$$T_{S_2^2} = 6\epsilon + 8\delta + w_1 + w_2.$$

Note that, when the robot arrives in front of a machine to unload it, if the processing of the part is already completed then the robot unloads the machine immediately without any waiting time. Otherwise, the waiting time is equivalent to the remaining processing time. As a consequence, the waiting times are $w_1 = max\{0, P_1 - 2\epsilon - 4\delta - w_2\}$ and $w_2 = max\{0, P_2 - 2\epsilon - 4\delta\}$. After some simple arithmetic operations, the cycle time is found as follows:

$$T_{S_2^2} = 6\epsilon + 8\delta + max\{0, P_1 - 2\epsilon - 4\delta, P_2 - 2\epsilon - 4\delta\}.$$

Now let us consider the basic assumptions that are common for most of the studies.

- All data are deterministic.
- The robot and the processing machines never experience breakdown and never require maintenance. Setup times are assumed to be negligible.
- No preemption is allowed in the processing of any operation.

- Parts are always available at the input buffer and there is always an empty place at the output buffer.

In the next section we will list the differences in robotic cell scheduling problems and present the standard classification scheme for those problems. In Section 2.2, we will present basic results from the previous studies on robotic cell scheduling problems. In Section 2.3, the multicriteria scheduling models considered in FMS scheduling literature are explained. The studies considering controllable processing times in scheduling are summarized in Section 2.4.

2.1 Problem Types and Classification Scheme

The robotic cell scheduling problems differ from each other in the following aspects:

1. Cell types,
2. Processing times,
3. Objective functions,
4. Robot travel times,
5. Loading and unloading times,
6. Number of machines and parts.

We will analyze each of them in detail in the following sections.

2.1.1 Cell types

In most general terms, a robotic cell consists of m machines denoted as $M_i, i = 1, 2, \ldots, m$. Also there is an input and an output buffer denoted as (M_0) and (M_{m+1}) respectively. In some implementations, the input device and the output device are at the same location, and this unit is called a *load lock* [30]. In most studies there is one robot that makes the loading/unloading of the machines and the transportation of the parts between these machines. Some studies also consider the multiple robots case.

An important characteristic of the robotic cells is the buffers in front of the machines. In the literature, robotic cells with finite or infinite buffers and no buffers are considered. In general, the additional freedom introduced by buffers tends to complicate scheduling problems [82]. The focus of this study is on robotic cells with no buffers. For the complexity of the robotic cells with buffers we refer to Hurink and Knust [51]. Other problems and approaches to this subject can be found in Kise [65], Hitomi and Yashimura [49], King et al. [64], Finke et al. [27], Levner [71], and Kogan and Levner [66].

For the bufferless problems, all parts must be either on the input buffer, on one of the machines, on the output buffer, or on the robot. This is equivalent to *blocking* condition in a classical flowshop: a part that has completed processing on M_i can not leave unless machine M_{i+1} is unoccupied [83]. This should not be confused with the more restrictive *no-wait* condition in which a part must be removed from a machine and transferred to the next one as soon as that first machine completes processing that part, a condition which will be analyzed in Section 2.1.2.

Some researchers consider cells with dual gripper robots instead of single gripper robots. Dual gripper robots can hold two parts at a time and unload

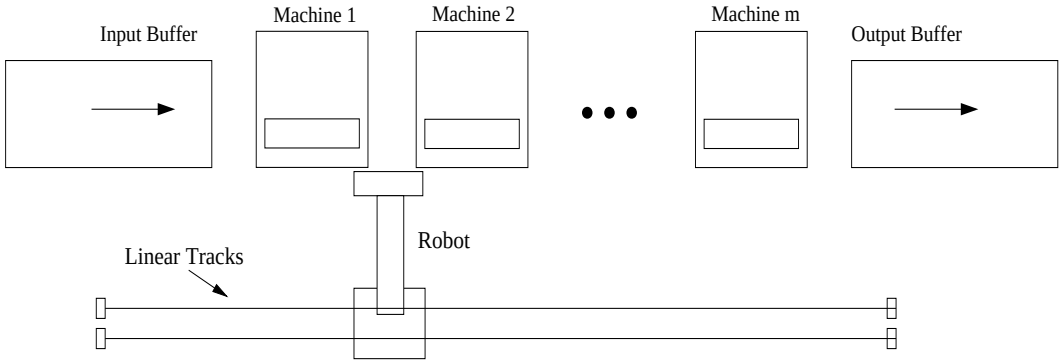

Figure 2.1: Inline robotic cell layout

and load a machine simultaneously. This increases the number of feasible robot move cycles drastically. As a consequence, the complexity of the problem also increases. Another stream of research considers parallel machines at each stage of production. The robot makes the transportation between the stages and the loading/unloading of the parallel machines is performed by another material handling device.

Another characteristic of the robotic cells is the layout of the cells. As Han et al. [47] proposed, cell formation may increase efficiency of the cell. Three different layouts are considered for the robotic cells: robot-centered cells denoted as RCCm for an m-machine robotic cell, (where the robot movement is rotational), in-line robotic cells denoted as IRCm (where the robot moves linearly) and mobile-robot cells denoted as MRCm (generalization of in-line robotic cell and robot-centered cell) [75]. In this thesis, consistent with most of the research on this area, we will assume an in-line robotic cell layout as shown in Figure 2.1.

2.1.2 Processing times

In most general terms, the processing times are represented as $[L_i^j, U_i^j]$ which gives the processing time of part j on machine M_i. The meaning of the processing window is that the time spent by part j on machine M_i must be at least L_i^j and may not exceed U_i^j. For example, if we consider the case that each part has a precisely defined processing time on each machine and can wait on the machine indefinitely long after it has been processed, it can be handled by setting all upper bounds U_i^j to $+\infty$. This case is referred to as *unbounded processing windows* [21]. Another type of processing requirement is referred to as *with blocking,* i.e., the machine becomes blocked if the part is not removed from it after the processing is finished. Another model is referred to as *no-wait;* here it is assumed that the parts must be removed from the machines immediately after the processing is finished. These two problems can be modelled by setting $L_i^j = U_i^j$. This setting of processing windows is referred to as *zero-width processing windows* [21]. No-wait type processes are commonly seen in chemical and electronic industries, where the parts are dipped into chemical substances, after a certain amount of time taken out and in order not to become defective should immediately proceed with the next operation in sequence. Also in plastic molding and steel manufacturing, where the raw material must maintain a certain temperature, no-wait type conditions are used. Such conditions also ensure freshness in food canning industries (Hall and Sriskandarajah [46]). There is vast amount of research focusing on these types of problems.

Another problem with a different processing requirement is formulated in Akturk et al. [2] and Gultekin et al. [38] where the processing times are assumed to be decision variables. The parts are assumed to have several operations to complete their processing. Each operation has its own operation

time. Then these operations are tried to be allocated to the machines in order to minimize the cycle time. Thus, the processing times depend on the allocation of the operations.

2.1.3 Objective functions

There are two objective functions that are commonly used in robotic cell scheduling literature. The first and the most widely used one is the minimization of the cycle time or the maximization of the throughput. Since the robot follows a computer program, there must be a finite activity sequence for the robot that it repeats to produce the parts. Thus, the robot activities must be cyclic because of its nature and minimizing this cycle time is a relevant objective. Cycle time is defined as the long run average time that is required to produce one part where each robot activity is performed an equal number of times and the initial and the final states of the system are the same. The cycle time for an n-unit cycle is found by dividing the total time required to finish the cycle by n so that the average time to produce one part is found. The second objective function that is used widely is the minimization of the makespan of the schedule, which is defined as the completion time of the last job in the sequence.

2.1.4 Robot travel times

In the most general case, robot travel time between machine i and j is assigned a value δ_{ij}, $0 \leq i, j \leq m+1$. The travel times are neither additive nor constant. By additive we mean that $\delta_{ij} = \delta_{i(i+1)} + \delta_{(i+1)(i+2)} + \ldots + \delta_{(j-1)(j)}$. That is, total time to travel between machine i to machine j is the summation of the travel times between consecutive machines on the way from machine i to machine j.

Brauner et al. [11] study this problem with the following assumptions:

1. The travel time from a machine to itself is zero, i.e., $\delta_{ii} = 0, \forall i$.
2. The travel times satisfy the triangular inequality, i.e. $\delta_{ij} + \delta_{jk} \geq \delta_{ik}$, $\forall i, j, k$.
3. The travel times are symmetric, i.e. $\delta_{ij} = \delta_{ji}$, $\forall i, j$.

A robotic cell that satisfies Assumptions 1 and 2 is called a Euclidean robotic cell; one that satisfies Assumptions 1,2 and 3 is called a Euclidean symmetric robotic cell [30]. The robotic cell scheduling problem for either case is NP-hard in the strong sense [11].

In some studies, the robot travel time is assumed to be additive and constant for any transportation between any two consecutive workstations in which case $\delta_{ij} = \delta \ \ \forall i, j$.

In more realistic cases, the acceleration and the deceleration of the robot is considered [75]. In this case, the travel time between two consecutive machines does not change while on the other hand, the travel time between non-consecutive machines is reduced. For each intervening machine, the robot is assumed to save γ units of time.

For mobile robot cells, since the robot both moves linearly and rotationally, the linear movement can take more time. For a two machine cell this occurs when the robot moves between machines 1 and 2 [75]. Thus, a time denoted δ_0 is added to the travel time for the movements that include transportation between machines 1 and 2.

Dawande et al. [24] consider another transportation time model in which additivity is not applicable. They assume that the robot travel time between

any pair of machines is constant δ. This happens when the cells are compact and the robots have varying acceleration and deceleration. Thus, the travel times between any pair of machines vary in negligible amounts.

2.1.5 Loading and unloading times

Several authors consider the loading and unloading times to be machine dependent, that is, it takes ϵ_i time to load or unload a part to machine i. In other cases, this time is assumed to be constant for all workstations and parts that is, $\epsilon_i = \epsilon, \forall i$. Dawande et al. [24] state that, when comparing cycle times of different cycles, the values of ϵ_i has no effect. This is because no matter what the robot's sequence may be, each part is unloaded from input buffer, loaded and unloaded on each machine and loaded on output buffer.

2.1.6 Number of machines and parts

In a robotic cell the number of machines may differ from 2 to m. Naturally most analytical results are for the cases where the number of machines are relatively small, namely two and three. For the cases where the machine number is more than three, most of the problems appear to be NP-hard. However, when a flexible robotic cell is considered which consists of CNC machines, the number of machines is relatively small due to physical space constraints.

The number of parts considered may also differ among the existing studies in the literature. Some of the studies assume identical parts for which there is no sequencing of the parts. The only problem is to find the robot move cycle that minimizes the cycle time. On the other hand, in multiple parts case the problem is to find the robot move sequence as well as the part input sequence

that jointly minimize the cycle time. It is obvious that this problem is much more difficult than its identical parts counterpart and most of the analytical results derived for the identical parts case fail to apply to this case. Again, because of the nature of industrial robots, the multiple parts case also must follow a cycle and this cycle can use the concept of minimal part set (MPS). The MPS for a production environment can be obtained, if the forecasted demand L_j is given for each part type j over planning horizon. If d is the largest common divisor of the integers $L_1, L_2, \ldots, L_k$, the integer ratio ($r_j = L_j/d$) of part types can be represented as $r = (r_1, r_2, \ldots, r_k)$. The vector is the minimal production ratio (MPR) and the part set corresponding to this ratio is known as the MPS. For the sake of clarity consider the following example. Let us assume that we have three products A, B and C, with forecasted demand, L_j, being 100, 150 and 250 respectively. Then, the largest common divisor, d, is 50 and the MPR is $(2, 3, 5)$. As a result, the MPS is composed of 2 A's, 3 B's and 5 C's. Given an MPS of n parts, Geismar et al. [30] define an *MPS cycle* to be a sequence of robot moves in which exactly n parts of an MPS enter the cell at the input station, exactly n parts of the MPS exit the cell at the output station and the cell returns to its initial state. The order in which the parts enter the cell is called the *MPS part schedule (*or simply *part schedule).* An MPS cycle is determined by the MPS part schedule and the *MPS robot move sequence* or simply *robot move sequence* that specifies all robot operations during the MPS cycle. Sriskandarajah et al. [89] define *Concatenated Robot Move Sequences (CRM Sequences)* as a class of MPS cycles in which the same 1-unit cycle of robot actions are repeated n times. Thus, the problem in multiple parts case is to find the robot move sequence and the part input sequence of the MPS.

Now let us present the classification scheme for robotic flowshops, which will help us through the rest of this study. The standard classification scheme for scheduling problems introduced by Graham et al. [36] can be denoted as

$\psi_1|\psi_2|\psi_3$ where ψ_1 indicates the scheduling environment, ψ_2 indicates the job characteristics or restrictive requirements and ψ_3 defines the objective function to be minimized. Hall et al. [43] extended this scheme to capture the scheduling problems arising in robotic cells, as follows:

Under ψ_1, we have:

$MRCm$	=	a mobile-robot cell with m machines.
$RCCm$	=	a robot-centered cell with m machines.
$IRCm$	=	an in-line robot cell with m machines.

Under ψ_2, we have:

k	=	the number of part-types.
r-unit(s)	=	the problem is being solved over robot move cycles that produce r units.
S	=	robot move cycle S is used alone.
$\delta_i = \delta$	=	the travel time between any pair of consecutive machines is equal.
$\epsilon_i = \epsilon$	=	the load and unload times at all machines are equal.

Under ψ_3, we have:

C_t	=	long run average time to produce one part.
C_{max}	=	the makespan for the manufacture of a given set of jobs.

In the next section, we will present basic results of the previous research on robotic cell scheduling problems.

2.2 Results from Previous Studies

There is vast amount of research on robotic cell scheduling problems. Some date as far as 1970s, but the majority has been performed since 1990. We will analyze the results from previous studies under the headings of identical parts case, multiple parts case and some other cell configurations that have particular assumptions. Crama et al. [21], Lee et al. [69] and Geismar et al. [30] also provide surveys in this area.

2.2.1 Identical Parts Case

The identical parts robotic cell scheduling problem is simpler than its multiple parts counterpart since the part sequencing problem vanishes in identical parts case. However, like most scheduling problems, analytical results are found for problems where there the number of machines is small.

The paper by Sethi et al. [86] can be considered as the initiation of the robotic cell scheduling literature. In this study, the objective is to maximize the throughput or in other words minimize the cycle time. One of the problems considered in this study is "one part type problem with two machines", more specifically, $RCC2|k = 1, \delta_i = \delta, \epsilon_i = \epsilon|C_t$. For this problem they prove that the optimal solution is a 1-unit cycle. Since there are a total of two feasible 1-unit cycles in a 2-machine cell, they determine the regions of optimality for each of these cycles by comparing the cycle times of these two cycles with each other. Another problem considered in the paper is "one part type problem with three machines", that is, $RCC3|k = 1, \delta_i = \delta, \epsilon_i = \epsilon, 1 - unit|C_t$. Determining the sequence of robot moves constituting a 1-unit cycle that minimizes the cycle time is considered. In this problem, only 1-unit cycles are considered since the analysis of the problem without this restriction is difficult and perhaps

intractable. As a solution to this problem, a decision tree is constructed in order to determine the optimal policy. It is also proved that the number of one-part cycles in the m-machine case is exactly $m!$. Another important result of the paper is the conjecture that *optimal 1-unit cycles are superior to every n-unit cycle, for* $n \geq 2$.

Crama et al. [19] consider the problem $RCCm|k = 1, \delta_i, \epsilon_i, 1 - unit|C_t$. They show that, when there is only one type of part to be produced and considering only 1-unit cycles, the problem can be solved in (strongly) polynomial time, even if the number of machines is viewed as an input parameter of the problem. This generalizes previous results established by Sethi et al. [86]. This result is achieved by proving that the set of pyramidal permutations necessarily contains an optimal solution of the problem. Pyramidal permutations have been previously introduced in the framework of the travelling salesman problem; see e.g. Gilmore et al. [35]. Let $\pi = (A_0, A_{i_1}, \ldots, A_{i_k}, A_{i_{k+1}}, \ldots, A_{i_m})$. Then, π is pyramidal if $1 \leq i_1 < \ldots < i_k = m$ and $m > i_{k+1} > \ldots > i_m \geq 1$. An algorithm is given which computes the cycle time of a schedule described by a pyramidal permutation. Lastly, a dynamic programming approach is presented that solves the identical parts cyclic scheduling problem with the restriction that one unit is produced in each cycle in $O(m^3)$ time where m is the number of machines in the cell. Another result of that study is the derivation of the upper and the lower bounds on the optimal cycle time.

Hall et al. [43] consider 3-machine cells producing single part-types and prove that, the repetition of 1-unit cycles dominates more complicated policies that produce two units. The validity of the conjecture of Sethi et al. [86] for 3-machine robotic flowshops is established by Crama et al. [20]. Brauner and Finke [9] simplify this proof. In a later study, Brauner et al. [10] prove that 1-unit cycles do not necessarily yield optimal solutions for cells of size four or

large. They present examples of such cases.

Dawande et al. [24] consider a different case of identical parts robotic cell scheduling problem. They study the problem of finding the optimal robot move cycle that minimizes the cycle time in an m machine robotic cell. However, they consider only the 1-unit cycles. Differing from the literature, they assume that the robot travel time between any pair of machines is constant, which is referred to as *constant travel time robotic cells*. Such cells are used in some manufacturing systems such as manufacturing of wafers. They provide a polynomial time algorithm for finding an optimal 1-unit cycle.

Dawande et al. [23] show that cyclic schedules which repeat a fixed sequence of robot moves indefinitely are the only ones that need to be considered in order to maximize the long-term average throughput. Additionally, for the different classes of robotic cells studied in the literature, the authors discuss the current state of knowledge with respect to cyclic schedules. Geismar et al. [29] consider an m-machine flexible robotic cell. They assume that each part has one operation to be performed on each machine which makes a total of m operations. They also assume that each part visits the machine in the same order. However, the operations can be performed in any order and each machine can be configured to perform any operation. They try to determine the assignment of the operations to the machines so that the throughput is maximized. They consider both the cases where the assignment of the operations remains the same throughout the processing of the lot and it varies for successive parts within a processing lot for 2, 3 and 4-machine cells.

A considerable amount of research in robotic cell scheduling area considers the no-wait constraints which are required in some manufacturing systems such as plastic molding, electroplating and steel manufacturing. In these systems, material handling is mainly done by an automated material handling device

such as AGV's, hoists or robots. There is a vast amount of literature on no-wait constraints in hoist and AGV scheduling problem areas. Since AGV scheduling, hoist scheduling and robotic cell scheduling problems can be considered as special cases of one another, the results for one of them can be extended to be used for another, For example a single loop, single hoist scheduling problem can be considered as a robotic cell scheduling problem.

One such study with no-wait constraints is the study of Kats et al. [61]. They consider an m machines identical parts robotic cell scheduling problem with the objective of finding the 1-unit robot move cycle that minimizes the cycle time. They assume that any machine may occur more than once in the processing sequence of the parts. A polynomial algorithm which solves the problem in $O(K^5)$ is presented. Here K is the number of processing stages in the part's production. If the re-entrance constraint is relaxed, they show that the same algorithm has complexity $O(m^4)$, where m is the number of machines. In a later study, Levner et al. [70] consider the same problem without re-entrance constraints. They present an algorithm which in turn improves the complexity of the previous one to $O(m^3 \log m)$.

In a most recent study in no-wait robotic cells, Che et al. [15] consider an m machine robotic cell with identical parts and constant processing times. The objective is to find the optimal 2-unit robot move cycle that minimizes the cycle time. They propose an algorithm which solves the problem in $O(m^8 \log m)$ time. They also extend this algorithm for the case of two nonidentical parts. They present computational results which show that the algorithm effectively finds the 2-unit cycles that minimizes the cycle time.

2.2.2 Multiple Parts Case

As already mentioned, multiple parts problems are harder than the identical parts problem even for small number of machines. Recall that, the problem is to find the robot move sequence and the part input sequence for the MPS that jointly minimize the cycle time.

In their study, Sethi et al. [86] consider multiple parts case also. More specifically they consider $RCC2|k \geq 2, \delta_i = \delta, \epsilon_i = \epsilon, S_1^2(S_2^2)|C_t$. Given a fixed sequence for the robot moves in a 2-machine cell and the desired production ratios of the part types to be produced (MPS), determining the schedule of parts at the input station that minimizes the cycle time is considered. As a solution to this problem a polynomial time algorithm that determines an optimal multi-part cycle is proposed.

Kise et al. [65] consider 2-machine multiple parts problem with the objective of minimizing the makespan. They propose an $O(n^3)$ procedure that solves the problem based on the known Gilmore and Gomory algorithm (Gilmore and Gomory [34]). For the same problem, if the transportation time between the machines is job dependent, then the problem is equivalent to an asymmetric travelling salesman problem and is NP-hard in the strong sense (Stern and Vitner [91]).

Again for the 2-machine multiple parts problem, Logendran and Sriskandarajah [75] consider three different layouts and establish optimal robot sequences for these layouts. The problem of determining the optimal sequence of multiple part types is shown to be equivalent with a 2-machines no-wait flow shop problem, and is solved by Gilmore and Gomory's algorithm. Besides the analysis of a single MPS, production of multiple MPSs is also analyzed.

Hall et al. [43] attack the part scheduling and robot move sequencing

problems simultaneously. They consider $MRC2|k \geq 2, \delta_i = \delta, \epsilon_i = \epsilon, |C_t$. They prove that CRM sequences generally do not give the optimal MPS robot move sequences and they provide an example depicting this situation. An $O(n^4)$ algorithm is provided for this case which gives the robot move cycle and part input sequence that jointly minimize the cycle time, where n is the number of parts in the MPS. They also consider the problem $MRC3|k \geq 2, \delta_i = \delta, \epsilon_i = \epsilon, S_1^3(S_2^3, S_3^3, S_4^3, S_5^3, S_6^3)|C_t$. They show that the optimal part sequencing problems associated with 4 of the 6 potentially optimal robot move cycles for producing 1-unit are polynomially solvable and for the remaining 2 cycles, they show that the recognition version of the part sequencing problem is NP-complete. More specifically, they show that the optimal part schedule based on S_1^3 is trivially solvable whereas the optimal part schedule based on S_3^3, S_4^3 and S_5^3 can be solved polynomially using an algorithm based on the Gilmore-Gomory [34] algorithm. On the other hand, finding the optimal part schedules for S_2^3 and S_6^3 are NP-hard. Also, the conditions on the relative lengths of processing times compared to robot move times under which the last 2 cycles are dominated by the other 4 are given.

Aneja et al. [5] also consider $MRC2|k \geq 2, \delta_i = \delta, \epsilon_i = \epsilon, |C_t$ like Hall et al. [43] and improve their algorithm which finds the optimal robot move sequence and the part input sequence. They model the problem as a special case of TSP and provide an algorithm of complexity $O(n \log n)$.

Sriskandarajah et. al. [90] consider $MRCm|k \geq 2, \delta_i, \epsilon_i|C_t$. They classify 1-unit robot move cycles in an m-machine cell, for $m \geq 2$, according to the tractability of their associated part sequencing problems. The classification is as follows:

U : Sequence independent (trivially solvable),

V1 : Capable of formulation as a TSP but polynomially solvable,

V2 : Unary NP-hard TSP models,

W : Unary NP-hard, but not having TSP structure.

As a consequence of this classification, it is proved that the part sequencing problems associated with exactly $2m-2$ of the $m!$ available robot cycles are polynomially solvable. The remaining cycles have associated part sequencing problems which are unary NP-hard.

Another important result of the paper is as follows: in an m-machine robotic cell with $m \geq 4$ there are $m!$ robot move cycles of which:

(a) One U-cycle defines a trivially solvable part sequencing problem,

(b) $2m-3$, V1-cycles define a part sequencing problem which is solvable in $O(nlogn)$ time,

(c) $\sum_{t=1}^{\lfloor m/2 \rfloor} \binom{m}{2t} - 2m + 3$, V2-cycles define a part sequencing problem which can be formulated as a TSP and which is unary NP-hard.

(d) $m! - 1 - \sum_{t=1}^{\lfloor m/2 \rfloor} \binom{m}{2t}$, W-cycles define a part sequencing problem which in general cannot be formulated as a TSP, and which is unary NP-hard.

Hall et al. [44] consider a 3-machine cell which produces multiple part-types and prove that in two out of six potentially optimal robot move cycles for producing one unit, the recognition version of the part sequencing problem is unary NP-complete. The intractability of the part sequencing problem not restricted to any one-unit cycle, that is, $MRC3|k \geq 2, \delta_i, \epsilon_i|C_t$, is also proved. Lastly, an algorithm is provided which initializes an empty cell into a steady state as quickly as possible for any potentially optimal one-unit cycle.

Kamoun et al. [56] consider various problems in multiple parts scheduling case and design and test heuristic procedures for the part sequencing problem.

They provide heuristics for the following problems: $MRC3|k \geq 2, \delta_i, \epsilon_i, S_2^3|C_t$ and $MRC3|k \geq 2, \delta_i, \epsilon_i, S_6^3|C_t$ which were shown to be NP-hard by Hall et al. [43]. They also consider the most general 3-machine case, $MRC3|k \geq 2, \delta_i, \epsilon_i|C_t$, in which all possible robot move cycles are considered. Hall et al. [44] prove that its recognition version is unary NP-complete. An efficient heuristic is defined and tested for this problem also. The ways of extending these heuristics to four machine cells as well as larger cells are illustrated. They also consider a cell design problem which involves organizing several machines into $R \geq 2$ cells, which are arranged in a serial production process with intermediate buffers.

Agnetis [1] consider a multiple parts no-wait robotic cell scheduling problem with m machines. They assume that the parts are grouped into lots of identical parts. They propose an ε-approximate algorithm which is based on the solution to a transportation problem. Computational analysis results are presented.

2.2.3 Other Cell configurations

A new area of research is focusing on the robotic cells served by a robot with a dual gripper. These robots are considered as a means to increase throughput. These types of robots can hold two parts simultaneously and thus, make it possible to unload a part from a machine and load it with another part at the same time. This ability results in a huge increase in the number of possible robot move cycles. For such robots, another time called the *repositioning time* is defined. This is the required time to reposition the second gripper of the robot in front of the machine after the first gripper finishes its loading/unloading operation. It is generally assumed that the repositioning requires much less time than the robot's movement between two adjacent machines or any machine's processing time [30].

Droboutchevitch et al. [26] consider identical parts case and develop a formula to find the number of nondominated cycles in a general cell with m machines. They show that for $m = 2$ there are 52 feasible 1-unit cycles. However, they show that 13 of them dominate the rest.

Sethi et al. [85] also consider the dual gripper robotic cell scheduling problem with identical parts and m machines. Since this problem is much more complex than the case of single gripper robotic cell scheduling problem, they extend the existing analytical framework to develop all 1-unit cycles for dual gripper robotic cells. They consider only the 1-unit robot move cycles and examine the cycle time advantage (or productivity advantage) of using a dual gripper robotic cell rather than a single gripper robot. The best possible improvement achieved by implementing a dual gripper robot appears to be reducing the cycle time by half. They also propose a practical heuristic algorithm to compare productivity of a single gripper and a dual gripper for given cell data. Conditions which indicate the use of a dual gripper robot include [30]:

1. m is not large and $max_i\{P_i/(\delta + 2\epsilon)\}$ is large.
2. m is large and $max_i\{P_i/(\delta + 2\epsilon)\}$ is not large.
3. $\epsilon/\delta \leq 1$.

Sriskandarajah et al. [89] consider dual gripper robotic cells with multiple parts. Focusing only on CRM sequences, they develop the notational and modelling framework for cyclic production of multiple parts in dual gripper robotic cells. They demonstrate that the recognition version of the part sequencing problem subject to a given robot move sequence is unary NP-complete for 6 out of 13 undominated sequences. For the special case of negligible gripper switching time, they identify the robot move sequence that

gives the minimum cycle time. For the general case, they provide an efficient heuristic, which is empirically tested. Also computational experiments are made to study the productivity gains of using a dual gripper robot in place of a single gripper. They find the mean relative improvement range between 18% and 36%.

Drobouchevitch et al. [25] also consider dual gripper robotic cells with multiple parts. They consider 2-machines case. They focus on the intractable problem of parts sequencing in a 2-machine dual gripper robot cell. They develop a heuristic based on Gilmore-Gomory [34] that provides 3/2-approximation of the optimum for the 6 NP-hard CRM sequences. A linear program is used to establish the performance guarantee without actually calculating a lower bound. This approach is original in the literature of scheduling robotic cells.

There are many other interesting robotic cell configurations in the literature. An important class of problems addresses robotic cells involving more than one robot. Problems of this type can be found in Karzanov and Livshits [59] and Lieberman and Turksen [73]. In one of such problems Kats et al. [60] study the problem of m machines identical parts no-wait robotic cell scheduling problem with the objective of minimizing the number of robots required to meet a cycle time of T. They propose an $O(m^5)$ time algorithm that solves the problem optimally. Also in another study, Kats and Levner [62] consider minimization of cycle time over the 1-unit robot move cycles with the no-wait constraint and identical parts. They solve the problem for single and several robots. They assume that the set of machines served by each robot is known in advance. They solve the problem in $O(m^3 log m)$ time. They also show that the cyclic multi-robotic problem of the paper can be interpreted as a new polynomially solvable case of the cyclic no-robot job-shop scheduling problem.

A special case of robotic cell scheduling is when all of the machines are identical and working in parallel. Hall et al. [45] consider these systems with different objectives (e.g. makespan, C_{max}; maximum lateness, L_{max}; total tardiness, $\sum T_j$ etc.) and derive complexity results for these problems. Geismar et al. [31] consider identical parts, parallel machines robotic cell where the robot travel time is assumed to be constant for any pair of machines. The authors provide guidelines to determine whether parallel machines will be cost-effective for a given implementation. They also provide a simple formula for determining how many copies of each machine are required to meet a particular throughput rate and an optimal sequence of robot moves for a cell with parallel machines under a certain common condition on the processing times.

Geismar et al. [32] combine dual gripper robotic cells and robotic cells with parallel machines. The robot travel time is assumed to be constant for any pair of machines. They provide a structural analysis of cells with one or more machines per processing stage to obtain first a lower bound on the throughput and subsequently an optimal solution under specific conditions. In another study, Geismar et al. [33] consider a parallel machine robotic cell with multiple robots that is used by a semiconductor equipment manufacturer. The travel times are assumed to be Euclidean. The authors describe a schedule of robot moves that is optimal under a common set of conditions for large cells containing multiple robots. When this set of conditions does not hold, even though optimality could not be proven, this schedule is shown to be superior to one currently in use by some semiconductor manufacturers. They also present a scheme that allows the robots to operate concurrently, efficiently, and with no risk of colliding.

Blazewicz et al. [8] consider a two-stage FMS, in which they assume limited buffers between the machines. A specific assumption of this problem was that some parts need additional operation so will leave the system after the first

machine and turn back later. Another difference of this research is that, they consider setup times and assume that the production is made with batches of identical parts. They also consider release dates of the parts and provide heuristics to minimize the makespan.

Some studies in this area consider different objective functions other than minimizing the makespan or the cycle time. For instance, Song et. al. [88] and Jeng et. al. [55] try to minimize the sum of completion times. On the other hand Levner and Vlach [72] consider an objective which minimizes some penalty function of the maximum lateness.

Hurink and Knust [52] consider a single machine scheduling problem which arises as a subproblem in a jobshop environment where the jobs have to be transported between the machines by a single transport robot. They present a tabu search algorithm for this problem, where they consider it as a generalized TSP.

Lastly, Han and Cook [47], develop a mathematical model and a solution algorithm for solving a robot acquisition and cell formation problem. They formulate the problem as a multi-type two-dimensional bin packing problem, which is known to be NP-hard. They develop and implement a heuristic algorithm. The computational results show that the problems can be solved with an optimality gap of less than 1% and over 70% of all solutions (334 out of 450) were optimal.

2.3 Multicriteria scheduling

A single criterion is used in most of the existing scheduling studies. Algorithms which focus entirely on optimizing one criterion may perform poorly for others

since most of the criteria are conflicting with each other. The trade-offs involved in considering several different criteria provide useful insights to the decision maker. For example, a solution which minimizes the cycle time (long run average time to produce one part) may perform poorly in terms of cost. Thus, in the context of real life scheduling problems it is more relevant to consider problems with more than one criterion. Multicriteria and bicriteria scheduling models can be reviewed from Hoogeveen [50] and Nagar et al. [78]. Multicriteria scheduling can provide good solutions for more than one objective. There are different ways to deal with multiple criteria. One way is combining objectives with linear, quadratic and Tchebycheff functions by assigning weights to each objective. Another way is finding efficient solutions which provide more than one alternative to the decision maker.

In multi-objective optimization problems, approximation quality of the generated efficient set is important to the decision maker. In the literature, there are different approximation quality evaluation metrics developed. These metrics are useful for comparing different algorithms. A review and discussion on existing metrics is available in Zitzler et al. [100]. Wu and Azarm [98] propose some quality evaluation measures to compare efficient sets generated by different multi-objective optimization methods.

The most common objectives used in multicriteria scheduling models are minimizing flow time, number of tardy jobs, maximum tardiness, total tardiness and total earliness. Koktener and Koksalan [68] use simulated annealing and Koksalan and Keha [67] use genetic algorithms to solve a bicriteria scheduling problem on a single machine. Ruiz-Torres et al. [84] study the bicriteria identical parallel machine scheduling problem with the objectives of minimizing the number of late jobs and minimizing the average flow time. A simulated annealing procedure is proposed to generate nondominated solutions. Suresh and Chaudhuri [92] propose a tabu search algorithm to solve bicriteria

scheduling problem for unrelated parallel machines with the objectives of minimizing the makespan and minimizing the maximum tardiness. Mohri et al. [76] solve the bicriteria scheduling problem on three identical parallel machines. The tradeoff curve which minimizes the makespan and maximum lateness is found. Tiwari and Vidyarthi [93] propose a genetic algorithm to solve machine loading problem with the availability of machining time and tool slots constraints. The objectives are minimizing the system unbalance and maximizing the throughput, which are the most commonly used objectives in FMS scheduling with multiple machines. Bernardo and Lin [6] consider the nonidentical parallel machine scheduling problem with the objectives of minimizing the total tardiness and minimizing the setup costs. Gupta and Ruiz-Torres [42] consider the objectives of minimizing total flow time and minimizing total number of tardy jobs simultaneously and propose heuristic algorithms to generate efficient solutions. Gupta and Ho [41] provide solution methods for the problem of minimizing makespan subject to minimum flow time for two parallel machines. Cao et al. [14] consider the machine selection and scheduling decisions together in order to minimize the sum of machine cost and job tardiness. Alagoz and Azizoglu [3] study a problem with the objectives of minimizing total completion time and minimizing number of disrupted jobs in a rescheduling environment.

Although, the local search heuristics do not guarantee to find all efficient solutions, they can find approximately efficient solutions for multiple criteria in reasonable computation times.

2.4 Controllable Processing Times

In robotic cells, highly flexible Computer Numerical Control (CNC) machines are used for the metal cutting operations so that the machines and the robot can interact in real time. Machining conditions such as the cutting speed and the feed rate are controllable variables for these machines. Consequently, the processing time of any operation on these machines can be reduced by changing the machining conditions at the expense of incurring extra cost resulting in the opportunity of reducing the cycle time. Due to this reasoning, assuming the processing times to be fixed on each machine is not realistic.

The first studies considering controllable processing times in scheduling problems are by Vickson [95], [96]. He considers the single machine problem with average flow cost and maximum tardiness objectives together with the total compression cost. The total compression cost is assumed to be a linear function of the processing time. An assignment model is proposed for the average flow cost objective. An algorithm is proposed for the maximum tardiness problem. Most of the studies consider single machine scheduling problems with linear compression cost functions (Panwalkar and Rajagopalan [81], Van Wassenhove and Baker [97], Daniels and Sarin [22]). Although this assumption simplifies the problem, it is not realistic in most cases because it does not reflect the law of diminishing returns. Nowicki and Zdrzalka [79] provide a survey for sequencing problems with controllable processing times which have a linear cost function. Mukhopadhyay and Sahu [77] consider the minimization of makespan as a primary objective and the minimization of the machining costs as a secondary objective.

There are few studies considering multiple machine problems. Zhang at al. [99] propose a 3/2-approximation algorithm for solving the unrelated parallel machine scheduling problem with the objectives of minimizing the

total weighted completion time and the processing cost. Ishii et al. [53] consider the uniform parallel machine scheduling problem with preemption. They propose polynomial time algorithms in order to find optimal speeds of the processors with the completion time and the processing costs objectives. Nowicki and Zdrzalka [80] consider parallel machine scheduling with completion time and processing cost objectives (preemption allowed). A greedy algorithm is proposed to find the efficient frontier for identical machines. An algorithm is proposed for the uniform machine case in order to find the ϵ-approximation of the efficient frontier. Trick [94] proposes an integer programming formulation and a heuristic to solve the multiple capacitated machine problem with makespan objective.

The minimization of earliness and tardiness objective is considered in a few number of papers considering controllable processing times. Most of them consider the common due date assumption. Panwalkar and Rajagopalan [81] consider a single machine problem with controllable processing times. The objective is minimizing the sum of earliness and tardiness penalties and total processing costs. The common due date is a decision variable that should be determined with the proposed assignment model. Liman et al. [74] replace the common due date assumption with the common due window assumption. They consider the objective of minimizing the costs associated with the common due window location, its size, processing time reduction and earliness and tardiness penalties. The problem is formulated as an assignment problem. Alidaee and Ahmadian [4] solve the non-identical parallel machine scheduling problem. They consider the total weighted earliness and tardiness objective with common due date assumption. The problem is formulated as a transportation problem. Cheng et al. [16] study the unrelated parallel machine scheduling problem with controllable processing times. The cost is a convex function of the amount compressed. Karabati and Kouvelis [57] discuss simultaneous scheduling and

optimal processing time decision problem to maximize the throughput for a multi-product, deterministic flow line operated under a cyclic scheduling approach. These studies assume linear job compression costs. A nonlinear relationship is considered between processing times and production resource by Shabtay and Kaspi [87]. They consider the classical single machine scheduling problem of minimizing the total weighted flow time with controllable processing times. In their setting, the processing times can be controlled by allocating a continuously nonrenewable resource such as financial budget, overtime and energy. They assume the processing times to be convex, nonlinear functions of the amount of the resource consumed. The objective in their case is to allocate the resource to the jobs and to sequence the jobs so as to minimize the total weighted flow time. They propose polynomial time exact algorithms for small to medium size problems and heuristic ones for large scale problems. Kayan and Akturk [63] consider a single machine bicriteria scheduling model with controllable processing times. They select total manufacturing cost and any regular scheduling measure-one which cannot be improved by increasing the processing times such as makespan, completion time or cycle time, as the two objectives. They derive lower and upper bounds on processing times and provide methods to determine an approximate efficient frontier for the problem.

The manufacturing cost we consider in this study is the sum of the machining and tooling costs which are determined according to the operating costs of machines and specific cutting tool related parameters that are taken from the machining handbooks. When the processing times increase, the machining cost increases and the tooling cost decreases. The total manufacturing cost is a convex function of processing times.

2.5 Summary

In this section the relevant literature is reviewed. In the robotic cell scheduling literature the cell is assumed to be working like a flowshop type system. That is, all parts pass through all machines in the same order and the processing time of each part on each machine is a predefined parameter. Furthermore, the design of the cell including the layout of the cell and the number of the machines inside the cell are assumed to be predetermined. This may be due to the fact that the robotic cell scheduling problem is originated from chemical and electroplating industries. As a consequence, the current literature ignores the flexibility of the machines. However, the use of robotic cells in metal cutting industries in which the highly flexible CNC machines are used is increasing rapidly. Considering the machines to be flexible leads to new research topics which are practical and more general than the problems considered in the current literature. Additionally, till now, a single criteria is used for the robotic cell scheduling problems and there are no studies considering the manufacturing cost as an objective which has the highest priority in process planning. In this study, we consider the deficiencies of the current literature. We assume the machines to be flexible CNC machines. As a consequence, for the first time in robotic cell scheduling literature the processing times are assumed to be decision variables. Again for the first time in this literature a cost based objective function is considered and a bicriteria model is formulated and solved. Rather than operational problems as in the current literature, some design problems are also considered which initiate a new research direction.

Chapter 3

Problem Definition

In this section we give a formal definition of our problem and introduce the basic terminology and notation. In chapters 7 and 8, different than chapters 4, 5 and 6, we will consider a bicriteria problem and a different source of flexibility. Thus, in order to increase the readability and the understanding, problem definitions and some required notations for chapters 7 and 8 are placed at the beginning of those chapters.

In the existing literature, the allocation of the operations to each machine is assumed to be constant. Each part goes through the machines in the same order and the processing time of each part on each machine is a known parameter, P_i, for machine $i = 1, 2, \ldots, m$. For given processing times, the optimum robot move cycle minimizing the cycle time is to be determined. In some manufacturing operations such as chemical electroplating and plastic molding, since the parts must follow the same sequence of operations, this assumption is meaningful and these operations mostly require no-wait constraints (see for example Geismar et al. [30]). However, in the case of metal cutting operations for which the CNC machines are used, the allocation of the operations to

the machines is also a decision problem and Akturk et al. (2005) prove that considering the allocation of the operations to the machines as decision variables can improve the efficiency of the cells. This is because CNC machines are capable of performing a wide range of operations with means of using different cutting tools. Therefore, assuming that processing times are fixed on each CNC machine may not accurately represent the capabilities of the CNC machines and limits the number of alternatives unnecessarily for these systems.

In this study, we consider a robotic cell consisting of identical CNC machines. We assume that there is an infinite number of identical parts to be processed, where a part is defined by a fixed set of operations to be performed in any order and each operation requires a unique type of cutting tool to be performed. More specifically, one tool can perform different operations but an operation can only be performed by a unique type of tool. The cutting tools are stored in the tool magazines of these machines. A machine is capable of performing any operation as long as the required cutting tools are loaded on its tool magazine. We will analyze the case where all the required cutting tools are loaded on all machines, that is all machines are capable of performing all the required operations of each part, as well as the case where some tools have single copies so that only the machine loaded with this tool can perform the corresponding operation while the rest of the tools are loaded on all machines so that all machines can perform the corresponding operations.

The identical parts have a number of operations to be completed on the machines and the individual operation times are known and identical for all machines. Let t_l be the processing time for operation l. Furthermore, let O be the set of operations that can be processed by all machines (the required cutting tools for these operations are loaded to all machines). The processing times of a part on each machine depend on the allocation of these operations to the machines. An allocation of operations to the m machines

means partitioning set O into m subsets; $O_1, O_2, \ldots, O_m$, where O_i is the set of operations allocated to machine i. Consequently, by finding the optimal allocation of the operations to the machines we can minimize the cycle time. Moreover, the allocation of the operations to the machines need not be the same for all parts. The operations of each part can be allocated differently. Since during one repetition of the cycle more than one part can be processed on different machines simultaneously, having different allocations for these parts is an opportunity to minimize the cycle time. Let us consider the 2-machine case. For one part, O can be partitioned as Z_1 and $(O \setminus Z_1)$ and for the next part to be processed it can be partitioned as Z_2 and $(O \setminus Z_2)$ where $Z_1 \neq Z_2$. So the processing time of the first part on the first machine is $\sum_{l \in Z_1} t_l$ while the processing time of the second part on the same machine is $\sum_{l \in Z_2} t_l$. The processing times of the first and second parts on the second machine are $\sum_{l \in (O \setminus Z_1)} t_l$ and $\sum_{l \in (O \setminus Z_2)} t_l$, respectively. Thus the processing times of the machines may change from one repetition of the robot move cycle to the other. However, since we consider cyclic production, that is, the robot performs the same set of activities repeatedly, after some point the allocation of the operations of a part, say the $(k+1)^{st}$ part $k = 1, 2, \ldots$, becomes identical with the first part as shown in Figure 3.1. Then, the allocation of the operations of the parts 1 through k is used in the same order repeatedly for the remaining parts. That is, k is the period of the allocation types. Note that, for a specific k, there are a finite (though large) number of different ways to allocate the operations of the parts. For the clarity of the consequent discussion, we need the following definition and notation.

Definition 3.1 *Let $\Pi_k = [\pi_{ij}]$ denote a specific allocation matrix with k different allocation types. The $(i,j)^{th}$ entry, π_{ij}, $i = [1, 2, \ldots, k]$ and $j = [1, 2, \ldots, m]$, of this matrix corresponds to the set of operations allocated to the j^{th} machine for every $(rk + i)^{th}$ part in the infinite sequence where*

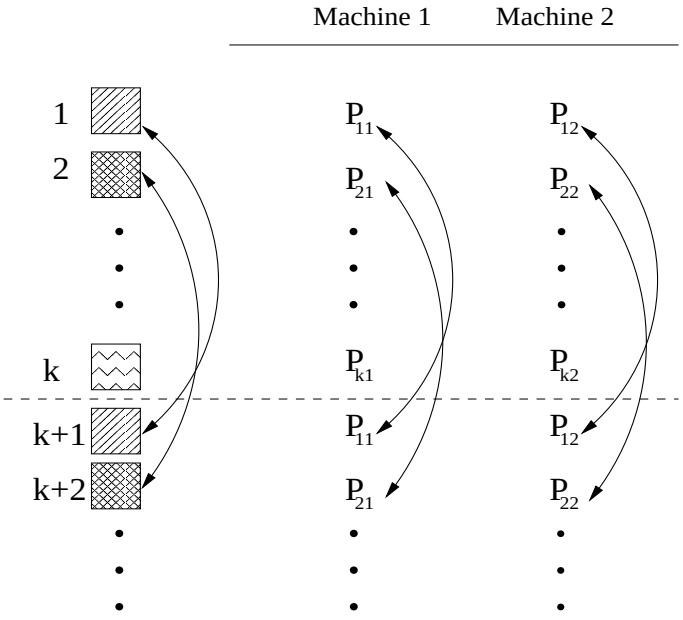

Figure 3.1: Different allocation of k parts to the machines

$r = 0, 1, 2, \ldots$. *For this matrix we have:*

- *Each row corresponds to a proper m-partitioning of the operation set O. With our notation, for any i,* $\pi_{i1} \cup \pi_{i2} \cup \ldots \cup \pi_{im} = O$ *and* $\pi_{ij} \cap \pi_{il} = \emptyset$, $j \neq l$. $\forall j, l$.

- *No two rows are identical.*

We also let Π_k^* *denote the optimal allocation of operations when a total of k different allocation types is used and let* Π^* *denote the optimal allocation of operations over all k.*

For example for a cycle in a 3-machine cell for which a specific two different allocation types are used, the allocations of the operations are represented as

follows:

$$\Pi_2 = \begin{bmatrix} \pi_{11} & \pi_{12} & \pi_{13} \\ \\ \pi_{21} & \pi_{22} & \pi_{23} \end{bmatrix}.$$

That is, there are two distinct 3-partitions of operations to the machines which are used alternatingly. Note that a 2-unit robot move cycle with one-allocation type means that all parts to be processed have the same allocation of operations to the machines. A 2-unit cycle with two-allocation types means that two identical parts with different allocation of operations to the machines are produced alternatingly. The cycle time for this case is found by calculating the total time to finish one repetition of the cycle and dividing by two. On the other hand, a 2-unit cycle with three-allocation types means that the allocation of operations for parts $1, 4, \ldots, 3z+1$ where $z = 1, 2, \ldots$ are the same. This also holds for parts $2, 5, \ldots 3z+2$ and parts $3, 6, \ldots 3z$. In order to find the cycle time of an n-unit cycle with k-allocation types, we have to find the least common multiple of n and k; let this number be M. To find the average time to produce one part, we have to repeat the cycle M/n times and divide the total time by M. In particular, for a 2-unit cycle with three-allocation types, we have to repeat the cycle 3 times and divide the total time by $M = 6$.

We will use the following notation throughout this thesis.

t_l : Processing time of operation l. Note that the processing time of operation l on all machines are equal, $\forall l = 1, 2, \ldots, p$, where p is the number of operations of each part.

P : Total processing time of the operations that will be allocated to the machines, $P = \sum_{l \in O} t_l$. Note that the sum of the processing times on m machines corresponding to each row of the allocation matrix is also equivalent to P.

P_{ij} : Total processing time on machine j for the part which corresponds to the i^{th} row of the specific allocation matrix Π. That is, $P_{ij} = \sum_{l \in \pi_{ij}} t_l$. Also, we let $P_\pi = [P_{ij}]$.

w_{ij} : Waiting time of the robot for machine j to finish the processing of the part which is produced according to the i^{th} row of the allocation matrix Π. Note that, if the processing of the part is already finished when the robot arrived to machine j, then the waiting time is 0.

ϵ : Consistent with the literature we assume that loading/unloading times for all machines are the same.

δ : Time taken by the robot to travel between any two adjacent stations. We assume this time to be additive. That is, the time required for the robot to move from machine i to machine j is the sum of the movement times between all of the intervening pairs of machines in the route from machine i to j where $i, j \in [1, 2, \ldots, m]$. That is, the robot travel time from machine i to j is $|j - i|\delta$. So the triangular equality is satisfied. For example, the transportation time from input buffer to the second machine is $|2 - 0|\delta = 2\delta$.

$T_{S(\Pi_k)}$: Cycle time, i.e., the long run average time that is required to produce one part using robot move cycle S and the specific allocation matrix Π_k.

The following example will be helpful in understanding how different allocations of the operations affect the cycle time.

Example 3.1 Let us consider a 3-machine cell and assume that each part has 5 operations to be performed on the three machines with corresponding operation times $t_1 = 30$, $t_2 = 25$, $t_3 = 35$, $t_4 = 30$ and $t_5 = 15$. Thus, total processing time of each part is $P = 135$. Let us also assume that $\epsilon = 2$ and

$\delta = 4$. Now consider the robot move cycle S_6^3 which is defined by the following activity sequence $A_0A_3A_2A_1$. In our study, the cycle time derived by Sethi et al. [86] correspond to the case where the allocations of the operations of all parts are identical. Let Π_1 be a specific allocation. Then, the cycle time for this case is the following:

$$T_{S_6^3(\Pi_1)} = 8\epsilon + 12\delta + max\{0, P_{11} - 4\epsilon - 8\delta, P_{12} - 4\epsilon - 8\delta, P_{13} - 4\epsilon - 8\delta\}.$$

The optimal allocation in this case is: $\pi_{11}^* = \{1, 5\}$ with $P_{11}^* = 45$, $\pi_{12}^* = \{2, 4\}$ with $P_{12}^* = 55$, and $\pi_{13}^* = \{3\}$ with $P_{13}^* = 35$. The corresponding cycle time is:

$$T_{S_6^3(\Pi_1^*)} = 64 + max\{0, 45 - 40, 55 - 40, 35 - 40\} = 79.$$

Now let us assume that two different allocation types are used repeatedly. That is, a specific allocation is now represented by Π_2. The new cycle time to produce one part for this case is the following:

$$\begin{aligned} T_{S_6^3}(\Pi_2) = 8\epsilon + 12\delta \quad & + \tfrac{1}{2} max\{0, P_{11} - 4\epsilon - 8\delta, P_{22} - 4\epsilon - 8\delta, P_{13} - 4\epsilon - 8\delta\} \\ & + \tfrac{1}{2} max\{0, P_{21} - 4\epsilon - 8\delta, P_{12} - 4\epsilon - 8\delta, P_{23} - 4\epsilon - 8\delta\}. \end{aligned}$$

The optimal allocations of the operations for this case are, in the first allocation type, $\pi_{11}^* = \{1, 2\}$, $\pi_{12}^* = \{3\}$, $\pi_{13}^* = \{4, 5\}$. In other words, $P_{11}^* = 55$, $P_{12}^* = 35$ and $P_{13}^* = 45$. As for the second allocation type, $\pi_{21}^* = \{4, 5\}$ with $P_{21}^* = 45$, $\pi_{22}^* = \{1, 2\}$ with $P_{22}^* = 55$, and finally $\pi_{23}^* = \{3\}$ with $P_{23}^* = 35$. Then the corresponding cycle time is the following:

$$\begin{aligned} T_{S_6^3}(\Pi_2^*) = 64 \quad & + \tfrac{1}{2} max\{0, 55 - 40, 55 - 40, 45 - 40\} \\ & + \tfrac{1}{2} max\{0, 45 - 40, 35 - 40, 35 - 40\} = 74. \end{aligned}$$

The Gantt chart in Figure 3.2 compares these two cases. In order to see the difference, the Gantt chart of one allocation case is drawn for two repetitions of the cycle. One can observe that the completion times of the first repetition of both cycles (bold dashed line in the figure) are the same but the completion

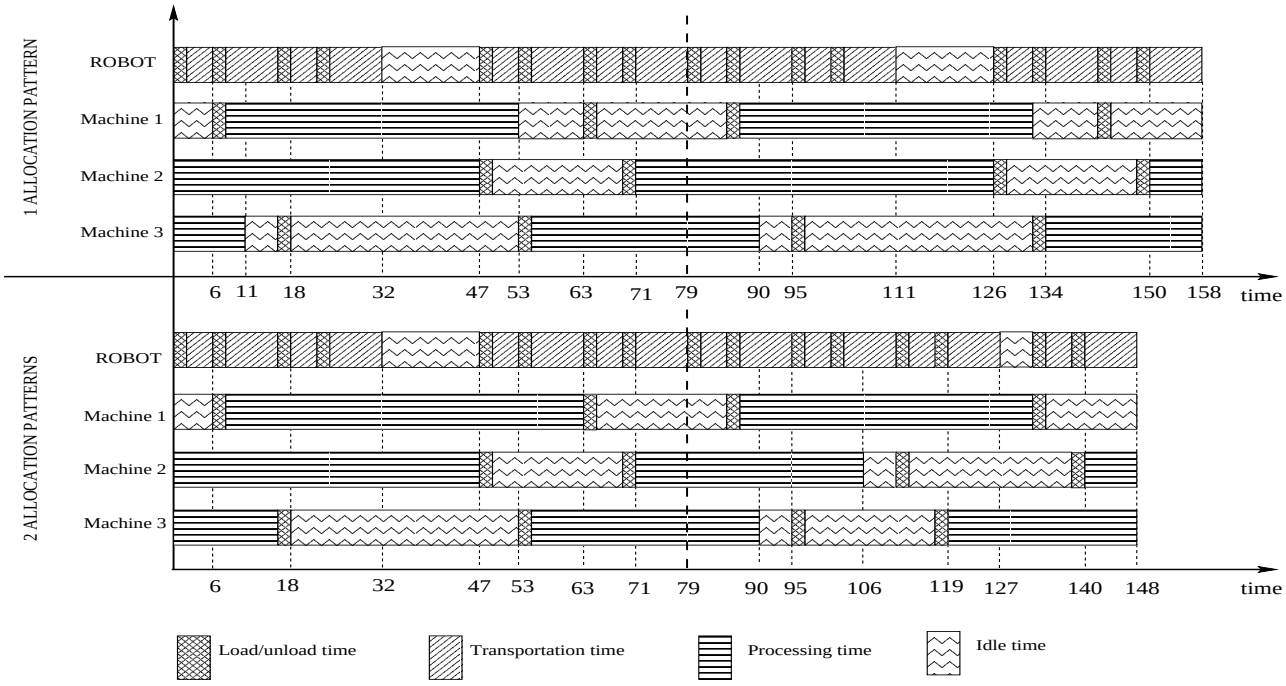

Figure 3.2: Gantt chart for example 3.1

times of the second repetition of the robot activities are different. In one allocation case the second repetition is exactly the same as the first repetition (which means the processing times on the machines are the same). However, for two different allocations case, the time of the second repetition is less than the first repetition because the total waiting time of the robot in front of the machines is reduced by 10 units. Then, in order to produce 1 part, this makes 5 units or 5/79=6.3% decrease in between the cycle times of these two cases. In fact, in this example using three different types of allocations is optimal. The optimal allocation matrix and the corresponding processing times are:

$$\Pi^* = \begin{bmatrix} \{1,2\} & \{4,5\} & \{3\} \\ \{3\} & \{1,2\} & \{4,5\} \\ \{4,5\} & \{3\} & \{1,2\} \end{bmatrix} \Rightarrow P_{\Pi^*} = \begin{bmatrix} 55 & 45 & 35 \\ 35 & 55 & 45 \\ 45 & 35 & 55 \end{bmatrix}.$$

The cycle time for this case is, $T_{S_6^3(\Pi_3^*)} = 70.67$. This corresponds to 8.33/79=10.5% decrease from the one allocation case. It is important to note

that this significant decrease in cycle time can be obtained with no additional cost, just by capturing the inherent flexibility of the CNC machines.

Chapter 4

Pure Cycles

In this chapter, we will assume that all of the machines are loaded with at least one copy of all of the required tools. As a consequence, each machine is capable of performing all of the operations of each part, and hence, each operation can be performed by any one of the machines. We will propose a new robot move cycle that fully utilizes the operational and process flexibility of CNC machines. This cycle will then be compared with the traditional robot move cycles present in the literature. Interestingly, this new cycle is used extensively in industry, not because it is proved to be optimal but because it is very practical, easy to understand and implement. We prove that this cycle is not only simple and practical but also dominates all classical robot move cycles when there exist two machines. For 3-machine cells we prove that the proposed cycle dominates all classical 1-unit cycles except one and all 2-unit cycles. For the general m-machine case, we provide the regions where the proposed cycle dominates the classical robot move cycles and for the remaining regions we analyze the worst case performance of the proposed cycle with respect to classical robot move cycles.

In the next section we will consider m-machine cells. In Section 4.2, we will analyze the 2- and 3-machine cells in further detail. Section 4.3 will conclude the chapter.

4.1 m-machine case

In this section we will analyze m-machine cells. The operational and process flexibilities of CNC machines allow the possibility of new cycles which necessitates definitions of new robot activities. Let,
A_{0i} = The robot activity in which the robot takes a part from the input buffer and loads machine $i = 1, 2, \ldots, m$.
$A_{i(m+1)}$ = The robot activity in which the robot unloads machine i and drops the part to the output buffer where $i = 1, 2, \ldots, m$.

In an m-machine robotic cell there are exactly $2m$ such activities. By using these activities we can define new cycles as follows:

Definition 4.1 *Under a* **pure cycle**, *starting with an initial state, the robot performs each of the* $2m$ *activities* $(A_{0i}, A_{i(m+1)}, \quad i = 1, \ldots, m)$ *exactly once and the final state of the system is identical with the initial state.*

Note that under these cycles all of the operations of each part are performed completely by one of the machines and between two loadings of any one of the machines, all other machines are loaded exactly once. One repetition of such a cycle produces m parts and in order to find the cycle time we divide the total time necessary to complete one repetition of this cycle with m. Each permutation of the $2m$ activities defines a pure cycle. However, some permutations define the same pure cycle. For example, in 2-machines case, $A_{01}A_{02}A_{13}A_{23}$ and $A_{13}A_{23}A_{01}A_{02}$ are different representations of the same

cycle. As a result, after eliminating the different representations, there exist a total of (2m-1)! different pure cycles in an m-machine cell. With this many different pure cycles, finding the best and later comparing it against all the classical flowshop type robot move cycles is extremely cumbersome and hence omitted from the scope of the current study. Instead, we focus on the simplest and most widely used pure cycle as a representative from this huge class. We prove that even this cycle dominates all classical robot move cycles for 2-machine cells and perform very well for the general $m-$machine cells. The proposed cycle is defined with the following activity sequence for m-machines:

Definition 4.2 $A_{01}A_{02}\dots A_{0m}A_{1(m+1)}A_{2(m+1)}\dots A_{m(m+1)}$*: The robot first loads machines 1 through m with a different part in respective order and each machine starts processing all of the operations of its loaded part. Then, the robot unloads machines 1 through m respectively. In order to unload machine i, the robot returns back to machine i, waits in front of the machine if the processing of the part is not finished, unloads the machine, transports the part to output buffer and drops the part.*

Let us consider Example 3.1 again. The cycle time of the proposed cycle for a 3-machine cell $A_{01}A_{02}A_{03}A_{14}A_{24}A_{34}$ with given parameters is $T = 69$, which is optimal for this example. This makes a 10/79=12.7% decrease from the best cycle time that can be found using the results reported in the literature.

4.1.1 Cycle time and lower bound calculations

In this subsection we will derive the cycle time of the proposed cycle and a lower bound for the classical robot move cycles. Let us first derive the cycle time of the proposed cycle. Assume the robot is idle at the input buffer at time 0. For $i = 1, \dots, m$, let T_i^{load} represent the time right after loading machine

i and T_i^{unload} represent the time the robot arrives at machine i for unloading. We set $D_i = T_i^{unload} - T_i^{load}$. Moreover, let w_i be the waiting time of the robot in front of machine i, i.e., $w_i = max\{0, P - D_i\}$. With our notation,

$$T_1^{load} = 2\epsilon + \delta,$$

since the robot takes a part from the input buffer (ϵ), transports to the first machine (δ) and loads this machine (ϵ). After the robot loads the $(i-1)^{st}$ machine, it moves to the input buffer ($(i-1)\delta$), takes a part (ϵ), transports to the i^{th} machine ($i\delta$) and loads this machine (ϵ). In other words,

$$T_i^{load} = T_{i-1}^{load} + (2i-1)\delta + 2\epsilon \quad \text{for} \quad i = 2, \ldots m.$$

Before arriving at the first machine for unloading, the robot loads the m^{th} machine and moves to the first one, i.e.,

$$T_1^{unload} = T_m^{load} + (m-1)\delta = 2m\epsilon + (m^2 + m - 1)\delta.$$

Before unloading machine i, the robot has to first unload machine $i-1$ $(T_{i-1}^{unload} + w_{i-1} + \epsilon)$, drop the part to output buffer $((m-i+2)\delta + \epsilon)$ and come back to machine i $((m-i+1)\delta)$. Hence for $i = 2, \ldots, m$,

$$T_i^{unload} = T_{i-1}^{unload} + (2m - 2i + 3)\delta + 2\epsilon + w_{i-1}.$$

Now, using the above relationships, it is easy to attain

$$D_1 = 2(m-1)\epsilon + (m^2 + m - 2)\delta \text{ and } D_i = D_{i-1} + (2m - 4i + 4)\delta + w_{i-1},$$

and that

$$D_i = D_1 + 2(i-1)(m-1)\delta + w_1 + \ldots w_{i-1} \text{ for } i = 2, \ldots, m.$$

Now, if $w_1 > 0$, in other words, $w_1 = P - D_1$ then $D_i \geq P$ for $i = 2, \ldots, m$ and therefore $w_2 = w_3 = \ldots w_m = 0$. If $w_1 = 0$, that is to say, $P \leq D_1$ then $P \leq D_i$ for $i = 2, \ldots, m$ as well since $D_1 \leq D_i$ and again we have

$w_2 = w_3 = \ldots = w_m = 0$. The total time to produce m parts with the proposed cycle is

$$T_m^{unload} + \epsilon + \delta + \epsilon + (m+1)\delta,$$

and after substituting for the easily calculated value of T_m^{unload}, this becomes

$$4m\epsilon + 2m(m+1)\delta + max\{0, P - 2(m-1)\epsilon - (m-1)(m+2)\delta\}.$$

Consequently, the cycle time of the proposed cycle with m machines is:

$$T_{proposed(m)} = 4\epsilon + 2(m+1)\delta + 1/m(max\{0, P - 2(m-1)\epsilon - (m-1)(m+2)\delta\}). \tag{4.1}$$

If the whole processing of a part can be done on a single machine, one can conjecture that there is no reason for performing a portion of it on a machine and the rest on another. In this way some load/unload time will be saved. Hence, the proposed cycle is always optimal and there is no reason to consider the robot move cycles derived under the assumption of a flow-shop type system. As we will see later in this chapter, this conjecture holds when the robot is the bottleneck, that is, when the total processing time of the parts is small relative to the load/unload time, ϵ and transportation time, δ. However, when the machines are bottleneck instead of the robot, that is, total processing time is large relative to the load/unload time, ϵ and transportation time, δ, the proposed cycle may result in higher cycle time values. If the processing time exceeds some value, then the average idle time of the machines waiting for some part to be loaded becomes greater in the proposed cycle. The following 3−machine example provides a situation of this kind:

Example 4.1 Let us assume that each part requires 6 operations with $t_1 = 40$, $t_2 = 45$, $t_3 = 50$, $t_4 = 60$, $t_5 = 50$, $t_6 = 55$ so the total processing time of each part $P = 300$. Also let $\epsilon = 2$ and $\delta = 10$. Consider the 1-unit cycle S_6^3 which

is defined by the following activity sequence: $A_0A_3A_2A_1$. The cycle time for this cycle is derived by Sethi et al. [86] as:

$$T_{S_6^3} = 8\epsilon + 12\delta + max\{0, a - 4\epsilon - 8\delta, b - 4\epsilon - 8\delta, b - 4\epsilon - 8\delta\},$$

where a, b and c are the processing times on M_1, M_2 and M_3 respectively. Let us consider the following allocation of operations: operations 1 and 4 are allocated to the first machine, $a = 100$, operations 2 and 6 are allocated to the second machine $b = 100$ and operations 3 and 5 are allocated to the last machine $c = 100$. Note that this allocation corresponds to a 1-allocation pattern and with our notation $a = P_{11}$, $b = P_{21}$ and $c = P_{31}$. The cycle time in this case is $T_{S_6^3} = 148$. On the other hand, using (4.1) with $m = 3$ and with the given data, the cycle time of the proposed cycle is 152.

This example shows that we cannot establish the dominance of the proposed cycle over the traditional robot move cycles for $m \geq 3$. However, the proposed cycle may not be the best pure cycle in this case. For example, consider $A_{01}A_{34}A_{03}A_{24}A_{02}A_{14}$. The cycle time of this cycle with the parameters of the above example turns out to be 129.33. However, there are 120 pure cycles in 3-machine cells and finding regions of optimality for these cycles is unnecessarily cumbersome. Hence, in the remainder we will only consider the proposed cycle and prove that even this cycle performs very efficiently.

The following theorem derives a lower bound for the cycle time of any robot move cycle in m-machine case for which the system is assumed to be flow-shop. Let $\underline{T_{flowshop}}$ represent the lower bound for these cycles. Note that, $\underline{T_{flowshop}} \leq min_{S,k}\{T_{S(\Pi_k^*)}\}$. That is, the lower bound is over all flowshop type robot move cycles, S's, and for the optimal allocations, Π_k^*, over all possible allocation periods, k.

Theorem 4.1 *For an m machine flow-shop type robotic cell, the cycle time of any n-unit cycle is no less than*

$$\underline{T_{fs(m)}} = max\{2(m+1)(\epsilon+\delta) + min\{P,\delta\}, 4\epsilon + 4\delta + (P/m)\}, \qquad (4.2)$$

where $\underline{T_{fs(m)}}$ *denotes the lower bound of the cycle time for an m-machine flow-shop type robotic cell.*

Proof. Geismar et al. (2005) derived the following lower bound for classical robot move cycles when there are no flexibilities, i.e. the allocation and the ordering of the operations are assumed to be fixed and known for each machine.

$$max\{2(m+1)(\epsilon+\delta) + \sum_{i=1}^{m} min\{P_i,\delta\}, 4\epsilon + 4\delta + max_i\{P_i\}\}, \qquad (4.3)$$

where P_i is the processing time on machine i. The reasoning behind the first argument of the *max* function in (4.3) is as follows: The robot loads and unloads all m machines exactly once ($2m\epsilon$), and also takes a part from the input buffer (ϵ), and drops a part to the output buffer (ϵ), in every cycle resulting in $2(m+1)\epsilon$. As the forward movement, the robot travels all the way from the input buffer to the output buffer in some sequence of robot activities which takes at least $(m+1)\delta$, and in order to return back to the initial state, the robot must travel back to the input buffer, taking at least $(m+1)\delta$. Additionally, note that each loading operation is followed by an unloading operation of either the same or a different machine. (Note that, taking a part from the input buffer is assumed to be an unloading operation). The summation term in the first argument of (4.3) represents the total time between all loading and the subsequent unloading operations. After loading a part to machine i, the robot has these options: it either waits in front of the machine to complete the processing of the part before unloading it (P_i), or travels to another machine to unload it or travels to input buffer to take another part. The minimum travel time from machine i to any other machine is

δ. Thus, for machine i, in order to find a lower bound we take the minimum of these two values and for all m machines this totals to $\sum_{j=1}^{m} min\{P_i, \delta\}$. With the assumptions of this study, the total robot travel time and load/unload time do not change. However, the sum term in (4.3) follows from the fact that the processing times on the machines are fixed. In this study, the processing times are not fixed but depend on allocation patterns, in other words, they are decision variables. For a cycle with k different allocation patterns where k is arbitrary, the cycle is repeated k times, each repetition with potentially different processing times. After loading a part to machine i, the robot either waits in front of the machine to finish processing (P_{ik}), or travels to another machine to unload it or to input buffer to take a part which takes at least δ time units. Hence, for all machines and for all repetitions of the cycle we have $\sum_{j=1}^{k} \sum_{i=1}^{m} min\{P_{ij}, \delta\}$. In order to find the lower bound to produce one part we must divide this by k. Furthermore, we know that $\sum_{i=1}^{m} P_{ij} = P$. As a consequence, $\sum_{i=1}^{m} min\{P_{ij}, \delta\} \geq min\{P, \delta\}$ for any allocation j . Then we have, $\sum_{j=1}^{k} \sum_{i=1}^{m} min\{P_{ij}, \delta\} \geq min\{P, \delta\}$. As a consequence, with the assumptions of this study, the first argument of the *max* function reduces to the following:

$$2(m+1)(\epsilon+\delta) + min\{P, \delta\}.$$

The reasoning behind the second argument of the *max* function in (4.3) is the following: The cycle time of any cycle is greater than the time between two consecutive loadings of a machine for which the consecutive loading time is the greatest. But in order to make a consecutive loading, the robot must at least perform the following activities: After loading a part to some machine i, the minimum time required before unloading this part is P_i. Then, the robot unloads machine i (ϵ), transports the part to machine $(i+1)$ (δ), loads it (ϵ), returns back to machine $(i-1)$ (2δ), unloads it (ϵ), transports the part to machine i (δ) and loads it (ϵ). This in total makes $4\epsilon + 4\delta + P_i$. In order to find the greatest consecutive loading time we take $max_i\{P_i\}$. However,

with the assumption of process and operational flexibilities, for a cycle with k different allocation patterns, where k is arbitrary, the longest processing time is $max_{i,j}(P_{ij})$. Since $\sum_{i=1}^{m} P_{ij} = P$, $\forall i$, we have $P/m \leq max_{i,j}(P_{ij})$. Hence, with the assumptions of this study the second argument of the *max* function reduces to $4\epsilon + 4\delta + P/m$. This completes the proof. □

4.1.2 Regions where the proposed cycle dominates the traditional robot move cycles

The number of pure cycles increases drastically as the number of machines increases, thus finding the best pure cycle is a huge enumerative task for $m \geq 3$. Henceforth, we will only compare the proposed cycle with the classical robot move cycles. Recall that the proposed cycle is a direct consequence of assuming the machines to be CNC machines which are loaded with at least one copy of each of the required tools. Since it is easy to control and implement, such a cycle is preferred in the industry to more complex cycles, even if it is not the provably optimal robot move cycle. With the following theorem, we find the regions where the proposed cycle dominates in cycle time the traditional robot move cycles for m-machine robotic cells.

Theorem 4.2 *In comparison with the traditional robot move cycles, the proposed cycle is the best if* $(m-2)\delta \leq 2\epsilon \quad or \quad P \leq 2(m^2-1)\epsilon+(m^2+2m-2)\delta$.

Proof. In order to prove this theorem, we will compare the cycle time of the proposed cycle with the lower bound value of the traditional robot move cycle times. Let us first recall that:
$\underline{T_{fs(m)}} = max\{2(m+1)(\epsilon+\delta) + min\{P,\delta\}, 4\epsilon + 4\delta + (P/m)\}$, and
$T_{proposed(m)} = 4\epsilon + 2(m+1)\delta + 1/m(max\{0, P - 2(m-1)\epsilon - (m-1)(m+2)\delta\})$.

Note that both of these are piecewise linear functions of P and can be rewritten as follows:

$$\underline{T_{fs(m)}} = \begin{cases} 2(m+1)(\epsilon+\delta)+P, & \text{if } P \le \delta, \\ 2(m+1)\epsilon+(2m+3)\delta, & \text{if } \delta < P \le 2m(m-1)\epsilon + m(2m-1)\delta, \\ 4\epsilon+4\delta+P/m, & \text{if } P > 2m(m-1)\epsilon + m(2m-1)\delta. \end{cases} \tag{4.4}$$

$$T_{proposed(m)} = \begin{cases} 4\epsilon+2(m+1)\delta, & \text{if } P \le 2(m-1)\epsilon + (m-1)(m+2)\delta, \\ 1/m(2(m+1)\epsilon+(m^2+m+2)\delta+P), & \text{if } P > 2(m-1)\epsilon + (m-1)(m+2)\delta. \end{cases} \tag{4.5}$$

Clearly: $\delta \le 2(m-1)\epsilon + (m-1)(m+2)\delta \le 2m(m-1)\epsilon + m(2m-1)\delta$. Hence, it is sufficient to consider the following cases:

1. If $0 \le P \le \delta$, then the cycle time of the proposed cycle is:
   $T_{proposed(m)} = 4\epsilon + 2(m+1)\delta$.
   The lower bound of the cycle times of traditional robot move cycles is:
   $\underline{T_{fs(m)}} = 2(m+1)\epsilon + 2(m+1)\delta + P$.
   Clearly, $T_{proposed(m)} \le \underline{T_{fs(m)}}$.

2. If $\delta < P \le 2(m-1)\epsilon + (m-1)(m+2)\delta$, then
   $T_{proposed(m)} = 4\epsilon + 2(m+1)\delta \le 2(m+1)\epsilon + (2m+3)\delta = \underline{T_{fs(m)}}$.

3. If $2(m-1)\epsilon + (m-1)(m+2)\delta < P \le 2m(m-1)\epsilon + m(2m-1)\delta$, then
   $T_{proposed(m)} = 1/m(2(m+1)\epsilon + (m^2+m+2)\delta + P)$,
   and $\underline{T_{fs(m)}} = 2(m+1)\epsilon + (2m+3)\delta$.
   When we compare these two values with each other we see that $T_{proposed(m)} \le \underline{T_{fs(m)}} \iff P \le 2(m^2-1)\epsilon + (m^2+2m-2)\delta$ which is one of the conditions in the statement of our theorem. Recall that,

in this region $P \leq 2m(m-1)\epsilon + m(2m-1)\delta$. If $(m-2)\delta \leq 2\epsilon$, then $P \leq 2m(m-1)\epsilon + m(2m-1)\delta \leq 2(m^2-1)\epsilon + (m^2+2m-2)\delta$. As a result if $(m-2)\delta \leq 2\epsilon$ and $P \leq 2m(m-1)\epsilon + m(2m-1)\delta$, then $T_{proposed(m)} \leq \underline{T_{fs(m)}}$.

4. If $P > 2m(m-1)\epsilon + m(2m-1)\delta$, then
   $T_{proposed(m)} = 1/m(2(m+1)\epsilon + (m^2+m+2)\delta + P)$,
   and $\underline{T_{fs(m)}} = 4\epsilon + 4\delta + P/m$.
   Comparing these two, one can show that for $(m-2)\delta \leq 2\epsilon$, $T_{proposed(m)} \leq \underline{T_{fs(m)}}$.

□

Outside these regions, we can provide a worst case performance bound of the proposed cycle with respect to the traditional robot move cycles. In particular:

Lemma 4.1 *In the region where* $(m-2)\delta > 2\epsilon$ *and* $P > 2(m^2-1)\epsilon + (m^2+2m-2)\delta$, *the cycle time of the proposed cycle,* $T_{proposed(m)} < C \cdot T^*$, *where* $C = 1 + \frac{m^2-3m+2}{m^2+6m-2}$ *and* T^* *is the optimal cycle time among the traditional robot move cycles.*

Proof. For $(m-2)\delta > 2\epsilon$ and $P > 2(m^2-1)\epsilon + (m^2+2m-2)\delta$, from (4.4), $T_{fs(m)} \geq 4\epsilon+4\delta+P/m$ and from (4.5), $T_{proposed(m)} = 1/m(2(m+1)\epsilon+(m^2+m+2)\delta + P)$. Hence, we can derive a worst case performance bound for using the proposed cycle instead of the best flowshop type robot move cycle as follows: Let T^* be the optimal cycle time among the traditional robot move cycles in this region.

$$\frac{T_{proposed(m)}}{T^*} \leq \frac{1/m(2(m+1)\epsilon + (m^2+m+2)\delta + P)}{1/m(4m\epsilon + 4m\delta + P)}$$

$$= \frac{2(m+1)\epsilon + (m^2+m+2)\delta + P}{4m\epsilon + 4m\delta + P}$$
$$= 1 + \frac{-2(m-1)\epsilon + (m-1)(m-2)\delta}{4m\epsilon + 4m\delta + P} .$$

Since $P > 2(m-1)(m+1)\epsilon + (m^2+2m-2)\delta$, we have:

$$\frac{T_{proposed(m)}}{T^*} < 1 + \frac{-2(m-1)\epsilon + (m-1)(m-2)\delta}{(2m^2+4m-2)\epsilon + (m^2+6m-2)\delta} .$$

Let $\delta = \alpha\epsilon$ where $\alpha > 2/(m-2)$:

$$\frac{T_{proposed(m)}}{T^*} < 1 + \frac{(m-1)(m-2)\alpha - 2(m-1)}{(m^2+6m-2)\alpha + (2m^2+4m-2)} .$$

The right hand side gets larger as α tends to infinity (loading/unloading time is negligible when compared with robot transportation time). Hence, the bound converges asymptotically to the following:

$$\frac{T_{proposed(m)}}{T^*} < \lim_{\alpha\to\infty}(1+\frac{(m-1)(m-2)\alpha - 2(m-1)}{(m^2+6m-2)\alpha + (2m^2+4m-2)}) = 1+\frac{m^2-3m+2}{m^2+6m-2} .$$

□

For $m = 2$ the worst case bound is 1 and for $m \to \infty$ the asymptotic bound is 2. As a consequence, the worst case bound takes values between 1 and 2 with respect to m. For example when $m = 4$ the worst case bound becomes $1 + 6/38 \approx 1.158$.

In the next section we will focus on the 2- and 3-machine cases and show the dominance of the proposed cycle over the traditional robot move cycles.

4.2 2- and 3-machine cells

In this section we will compare the cycle times of the proposed cycle and the traditional robot move cycles for 2- and 3-machine cells. Let us first consider the 2-machine case.

Using (4.1), the cycle time of the proposed cycle with $m = 2$ becomes:

$$T_{proposed(2)} = 4\epsilon + 6\delta + 1/2max\{0, P - (2\epsilon + 4\delta)\}. \quad (4.6)$$

and using (4.2), the lower bound for the traditional robot move cycles becomes:

$$\underline{T_{fs(2)}} = max\{6\epsilon + 6\delta + min\{P, \delta\}, 4\epsilon + 4\delta + P/2\}. \quad (4.7)$$

The next theorem will establish an important contribution.

Theorem 4.3 *The proposed robot move cycle $A_{01}A_{02}A_{13}A_{23}$ gives the minimum cycle time for 2-machine identical parts robotic cell scheduling problem with process and operational flexibility.*

Proof. A simple comparison of equations (4.6) and (4.7) for $P \in [0, \delta]$, $P \in (\delta, 2\epsilon + 4\delta]$, $P \in (2\epsilon + 4\delta, 4\epsilon + 6\delta]$ and $P \in (4\epsilon + 6\delta, \infty)$ yields $T_{proposed(2)} \leq \underline{T_{fs(2)}}$. □

Note that the proposed cycle is not necessarily the best pure cycle. However, Theorem 4.3 proves that even this cycle dominates all of the classical robot move cycles. In a 2-machine cell there are 6 pure cycles, $C1$ through $C6$, for which the activity sequences and the cycle time values are presented in Appendix A. The following theorem compares the pure cycles with each other and determines the regions of optimality.

Theorem 4.4 *If $P < 2\epsilon + 4\delta$ then $C1$ is optimal, if $P > 2\epsilon + 4\delta$ then $C6$ is optimal, if $P = 2\epsilon + 4\delta$ then both $C1$ and $C6$ perform equally well.*

Proof. Observing the cycle times of the cycles presented in Appendix A, one can easily conclude that $C1$ dominates $C2$, $C3$, $C4$ and $C5$. A simple

comparison of the cycle times of $C1$ and $C6$ concludes the proof. □

Now let us consider 3-machine cells. Using (4.1), the cycle time of the proposed cycle with $m = 3$ becomes:

$$T_{proposed(3)} = 4\epsilon + 8\delta + 1/3max\{0, P - (4\epsilon + 10\delta)\}. \tag{4.8}$$

Recall that in a 3-machine cell there are six feasible 1-unit cycles which can be listed as follows:

$$S_1^3 = (A_0A_1A_2A_3), \quad S_2^3 = (A_0A_2A_1A_3), \quad S_3^3 = (A_0A_1A_3A_2),$$
$$S_4^3 = (A_0A_3A_1A_2), \quad S_5^3 = (A_0A_2A_3A_1), \quad S_6^3 = (A_0A_3A_2A_1).$$

The lower bound of the classical robot move cycles found in Theorem 4.1 becomes the following for 3-machine robotic cells:

$$\underline{T_{flowshop}} = max\{8(\epsilon + \delta) + min\{P, \delta\}, 4\epsilon + 4\delta + (P/3)\}. \tag{4.9}$$

The forthcoming corollary to Theorem 4.2 provides the regions where the proposed cycle is the best for 3-machine cells.

Corollary 4.1 *If $\delta \leq 2\epsilon$ or $P \leq 16\epsilon + 13\delta$, then the proposed cycle gives the minimum cycle time for 3-machine cells.*

Now let us consider the region where the lower bound of the flowshop type robot move cycles is less than the cycle time of the proposed robot move cycle. That is, $\delta > 2\epsilon$ and $P > 16\epsilon + 13\delta$. First we concentrate on the 1-unit robot move cycles since they are simple, practical, easy to understand and provably optimal for 3-machine flowshop type systems. The following lemma is very useful in reducing the number of potentially optimal robot move cycles:

Lemma 4.2 *The proposed cycle dominates all flowshop type 1-unit cycles except* S_6^3.

Proof. Let us consider each 1-unit cycle one by one:
$\mathbf{S_1^3}$: For the cycle S_1^3, whatever the allocation of the operations is, the cycle time is the same. The cycle time derived by Sethi et al. [86] is:

$$T_{S_1^3(\Pi_k)} = 8\epsilon + 8\delta + P.$$

As it is seen, the cycle time does not depend on the allocation. When we compare this cycle time with the cycle time of the proposed cycle given in (4.8), $T_{proposed(3)} < T_{S_1^3(\Pi_k)}$. Thus we conclude that the proposed cycle dominates S_1^3.
$\mathbf{S_2^3}$: Let us derive the cycle time of the cycle S_2^3 considering the assumptions of this study. Consider an arbitrary allocation matrix Π_k and the i^{th} repetition of this cycle. Initially the second machine is loaded with a part having allocation type $(i-1)$ and the robot is in front of the input buffer. The robot takes a part from the input buffer and loads it to the first machine, $(2\epsilon + \delta)$, moves to second machine, waits if necessary for the machine to finish the processing of the part with allocation type $(i-1)$, $(\delta + w_{(i-1)2})$, unloads the second machine and loads the third machine, $(2\epsilon + \delta)$, moves to the first machine, waits if necessary for the machine to finish the processing of the part with allocation type i, $(2\delta + w_{i1})$, unloads the first machine and loads the second machine, $(2\epsilon + \delta)$, moves to the third machine and waits if necessary for the part with allocation type $(i-1)$, $(\delta + w_{(i-1)3})$, unloads the machine and drops the part to the output buffer, $(2\epsilon + \delta)$, returns back to input buffer, (4δ). Hence the time for the i^{th} repetition of the cycle S_2^3 with allocation matrix Π_k becomes:

$$8\epsilon + 12\delta + w_{i1} + w_{(i-1)2} + w_{(i-1)3}.$$

Let us denote $max\{0, a\}$ as $(a)^+$. With this notation,
$w_{i1} = (P_{i1} - 4\epsilon - 8\delta - w_{(i-1)2})^+$, $w_{(i-1)2} = (P_{(i-1)2} - 2\epsilon - 4\delta - w_{(i-2)3})^+$ and

$w_{(i-1)3} = (P_{(i-1)3} - 2\epsilon - 4\delta - w_{i1})^+$ are the waiting times in front of the machines 1, 2 and 3, respectively. For all k repetitions, we have the following:

$$T_{S_2^3(\Pi_k)} = 8\epsilon + 12\delta + 1/k(\sum_{i=1}^{k}(w_{i1} + w_{(i-1)2} + w_{(i-1)3})).$$

Using the fact that $a \leq (a)^+$ we get:

$$\begin{aligned}\textstyle\sum_{i=1}^{k}(w_{i1} + w_{(i-1)2} + w_{(i-1)3}) \geq \ & \textstyle\sum_{i=1}^{k}(P_{i1} + P_{(i-1)2} + P_{(i-1)3} - 8\epsilon - 16\delta) \\ & \textstyle- \sum_{i=1}^{k}(w_{i1} + w_{(i-1)2} + w_{(i-2)3}).\end{aligned}$$

Since for $r = 0, 1, \ldots$ the allocation of every $(rk + i)^{th}$ part in the infinite sequence is identical, then

$$\sum_{i=1}^{k}(P_{i1} + P_{(i-1)2} + P_{(i-1)3} - 8\epsilon - 16\delta) = k(P - 8\epsilon - 16\delta),$$

and

$$\sum_{i=1}^{k}(w_{i1} + w_{(i-1)2} + w_{(i-1)3}) = \sum_{i=1}^{k}(w_{i1} + w_{(i-1)2} + w_{(i-2)3}) = W,$$

where W is defined to be the total waiting time in front of the three machines for the k parts produced according to the allocation matrix Π_k. This yields;

$$2W \geq k(P - 8\epsilon - 16\delta) \Rightarrow W \geq k/2(P - 8\epsilon - 16\delta).$$

Thus for S_2^3 we have:

$$T_{S_2^3(\Pi_k)} \geq 8\epsilon + 12\delta + 1/k(k/2(P - 8\epsilon - 16\delta)) = 1/2(8\epsilon + 8\delta + P). \quad (4.10)$$

$\mathbf{S_3^3}$: Using the above procedure, the time for the i^{th} repetition of S_3^3 is:

$$8\epsilon + 10\delta + P_{i1} + w_{i2} + w_{(i-1)3},$$

where $w_{i2} = (P_{i2} - 2\epsilon - 4\delta - w_{(i-1)3})^+$, and $w_{(i-1)3} = (P_{(i-1)3} - 4\epsilon - 6\delta - P_{i1})^+$ are the waiting times in front of machines 2 and 3, respectively. Then we have the following:

$P_{i1} + w_{i2} + w_{(i-1)3} \geq P_{i1} + P_{i2} - 2\epsilon - 4\delta - w_{(i-1)3} + P_{(i-1)3} - 4\epsilon - 6\delta - P_{i1}$.
This yields:
$2(P_{i1} + w_{i2} + w_{(i-1)3}) \geq 2(P_{i1} + w_{(i-1)3}) + w_{i2} \geq P_{i1} + P_{i2} + P_{(i-1)3} - 6\epsilon - 10\delta$.
For all k repetitions we have the following:

$$2\sum_{l=1}^{k}(P_{i1} + w_{i2} + w_{(i-1)3}) \geq \sum_{l=1}^{k}(P_{i1} + P_{i2} + P_{(i-1)3} - 6\epsilon - 10\delta).$$

Let W be as defined previously. Then we can write the following:

$$2W \geq P - 6\epsilon - 10\delta \Rightarrow W \geq 1/2(P - 6\epsilon - 10\delta).$$

Then for S_3^3 we have the following:

$$T_{S_3^3(\Pi_k)} \geq 8\epsilon + 10\delta + 1/2(P - 6\epsilon - 10\delta) = 1/2(P + 10\epsilon + 10\delta). \tag{4.11}$$

$\mathbf{S_4^3}$: Total time for the i^{th} repetition of S_4^3 is the following:

$$8\epsilon + 12\delta + w_{i1} + P_{i2} + w_{(i-1)3},$$

where $w_{i1} = (P_{i1} - 2\epsilon - 6\delta - w_{(i-1)3})^+$ and $w_{(i-1)3} = (P_{(i-1)3} - 2\epsilon - 6\delta)^+$ are the waiting times in front of machines 1 and 3, respectively. A similar procedure that we used for S_3^3 yields, $W \geq 1/2(P - 4\epsilon - 12\delta)$ and

$$T_{S_4^3(\Pi_k)} \geq 1/2(12\epsilon + 12\delta + P). \tag{4.12}$$

$\mathbf{S_5^3}$: Total time for the i^{th} repetition of S_5^3 is the following:

$$8\epsilon + 10\delta + w_{i1} + w_{(i-1)2} + P_{(i-1)3},$$

where $w_{i1} = (P_{i1} - 4\epsilon - 6\delta - w_{(i-1)2} - P_{(i-1)3})^+$, and $w_{(i-1)2} = (P_{(i-1)2} - 2\epsilon - 4\delta)^+$ are the waiting times in front of the machines 1 and 2, respectively. From here we get, $W \geq 1/2(P - 6\epsilon - 10\delta)$ and the lower bound for S_5^3 becomes:

$$T_{S_5^3(\Pi_k)} \geq 1/2(10\epsilon + 10\delta + P). \tag{4.13}$$

Comparing the lower bounds for the cycles S_2^3, S_3^3, S_4^3 and S_5^3, given in equations (4.10), (4.11), (4.12) and (4.13) respectively we get the following:

$$\underline{T_{S_2^3(\Pi_k)}} < \underline{T_{S_3^3(\Pi_k)}} = \underline{T_{S_5^3(\Pi_k)}} < \underline{T_{S_4^3(\Pi_k)}},$$

where $\underline{T_{S_j^3(\Pi_k)}} = min_{\Pi_k}\{T_{S_j^3(\Pi_k)}\}$. Let us compare $\underline{T_{S_2^3(\Pi_k)}} = 1/2(P + 8\epsilon + 8\delta)$ with the cycle time of the proposed cycle given in (4.8):

$$1/2(8\epsilon + 8\delta + P) = 1/2(1/3(24\epsilon + 24\delta + 3P)).$$

Since in this region $P > 16\epsilon + 13\delta$,

$$\underline{T_{S_2^3(\Pi_k)}} = 1/6(24\epsilon + 24\delta + 2P + P) > 1/6(40\epsilon + 37\delta + 2P)$$

$$= 1/3(20\epsilon + (18.5)\delta + P) > 1/3(8\epsilon + 14\delta + P) = T_{proposed}.$$

Thus we can conclude that the proposed cycle dominates S_2^3, S_3^3, S_4^3 and S_5^3.

$\mathbf{S_6^3}$: Example 4.1 shows that S_6^3 cannot be dominated by the proposed robot move cycle. □

1-unit cycles are important because they are simple, practical and easy to understand. Also if the system is assumed to be a flowshop then 1-unit cycles are provably optimal for 2-machine cells ([86]) and 3-machine cells ([20]). However, Akturk et al. [2] proved that with the assumption of operational flexibility, even in 2-machines case, a 2-unit cycle can result in smaller cycle times than the 1-unit robot move cycles for some parameter ranges. This motivates us to consider the 2-unit cycles. Hall et al. [43] derived the activity sequences of all feasible 2-unit cycles in a 3-machine robotic cell. In Appendix B we present a completely new procedure to derive the activity sequences of these cycles and list them. This new procedure utilizes the fact that all 2-unit cycles are made up from two 1-unit cycles. That is, let S_i^3 and S_j^3 be two different 1-unit cycles. Then, in a 2-unit cycle, S_{ij}^3 is simply a combination of

S_i^3 and S_j^3; during some part of the cycle the robot follows the activity sequence of S_i^3 and during the remaining part of the cycle the robot follows the activity sequence of S_j^3.

The following lemma derives a general lower bound, $\underline{T_2}$, for all of the 2-unit robot move cycles with any allocation matrix Π_k.

Lemma 4.3 $\underline{T_2} = 1/2(P + 8\epsilon + 8\delta)$ *where* $\underline{T_2} \leq min_{S_{ij}^3,k}\{T_{S_{ij(\Pi_k^*)}^3}\}$.

Proof. For the clarity of the presentation, we refer the reader to Appendix C for the proof. □

The following lemma proves that the proposed cycle dominates all flowshop type 2-unit robot move cycles.

Lemma 4.4 *The proposed cycle dominates all flowshop type 2-unit cycles.*

Proof. With Corollary 4.1 we assert that the proposed cycle gives the minimum cycle time for $P \leq 16\epsilon + 13\delta$. Now let us consider the region where $P > 16\epsilon + 13\delta$. In this region the cycle time of the proposed cycle given in (4.8) becomes $4\epsilon + 8\delta + 1/3(P - 4\epsilon - 10\delta)$ which can be rewritten as $1/3(P + 8\epsilon + 14\delta)$. When we compare this cycle time with the lower bound we found in Lemma 4.3, we have the following:

$$\underline{T_2} = 1/2(P + 8\epsilon + 8\delta) = 1/6(3P + 24\epsilon + 24\delta) \geq 1/6(2P + 40\epsilon + 37\delta)$$

$$= 1/3(P + 20\epsilon + (18.5)\delta) > 1/3(P + 8\epsilon + 14\delta) = T_{proposed}.$$

This completes the proof. □

Until now we considered all the 1 and 2-unit cycles and showed that the proposed cycle dominates all except the 1-unit cycle S_6^3. Knowing that the

proposed cycle dominates all 2-unit cycles, one can conjecture that it also dominates 3 and higher unit cycles. Proving or disproving this conjecture is not so simple because the number of feasible robot move cycles increases drastically as n, the number of units produced in one cycle increases and deriving and comparing these cycles with the proposed cycle become quite complex. Additionally, the proposed cycle is simple, practical and easy to implement. Furthermore, there is no allocation problem to be solved for this cycle. More importantly, the worst case bound of the proposed cycle found in Lemma 4.1 becomes 1.08 for 3-machine robotic cells. As a result of these observations, we conclude that what little improvement we might attain (if any) by considering 3 and higher unit cycles will not be sufficient enough to justify the effort that will be spent for this purpose.

4.3 Concluding Remarks

In this chapter, we considered a new class of robot move cycles, called the pure cycles, resulting from the flexibility of the CNC machines. Since there is a huge number of such cycles in an m-machine robotic cell, we proposed one of the cycles among this class which is extensively used in industry due to its simplicity in understanding and implementation. We proved in Theorem 4.3 that this cycle, in fact, dominates the traditional robot move cycles for $m = 2$. With Theorem 4.2 we found the regions where the proposed cycle dominates the traditional robot move cycles for $m \geq 3$. In order to prove this theorem, we compared the proposed cycle with the lower bound of the classical robot move cycles. For the remaining regions we proved that the proposed cycle dominates all of the 1-unit robot move cycles except S_6^3 and all of the 2-unit robot move cycles in 3-machine cells. We also found a worst case performance bound of the proposed cycle with respect to the traditional robot move cycles for

the remaining regions. Furthermore, with the reduced cycle times (increased throughput), our results enable the justification of additional tool inventories that will be incurred when loading a copy of every required tool to both of the machines (this might also necessitate a larger tool magazine). As a final remark, in the new move cycle each part is loaded and unloaded only once, which means less gaging; probably one of the important reasons why this cycle is preferred in practice. An extended version of this chapter is accepted for publication [39].

Chapter 5

Cell Design

In this chapter we will consider certain design problems arising in the context of robotic manufacturing cells. In Section 5.1, we will consider different layouts of the cell in order to improve the efficiency of the cell. In Section 5.2, we will determine the number of machines which minimizes the cycle time for given parameters such as the processing times of the operations, load/unload times of the machines, ϵ and robot transportation time, δ. Section 5.3 is devoted to the concluding remarks.

5.1 Layout analysis

Till now we have assumed an in-line robotic cell layout (IRC). For this layout we proved that if we assume operational and process flexibility, a new cycle gives better results than all of the common cycles reported in the literature. At this point we will consider changing the layout of the cell to a robot centered one (RCC) as shown in Figure 5.1. Although Han and Cook [47] stated that the layout analysis can improve efficiency of the cells, classical robotic cell

scheduling literature does not compare the cycle times of the robot move cycles with IRC and RCC layouts. This is due to the fact that for the common cycles reported in the literature both layout types give the same cycle time assuming that for both type of cell layouts, the robot transportation time between two adjacent machines is fixed as δ and assumed to be additive. In the robot centered cell layout, the travel time from input buffer to machine 1 or machine 2 is δ and the travel time from machine 1 to machine 2 is equivalent to the summation of travel times from machine 1 to input buffer (or output buffer) (δ) and from input buffer (output buffer) to machine 2 (δ) which makes 2δ. The travel times for the IRC and RCC layouts are different from each other. For example, travel time from machine 1 to machine 2 is δ in the IRC layout whereas it is 2δ in the RCC layout. As a consequence, the cycle time of the proposed cycle will be different for these two layouts. In the following theorem we compare the cycle times of the proposed cycle with IRC and RCC layouts and prove that the cycle time with RCC layout is less than the cycle time with IRC layout.

Theorem 5.1 *For 2-machine robotic cells, the cycle time of the proposed cycle with RCC layout is less than the cycle time with IRC layout.*

Proof. First let us derive the cycle time of the proposed cycle with the RCC layout: Initially the machines are empty and the robot is in front of the input buffer. The robot takes a part (ϵ), transports to the first machine (δ), loads it (ϵ), returns back to input buffer (δ), takes another part (ϵ), transports to the second machine (δ), loads it (ϵ), returns back to the first machine (2δ), waits if necessary for the machine to finish the processing of the part (w_1), unloads the machine (ϵ), transports the part to output buffer (δ), drops it (ϵ), moves back to second machine (δ), waits if necessary (w_2), unloads the machine (ϵ), transports the part to the output buffer (δ), drops it (ϵ) and returns back to the

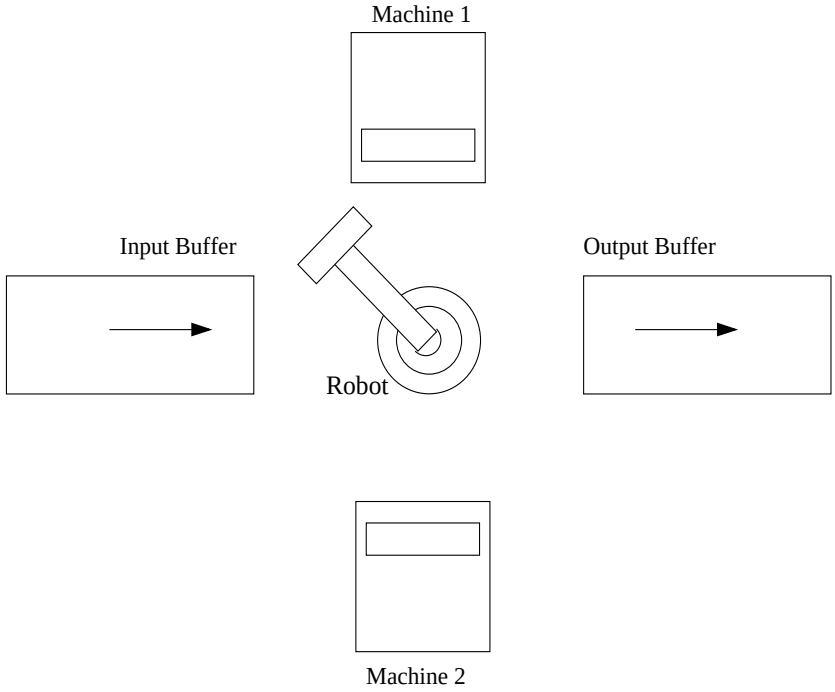

Figure 5.1: Robot centered cell layout

input buffer (2δ). Note that during one cycle two parts are produced. Thus, in order to find the cycle time we divide the total time by 2 which makes:

$$T_{proposed(RCC)} = 4\epsilon + 5\delta + w_1 + w_2,$$

where $w_1 = max\{0, P - (2\epsilon + 4\delta)\}$ and $w_2 = max\{0, P - (2\epsilon + 4\delta + w_1)\}$ and hence

$w_1 + w_2 = max\{w_1, P - (2\epsilon + 4\delta)\} = max\{0, P - (2\epsilon + 4\delta)\}$.

Consequently, the cycle time of the proposed cycle with the RCC layout is:

$$T_{proposed(RCC)} = 4\epsilon + 5\delta + 1/2max\{0, P - (2\epsilon + 4\delta)\}.$$

On the other hand, the cycle time for the proposed cycle with the IRC layout is given in (4.6). After a simple comparison we conclude that changing the layout proves to be favorable for the proposed cycle. □

The above theorem is important since in many practical applications, robot centered cells are used simply because particular type of cellular layout requires

less space than an in-line robotic cell layout. Furthermore, stationary base robots (as in robot centered cells) are cheaper to install and easier to program and consequently more robust than the mobile robots. In the next section we will consider the m-machine case.

5.2 Determining the optimal number of machines for the proposed robot move cycle

In the previous chapter, we studied the operational problem of determining the robot move sequences for a given number of machines. Now let us consider the number of machines as a decision variable and try to find the optimal number of machines which minimizes the cycle time for given parameters ϵ, δ and P. The cycle time for the proposed cycle for the most general m-machine case is given in (4.1). In the following Lemma we show that this function is convex with respect to m.

Lemma 5.1 *The cycle time of the proposed cycle given in (4.1) is convex with respect to m.*

Proof. We can rewrite this function as follows:

$$max\{2m\delta + 4\epsilon + 2\delta, 1/m(m^2\delta + 2m\epsilon + m\delta + P + 2\epsilon + 2\delta)\},$$

which is equivalent to the following:

$$max\{2m\delta + 4\epsilon + 2\delta, m\delta + 2\epsilon + \delta + 1/m(P + 2\epsilon + 2\delta)\}. \tag{5.1}$$

The first argument of the above max function is linear with respect to m. The second argument is a summation of two convex functions: $m\delta + 2\epsilon + \delta$ and

$1/m(P + 2\epsilon + 2\delta)$ (note that $m > 0$). Thus, it is also convex. Finally, the maximum of two convex functions is also a convex function. □

Let a be a real number. We will denote the largest integer smaller than or equal to a with $\lfloor a \rfloor$. The following theorem determines the optimal number of machines given the parameters ϵ, δ and P.

Theorem 5.2 *The optimal number of machines, m^*, is one of the two integers, $\lfloor 1/2\delta(-2\epsilon - \delta + \alpha) \rfloor$ or $\lfloor 1/2\delta(-2\epsilon - \delta + \alpha) \rfloor + 1$, where $\alpha = \sqrt{4\epsilon^2 + 12\epsilon\delta + 9\delta^2 + 4\delta P}$.*

Proof. We are trying to minimize a function of m of the form $f(m) = max\{g(m), h(m)\}$, where, $g(m) = 2m\delta + 4\epsilon + 2\delta$ and $h(m) = m\delta + 2\epsilon + \delta + 1/m(P + 2\epsilon + 2\delta)$. Let m^* denote the minimizer of $f(m)$. Then, m^* satisfies at least one of the following: m^* is a minimizer of $g(m)$, it is a minimizer of $h(m)$ or $g(m^*) = h(m^*)$. Let us consider each of these cases:

1. $g(m)$ is a linear increasing function and is minimized for $m = 0$. However, $h(m)$ tends to ∞ for $m \to 0$. Since $f(m)$ takes the maximum of $g(m)$ and $h(m)$, the minimizer of $g(m)$ can not be a minimizer of $f(m)$.

2. $h(m)$ is a convex continuous function for $m > 0$.
   $\frac{\partial h(m)}{\partial m} = 0 \Rightarrow \delta - 1/m^2(2\epsilon + 2\delta + P) = 0 \Rightarrow \hat{m} = \sqrt{1/\delta(2\epsilon + 2\delta + P)}$.
   However, at this point:
   $g(\hat{m}) = 2\delta\sqrt{1/\delta(2\epsilon + 2\delta + P)} + 4\epsilon + 2\delta > 2\delta\sqrt{1/\delta(2\epsilon + 2\delta + P)} + 2\epsilon + \delta = h(\hat{m})$.
   Hence, the minimizer of $h(m)$ can not be a minimizer of $f(m)$.

3. Hence, we can conclude that the minimizer of (5.1) is at the intersection point of the two arguments of the max function which is found as follows:
   $$g(m) = h(m) \Rightarrow 2m\delta + 4\epsilon + 2\delta = m\delta + 2\epsilon + \delta + 1/m(2\epsilon + 2\delta + P)$$

$$\Rightarrow m^2\delta + (2\epsilon + \delta)m - 2\epsilon - 2\delta - P = 0.$$

We can find the roots of this equation by using the discriminant. There are two roots one of which is less than 0. But since we consider the region where $m > 0$ we take the nonnegative root as the solution of this equation.

$$m = 1/2\delta(-2\epsilon - \delta + \sqrt{4\epsilon^2 + 4\epsilon\delta + \delta^2 + 4\delta(2\epsilon + 2\delta + P)})$$

$$= 1/2\delta(-2\epsilon - \delta + \alpha),$$

where $\alpha = \sqrt{4\epsilon^2 + 12\epsilon\delta + 9\delta^2 + 4\delta P}$. This is a real number. However, m represents the number of machines which means it must be an integer. From Lemma 5.1, the function is convex with respect to m. As a consequence, in order to find the best integer value we have to consider both sides of the real number. That is, the largest integer smaller than $1/2\delta(-2\epsilon - \delta + \alpha)$ and the smallest integer larger than this number. The best integer value is one of the following: $\lfloor 1/2\delta(-2\epsilon - \delta + \alpha)\rfloor$ or $\lfloor 1/2\delta(-2\epsilon - \delta + \alpha)\rfloor + 1$ where α is defined as before. In order to find which one of these two gives the minimum cycle time value, we evaluate Equation (5.1) at these two integer values and take the one which gives the minimum cycle time value.

□

As a result of this theorem, with given parameters such as the total processing time, loading/unloading time and robot transportation time we can easily determine the optimal number of machines to be placed inside a robotic cell that minimizes the cycle time.

5.3 Concluding Remarks

Till now the robotic cell scheduling literature considered the operational problems such as determining the robot move cycle and/or the part input sequence in order to optimize some objective function. In this chapter, rather than an operational problem we considered some design problems and showed that the efficiency of the cells can be improved by solving such design problems. We first considered the layout of the cell as a design problem and in Theorem 5.1 we proved that RCC layout is preferable to IRC layout in the 2-machine case when using the pure cycles. As a second design problem we considered the number of machines to be placed inside a cell as a decision variable and in Theorem 5.2 we determined the optimal number of machines to be used for given parameter values. This study initiates a new research direction for the robotic cell scheduling literature. An extended version of this chapter is accepted for publication [40].

Chapter 6

Tooling Constraints

In this chapter we consider a 2-machine robotic cell. In an ideal FMS, each machine is capable of performing all operations of all parts scheduled for production as long as it has the required tools in its tool magazine. However, Gray et al. [37] observe that a CNC has a limited tool magazine capacity and the total set of tools required to process all jobs usually exceeds this capacity. Furthermore, duplicating all the required tools and loading them to each tool magazine may not be economically justifiable due to high tool investment costs. Therefore, in this chapter, we assume that there is a single copy of some tools. A subset of these single copies is loaded on the first machine and the remaining ones are loaded on the second machine. On the other hand, some tools are duplicated and loaded on both machines. As a consequence, each part to be processed has three sets of operations. O_1 is the set of operations that can only be processed on the first machine, O_2 is the set of operations that can only be processed on the second machine, and O is the set of operations that can be processed on either machine. Then the problem is not only sequencing the robot's activities but also partitioning the set of flexible operations into two machines. The objective is to minimize the cycle time. As a consequence of the

tooling constraints, we will assume that each part being processed goes through the input buffer, the first machine, the second machine and finally the output buffer in that order. In the next section we introduce some new definitions, notations and the assumptions pertaining to this chapter. In Section 6.2 we will reduce the number of potentially optimal robot move cycles to three and we will find the regions of optimality for these robot move cycles according to the given parameters such as the loading and unloading time ϵ and the robot transportation time δ. The last section is devoted to the concluding remarks.

6.1 Problem Definition

Recall that for two machines, we have two 1-unit robot move cycles: S_1^2: $A_0A_1A_2$ and S_2^2: $A_2A_1A_0$. The processing time of a part on a machine depends on the allocation of the operations to the machines. As a consequence, although the parts are assumed to be identical, their processing times on the machines may be different than each other. Hence, 1-unit cycles need not be optimal in all regions. In order to represent higher unit cycles we need the following observations. Let us recall that a state of the system is defined by whether the robot and the machines are loaded or empty and by the location of the robot. These two 1-unit cycles have one common state in which the first machine is empty and the second machine has just been loaded by the robot. Thus, a transition from one of these cycles to the other can only be made at this common state. If the robot waits in front of the second machine to finish the processing of the part, then the robot follows the activities of the S_1^2 cycle. Else if the robot travels to input buffer to take a part and load the first machine, then the robot follows the activities of the S_2^2 cycle. During these transitions no extra movements are made, thus no loss occurs in terms of robot transportation time.

Hall et al. [43] define two other robot move sequences to represent higher unit robot move cycles. These are the transition movements of the robot from performing cycle S_1^2 to S_2^2 (represented as S_{12}) and S_2^2 to S_1^2 (represented as S_{21}). If a part is produced according to the S_1^2 (S_2^2) cycle in the first machine and according to the S_2^2 (S_1^2) cycle in the second machine then the transition movement S_{12} (S_{21}) is made. To properly clarify these new robot move sequences, we shall use the terminology of Dawande et al. [24]. *Full waiting* is defined as the robot waiting in front of a machine through the whole processing time of a part. On the other hand, *partial waiting* is the waiting time from the arrival of the robot at the machine till the processing of the part completes at this machine. Now, we are ready to list the S_{12} robot movements which can be described as $A_0A_1A_0$: (i) load the first machine and perform a full wait, (ii) load the second machine and immediately return to input buffer, (iii) load the first machine. Furthermore, the only activity that can follow this activity sequence is to travel to the second machine while processing continues on the first machine and perform a partial wait.

In a similar fashion, in S_{21} movement which can be described as $A_2A_1A_2$, the robot unloads the second machine, drops the part to the output buffer and returns back to the first machine to unload the part. After a partial waiting, unloads this machine and loads the second machine. Performs a full wait, unloads the machine and drops the part to output buffer. That is, a partial waiting on the first machine and a full waiting on the second machine are encountered.

Further analysis of the sequences yields the following: (i) an execution of S_1^2 starts and ends with empty machines, (ii) an execution of S_2^2 starts and ends with loaded machines, (iii) an execution of S_{12} starts with empty machines and ends with full machines, and (iv) an execution of S_{21} starts with loaded machines and ends with empty machines. Based on these observations,

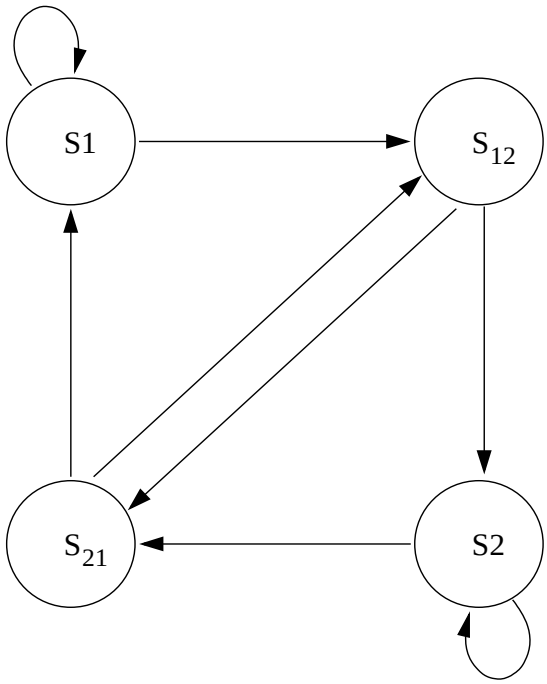

Figure 6.1: Transition digraph

the transition digraph in Figure 6.1 depicts the feasible transitions among sequences.

Hall et al. [43] showed that any robot move cycle can be represented by the four robot move sequences: S_1^2, S_2^2, S_{12} and S_{21}. For example, $S_{12}S_{21}$ is a 2-unit robot move cycle, actually it is the only 2-unit robot move cycle and we can describe this cycle by the robot activity sequence: $A_0A_1A_0A_2A_1A_2$. $S_{12}S_{21}$ and $S_{21}S_{12}$ represent the same robot move cycle, the only difference being the starting state of the cycle.

We will use the following additional notation throughout this chapter.

P^{M1} : Total processing time of the operations that are in set O_1, $P^{M1} = \sum_{l \in O_1} t_l$.

P^{M2} : Total processing time of the operations that are in set O_2, $P^{M2} = \sum_{l \in O_2} t_l$.

P : Total processing time of the operations that are to be allocated to either

one of the machines, $P = \sum_{l \in O} t_l$.

Note that if set O is empty, that is when $P = 0$, there is no allocation problem. As a consequence, the problem becomes identical with the problem considered by Sethi et al. [86].

6.2 Solution Procedure

In order to find the optimal robot move cycle, we have to compare the cycle times of the robot move cycles. The cycle times depend on the allocation of the operations. In the next subsection we will determine the optimal allocation types for three robot move cycles and prove that according to given parameter values one of these three robot move cycles is optimal. In Subsection 6.2.2 we will determine the regions of optimality for each of the three robot move cycles and Subsection 6.2.3 is devoted to the sensitivity analysis on problem parameters.

6.2.1 Optimal Allocation of Operations

In this section we will first observe that there is no allocation problem for S_1^2. Theorems 6.1 and 6.3 will determine the optimal k to be used with the cycles S_2^2 and $S_{12}S_{21}$, respectively. In Theorem 6.2 we will prove that determining the optimal allocation of operations to the machines is NP-complete for robot move cycle S_2^2. We shall assume $P^{M1} \geq P^{M2}$ throughout all proofs. The other case can be treated in a similar fashion.

The cycle times of the three robot move cycles discussed previously are presented in Appendix D. The cycle time of S_1^2, given in equation (D.1) does

not depend on the processing times of the parts on the machines individually but only depends on the total processing time, in our case $P + P^{M1} + P^{M2}$. Thus, regardless of the allocation of the operations, the cycle time is the same and hence there is no allocation problem for S_1^2.

The following theorem provides the optimal k to be used with S_2^2:

Theorem 6.1 *Consider a cyclic production performing the 1-unit cycle S_2^2. We have:*

1. *If $P^{M1} \geq P + P^{M2}$, then using one-allocation is optimal,*
2. *Otherwise, using either one-allocation or two-allocation is optimal.*

Proof.

1. The total time to complete production of all k parts with k-allocation type for cycle S_2^2 under specific allocation matrix Π_k, is given in Appendix D, equation (D.4). Since $P^{M1} \geq P + P^{M2}$, then $P^{M1} + P_{i1} \geq P^{M2} + P_{(i-1)2}$, $\forall i \in [1, \ldots, k]$ where for notational purposes we take $P_{01} \equiv P_{k1}$. Therefore, under optimal allocation matrix Π_k^*, we have $P_{i1}^* = 0$, $\forall i$. However, this is nothing but one-allocation type and the optimal cycle time is:
$$T_{S_2^2(\Pi^*)} = 6\epsilon + 8\delta + max\{0, P^{M1} - (2\epsilon + 4\delta)\}$$
2. Consider again (D.4). Since $P^{M1} < P + P^{M2}$, under optimal allocation matrix Π_k^*, each max term will individually be minimized when the last two nonzero components are as close as possible, i.e., when $P^{M1} + P_{i1}^* \approx P^{M2} + P_{(i-1)2}^*$, $\forall i$. Consider two such consecutive relations, say for i^{th} and $(i+1)^{th}$ max terms:
$$P^{M1} + P_{i1}^* \approx P^{M2} + P_{(i-1)2}^* \tag{6.1}$$

$$P^{M1} + P^*_{(i+1)1} \approx P^{M2} + P^*_{i2} \tag{6.2}$$

From relations 6.1 and 6.2, if the equalities cannot be satisfied, then the difference between both sides of each equality must be minimized. That is, both $|P^{M1} + P^*_{i1} - (P^{M2} + P^*_{(i-1)2})|$ and $|P^{M1} + P^*_{(i+1)1} - (P^{M2} + P^*_{i2})|$ must be minimized. Note however that $P^*_{i2} = P - P^*_{i1}$ and $P^*_{(i+1)1} = P - P^*_{(i+1)2}$. Plugging in these two values and arranging the terms both $|P^{M1} + P^*_{i1} - (P^{M2} + P^*_{(i-1)2})|$ and $|P^{M1} + P^*_{i1} - (P^{M2} + P^*_{(i+1)2})|$ must be minimized. This yields either a one-allocation ($P^*_{(i-1)1} = P^*_{i1} = P^*_{(i+1)1}$, $\forall i$) or a two-allocation ($P^*_{(i-1)1} = P^*_{(i+1)1}$ and $P^*_{i1} \neq P^*_{(i-1)1}$, $\forall i$). □

Though Theorem 6.1 guides us in selecting the optimal allocation type in a cyclic production S^2_2, finding the optimal allocation of operations to the two machines is not an easy job even when there is a fixed allocation type and even when $P^{M1} = P^{M2} = 0$. More formally, we shall now show that the following decision problem is NP-complete.

S_2 Operation Allocation for one-allocation Type Decision Problem (Problem S2TAP):

Instance: A finite set of operations O with respective integer processing times $\{t_1, \ldots, t_p\}$, loading/unloading time ϵ, transportation time δ and a real number K.

Question: Can we find an allocation matrix for operations to the two machines, Π_1, so that the long run average cycle time $T_{S^2_2(\Pi_1)} \leq K$?

Theorem 6.2 *Problem S2TAP is NP-complete.*

Proof. S2TAP is in NP since whenever we are given a specific allocation of operations, we can readily find the corresponding long run average cycle time

and hence decide if it is less than or equal to K. We will show that S2TAP is NP-complete by reducing the 2-Partition problem to it. As a reminder the 2-Partition problem can be stated as (see [28]):

> *Instance:* Given $A = \{a_1, \ldots, a_r\}$, a_i integer and $s(a_i) \in Z^+$ size of item i.
>
> *Question:* Is there a partition of A into $A^{'}$ and $A \setminus A^{'}$ ($A^{'} \subseteq A$) such that $\sum_{i \in A'} s(i) = \sum_{i \in A \setminus A'} s(i)$?

Suppose we have an arbitrary instance of 2-Partition. From this we are going to construct a specific instance of S2TAP and show that S2TAP has a solution if and only if 2-Partition instance has a solution. Let $A = a_1, \ldots, a_r$, $s(a_i)$ $i \in [1, \ldots, r]$ be the given instance of 2-Partition. We shall have r operations in our set O each corresponding to an item from the given set A. Each t_i will have processing time $s(a_i)$. Let $\epsilon = (\sum_{i \in [1,\ldots,r]} s(a_i))/8$ and $\delta = (\sum_{i \in [1,\ldots,r]} s(a_i))/16$.

Claim: S2TAP has a solution with these specifications and $K = \frac{7}{4} \sum_{i \in [1,\ldots,r]} s(a_i) \Leftrightarrow$ 2-Partition instance has a yes answer.

For one-allocation type, $T_{S_2^2(\Pi_1)} = 6\epsilon + 8\delta + max\{0, P_{11} - (2\epsilon + 4\delta), P_{12} - (2\epsilon + 4\delta)\}$ and $T_{S_2^2(\Pi_1)} \geq 6\epsilon + 8\delta = \frac{7}{4} \sum_{i \in [1,\ldots,r]} s(a_i)$. With given ϵ and δ values, the cycle time for a one-allocation S_2^2 becomes $T_{S_2^2(\Pi_1)} = 6\epsilon + 8\delta + max\{0, P_{11} - P/2, P_{12} - P/2\}$. Thus, the minimum cycle time $7/4P$ is attained if and only if $P_{11} = P_{12} = P/2$, if and only if 2-Partition has a yes answer. □

The following theorem determines the optimal k to be used with $S_{12}S_{21}$.

Theorem 6.3 *For 2-unit robot move cycle $S_{12}S_{21}$, using two-allocation types is optimal.*

Proof. In order to prove this theorem, we will first compare the cycle time of two-allocation type for the cycle $S_{12}S_{21}$ with the cycle time of one-allocation type for the cycle $S_{12}S_{21}$. The long run average cycle times of one-allocation and two-allocation types for the cycle $S_{12}S_{21}$ are given in equations (D.5) and (D.6), respectively.

We first argue that the optimal allocation for two-allocation type for the cycle $S_{12}S_{21}$ is found by letting $P_{11}^* = P_{22}^* = 0$. Let us first rewrite (D.6) by entering $\frac{1}{2}(P_{11} + P_{22})$ into the max term, i.e., adding it to all the three arguments of max. This leads us to the following form:

$$T_{S_{12}S_{21}(\Pi_2)} = 1/2(12\epsilon + 14\delta + P^{M1} + P^{M2})$$
$$+1/2(max\{P_{11} + P_{22}, P^{M1} + P_{11} + P - (2\epsilon + 4\delta), P^{M2} + P + P_{22} - (2\epsilon + 4\delta)\}).$$

Now, P_{11} and P_{22} only appear with positive coefficients. To minimize, we must decrease both of these. So, we take $P_{11}^* = P_{22}^* = P$. Since we also have $P^{M1} \geq P^{M2}$ the cycle time corresponding to this allocation type becomes:

$$T_{S_{12}S_{21}(\Pi_2^*)} = \frac{12\epsilon + 14\delta + P^{M1} + P^{M2} + max\{0, (P + P^{M1}) - (2\epsilon + 4\delta)\}}{2}. \tag{6.3}$$

Let us move P in equation (D.5) inside the max term. Thus we have:

$$\begin{aligned} T_{S_{12}S_{21}(\Pi_1^*)} = \;& 1/2(12\epsilon + 14\delta + P^{M1} + P^{M2}) \\ + \;& 1/2(max\{P, P + P^{M1} + P_{11}^* - (2\epsilon + 4\delta), P + P^{M2} + P_{12}^* - (2\epsilon + 4\delta)\}). \end{aligned}$$

Comparing this with equation (6.3), it is easily seen that $T_{S_{12}S_{21}(\Pi_2^*)} \leq T_{S_{12}S_{21}(\Pi_1^*)}$.

Now let us consider the cycle time for k-allocation ($k > 2$) type for the cycle $S_{12}S_{21}$. Observe that, in robot move cycle $S_{12}S_{21}$, initially, the machines are empty and the robot is waiting in front of the input buffer. Since this is a 2-unit cycle, exactly two parts are loaded and unloaded to each machine and

at the end, identical to the initial state, the machines are again empty and the robot is waiting in front of the input buffer. Then, in a k-allocation type for the cycle $S_{12}S_{21}$, this 2-unit cycle is repeated exactly $k/2$ times to produce all k parts with different allocation types. The objective is to find the optimal allocation for these parts to the machines. However, the optimal allocation at each repetition of the cycle must be the same as the optimal allocation of the operations for a unique cycle $S_{12}S_{21}$. This is because, a k-allocation type is a concatenation of $k/2$ two-allocation types for the cycle $S_{12}S_{21}$. Thus, we conclude that the optimal cycle time for a k-allocation type for the cycle $S_{12}S_{21}$ is the same as the optimal cycle time for a two-allocation type for the cycle $S_{12}S_{21}$. □

Now, we are ready to provide one of the major results of our paper which will restrict our search for the optimal cycle to three robot move cycles. We first recall a result due to Hall et al. [43].

Lemma 6.1 *(Hall et al. [43]) In any feasible robot move cycle, the number of S_{12} sequences is equal to the number of S_{21} sequences.*

Theorem 6.4 *At least one of the cycles S_1^2, S_2^2 or $S_{12}S_{21}$ has a cycle time that is less than or equal to the cycle time of any given n-unit robot move cycle.*

For the clarity of the presentation, this proof is deferred to Appendix E.

In summary, we have three potentially optimal robot move cycles. For the robot move cycle S_2^2, we have an allocation problem. However, we know from Theorem 6.1 that if $P^{M1} \geq P + P^{M2}$, then the allocation problem disappears and we have $T_{S_2^2(\Pi^*)} = 6\epsilon+8\delta+max\{0, P^{M1}-(2\epsilon+4\delta)\}$. Else, we can find lower and upper bounds for this cycle time. We get a lower bound when it is possible to have $P^{M1} + P_{i1} = P^{M2} + P_{i2}$, $i \in \{1, 2\}$. Since any feasible solution provides

an upper bound for the optimal cycle time, we will present a feasible solution that can be attained with any given problem instance and use this feasible solution as an upper bound. For a given problem instance, let us allocate all of the operations that are in set O to the first machine for the first part and to the second machine for the second part. Note that such an allocation is feasible with any given problem parameter values. Thus, $P_{11} = P_{22} = P$ in equation (D.3). Let $\underline{T_{S_2^2}}$ represent the lower bound of cycle S_2^2 and $\overline{T_{S_2^2}}$ represent the upper bound. In other words:

$$\underline{T_{S_2^2}} = 6\epsilon + 8\delta + max\{0, (P + P^{M1} + P^{M2})/2 - (2\epsilon + 4\delta)\},$$

$$\overline{T_{S_2^2}} = 6\epsilon + 8\delta + \frac{max\{0, P + P^{M1} - (2\epsilon + 4\delta), P + P^{M2} - (2\epsilon + 4\delta)\}}{2} + \frac{max\{0, P^{M1} - (2\epsilon + 4\delta), P^{M2} - (2\epsilon + 4\delta)\}}{2}.$$

Since we assumed that $P^{M1} \geq P^{M2}$ the upper bound becomes:

$$\overline{T_{S_2^2}} = 6\epsilon + 8\delta + \frac{1}{2}max\{0, P + P^{M1} - (2\epsilon + 4\delta)\} + \frac{1}{2}max\{0, P^{M1} - (2\epsilon + 4\delta)\}.$$

Now we are ready to present the optimal S_2^2 cycle times along with lower and upper bounds based on a set of breakpoints partitioning the search space.

1- If $P^{M1} \geq P + P^{M2}$, then

1.1- If $P^{M1} \geq 2\epsilon + 4\delta$, then

$$T_{S_2^2(\Pi^*)} = 4\epsilon + 4\delta + P^{M1}. \tag{6.4a}$$

1.2- Otherwise,

$$T_{S_2^2(\Pi^*)} = 6\epsilon + 8\delta. \tag{6.4b}$$

2- Else (i.e. $P^{M1} < P + P^{M2}$) we have lower and upper bounds. The lower bounds are as follows:

i- If $(P + P^{M1} + P^{M2})/2 \geq 2\epsilon + 4\delta$, then

$$\underline{T_{S_2^2}} = 4\epsilon + 4\delta + (P + P^{M1} + P^{M2})/2. \tag{6.4c}$$

ii- Otherwise,

$$\underline{T_{S_2^2}} = 6\epsilon + 8\delta. \tag{6.4d}$$

The upper bounds are as follows:

i- If $P^{M1} \geq 2\epsilon + 4\delta$, then

$$\overline{T_{S_2^2}} = 4\epsilon + 4\delta + P^{M1} + P/2. \tag{6.4e}$$

ii- Else if $P + P^{M1} < 2\epsilon + 4\delta$, then

$$\overline{T_{S_2^2}} = 6\epsilon + 8\delta. \tag{6.4f}$$

iii- Else,

$$\overline{T_{S_2^2}} = 5\epsilon + 6\delta + (P + P^{M1})/2. \tag{6.4g}$$

On the other hand, the cycle time of two-allocation type for the cycle $S_{12}S_{21}$ is given in equation (D.6). Since $P^{M1} \geq P^{M2}$, employing Theorem 6.3, we get the following cycle time:

$$T_{S_{12}S_{21}(opt)} = 6\epsilon + 7\delta + \frac{(P^{M1} + P^{M2})}{2} + \frac{1}{2}max\{0, (P^{M1} + P) - (2\epsilon + 4\delta)\}.$$

Therefore, we have the following breakpoints for this cycle time:

1- If $P + P^{M1} \geq 2\epsilon + 4\delta$ then,

$$T_{S_{12}S_{21}(\Pi^*)} = 5\epsilon + 5\delta + P^{M1} + \frac{(P + P^{M2})}{2}\ . \tag{6.5a}$$

2- Else,

$$T_{S_{12}S_{21}(\Pi^*)} = 6\epsilon + 7\delta + \frac{(P^{M1} + P^{M2})}{2}\ . \tag{6.5b}$$

6.2.2 Regions of Optimality

In the sequel, we will prove a sequence of lemmas which will jointly lead to Theorem 6.5 presenting the regions of optimality for these three robot move cycles.

Lemma 6.2 *If $P^{M1} + P^{M2} \geq 2\delta$, then S_2^2 gives the minimum cycle time.*

Proof. Assume $P^{M1} + P^{M2} \geq 2\delta$. Let us first compare $T_{S_1^2}$ and $T_{S_{12}S_{21}(\Pi^*)}$. $T_{S_1^2}$ is as given in equation (D.1).

1. If $P + P^{M1} \geq 2\epsilon + 4\delta$, $T_{S_{12}S_{21}(\Pi^*)}$ isas given in equation (6.5a). A simple comparison yields $T_{S_{12}S_{21}(\Pi^*)} < T_{S_1^2}$.

2. Otherwise, $T_{S_{12}S_{21}(\Pi^*)}$ is given by equation (6.5b). Then we have the following:

   $$T_{S_1^2} = 6\epsilon+6\delta+P+P^{M1}+P^{M2} \geq 6\epsilon+6\delta+P+\frac{(P^{M1}+P^{M2})}{2}+\frac{2\delta}{2} \geq T_{S_{12}S_{21}(\Pi^*)}$$

   $$\Rightarrow T_{S_{12}S_{21}(\Pi^*)} \leq T_{S_1^2}.$$

We will now compare $T_{S_2^2(\Pi^*)}$ with $T_{S_{12}S_{21}(\Pi^*)}$.

1- If $P^{M1} \geq P + P^{M2}$, then

   1.1- If $P^{M1} \geq 2\epsilon+4\delta$, this implies that $P+P^{M1} \geq 2\epsilon+4\delta$. Then, $T_{S_2^2(\Pi^*)}$ and $T_{S_{12}S_{21}(\Pi^*)}$ are given by equations (6.4a) and (6.5a) respectively. Hence, we conclude that in this region $T_{S_2^2(\Pi^*)} < T_{S_{12}S_{21}(\Pi^*)}$.

   1.2- If $P^{M1} < 2\epsilon+4\delta$ and $P+P^{M1} \geq 2\epsilon+4\delta$, then $T_{S_{12}S_{21}(\Pi^*)}$ is the same as above. $T_{S_2^2(\Pi^*)}$ in this region is given in equation (6.4b). Since

$P + P^{M1} \geq 2\epsilon + 4\delta$ and $P^{M1} + P^{M2} \geq 2\delta$ we have:

$$\begin{aligned} T_{S_{12}S_{21}(\Pi^*)} &= 5\epsilon + 5\delta + \tfrac{P+P^{M1}+P^{M1}+P^{M2}}{2} \\ &\geq 5\epsilon + 5\delta + \tfrac{2\epsilon+4\delta+2\delta}{2} = 6\epsilon + 8\delta = T_{S_2^2(\Pi^*)} \end{aligned}$$

$\Rightarrow T_{S_2^2(\Pi^*)} \leq T_{S_{12}S_{21}(\Pi^*)}$.

1.3- If $P + P^{M1} < 2\epsilon + 4\delta$, $T_{S_2^2(\Pi^*)}$ and $T_{S_{12}S_{21}(\Pi^*)}$ are presented in equations (6.4b) and (6.5b), respectively. Since $P^{M1} + P^{M2} \geq 2\delta$ we have:

$$T_{S_{12}S_{21}(\Pi^*)} = 6\epsilon + 7\delta + \frac{(P^{M1} + P^{M2})}{2} \geq 6\epsilon + 8\delta = T_{S_2^2(\Pi^*)}$$

$\Rightarrow T_{S_2^2(\Pi^*)} \leq T_{S_{12}S_{21}(\Pi^*)}$.

2- Else if $P^{M1} < P + P^{M2}$, then we have upper and lower bounds for the cycle time of S_2^2. If we can show that $\overline{T_{S_2^2}} \leq T_{S_{12}S_{21}(\Pi^*)}$, then we can conclude that $T_{S_2^2(\Pi^*)} \leq T_{S_{12}S_{21}(\Pi^*)}$. We have the following cases:

2.1- If $P^{M1} \geq 2\epsilon + 4\delta$, this implies that $P + P^{M1} \geq 2\epsilon + 4\delta$. $\overline{T_{S_2^2}}$ and $T_{S_{12}S_{21}(\Pi^*)}$ are given in equations (6.4e) and (6.5a), respectively. We conclude easily that $T_{S_2^2(\Pi^*)} \leq \overline{T_{S_2^2}} < T_{S_{12}S_{21}(\Pi^*)}$.

2.2 If $P^{M1} < 2\epsilon+4\delta$ and $P+P^{M1} \geq 2\epsilon+4\delta$, $\overline{T_{S_2^2}}$ and $T_{S_{12}S_{21}(\Pi^*)}$ are given in equations (6.4g) and (6.5a), respectively. Since $P^{M1}+P^{M2} \geq 2\delta$, we have:

$$\begin{aligned} T_{S_{12}S_{21}(\Pi^*)} &= 5\epsilon + 5\delta + \tfrac{P+P^{M1}+P^{M1}+P^{M2}}{2} \\ &\geq 5\epsilon + 5\delta + \tfrac{P+P^{M1}+2\delta}{2} = 5\epsilon + 6\delta + \tfrac{(P+P^{M1})}{2} = \overline{T_{S_2^2}} \end{aligned}$$

$\Rightarrow T_{S_2^2(\Pi^*)} \leq \overline{T_{S_2^2}} \leq T_{S_{12}S_{21}(\Pi^*)}$.

2.3 If $P + P^{M1} < 2\epsilon + 4\delta$, this implies that $P^{M1} < 2\epsilon + 4\delta$. For this case, $\overline{T_{S_2^2}}$ and $T_{S_{12}S_{21}(\Pi^*)}$ are given in equations (6.4f) and (6.5b), respectively. Since $P^{M1} + P^{M2} \geq 2\delta$ we conclude easily that for this case also $T_{S_2^2(\Pi^*)} \leq \overline{T_{S_2^2}} \leq T_{S_{12}S_{21}(\Pi^*)}$.

Since in all of the cases $\overline{T_{S_2^2}} \leq T_{S_{12}S_{21}(\Pi^*)}$, we conclude that S_2^2 has the minimum cycle time. □

Lemma 6.3 *If* $2P + P^{M1} + P^{M2} \leq 2\delta$, *then* S_1^2 *gives the minimum cycle time.*

Proof. $2P + P^{M1} + P^{M2} \leq 2\delta \Rightarrow P + P^{M1} < 2\epsilon + 4\delta$. $T_{S_{12}S_{21}(\Pi^*)}$ is given in equation (6.5b). When we compare this with $T_{S_1^2}$ given in equation (D.1), since $2P + P^{M1} + P^{M2} \leq 2\delta$, we have:

$$\begin{aligned} T_{S_1^2} &= 6\epsilon + 6\delta + P + P^{M1} + P^{M2} = 6\epsilon + 6\delta + \tfrac{2P+P^{M1}+P^{M2}+P^{M1}+P^{M2}}{2} \\ &\leq 6\epsilon + 7\delta + \tfrac{(P^{M1}+P^{M2})}{2} = T_{S_{12}S_{21}(\Pi^*)} \end{aligned}$$

$\Rightarrow T_{S_1^2} \leq T_{S_{12}S_{21}(\Pi^*)}$.

Now we will compare $T_{S_1^2}$ with $T_{S_2^2(\Pi^*)}$. We have the following cases:

1- Assume $P^{M1} \geq P + P^{M2}$. Since $2P + P^{M1} + P^{M2} \leq 2\delta$, then $P^{M1} < 2\epsilon + 4\delta$. Using $T_{S_2^2(\Pi^*)}$ as given in equation (6.4b) we have:

$T_{S_1^2} = 6\epsilon + 6\delta + P + P^{M1} + P^{M2} \leq 6\epsilon + 6\delta + 2P + P^{M1} + P^{M2} \leq 6\epsilon + 8\delta$

$\Rightarrow T_{S_1^2} \leq T_{S_2^2(\Pi^*)}$.

2- If $P^{M1} < P + P^{M2}$, since $2P + P^{M1} + P^{M2} \leq 2\delta$, then $(P + P^{M1} + P^{M2})/2 < 2\epsilon + 4\delta$. $\underline{T_{S_2^2}}$ for this region is given in equation (6.4d). The comparison in Case 1 above is valid for this case also. Hence, we conclude that $T_{S_1^2} \leq T_{S_2^2(\Pi^*)}$. □

Lemma 6.4 *If* $P^{M1} + P^{M2} < 2\delta$, $2P + P^{M1} + P^{M2} > 2\delta$ *and* $P + 2P^{M1} + P^{M2} \leq 2\epsilon + 6\delta$, *then* $S_{12}S_{21}$ *gives the minimum cycle time.*

Proof. Since $P^{M1} + P^{M2} < 2\delta$, then $P^{M1} < 2\epsilon + 4\delta$. If $P^{M1} \geq P + P^{M2}$ then $T_{S_2^2(\Pi^*)}$ is given in equation (6.4b), which is $6\epsilon+8\delta$. If $P^{M1} < P+P^{M2}$, we have to show that $T_{S_{12}S_{21}(\Pi^*)} \leq \underline{T_{S_2^2}}$. Since $P + P^{M1} + P^{M2} \leq P + 2P^{M1} + P^{M2} \leq 2\epsilon + 6\delta$, then $(P + P^{M1} + P^{M2})/2 \leq \epsilon + 3\delta < 2\epsilon + 4\delta$. $\underline{T_{S_2^2}}$ for this region is given in equation (6.4d), which is also $6\epsilon + 8\delta$. Thus, in both cases we will compare $T_{S_{12}S_{21}(\Pi^*)}$ with $6\epsilon + 8\delta$. We have the following cases:

1- If $P + P^{M1} \geq 2\epsilon + 4\delta$, $T_{S_1^2}$ and $T_{S_{12}S_{21}(\Pi^*)}$ are given in equations (D.1) and (6.5a), respectively. Observing these cycle times we conclude that $T_{S_{12}S_{21}(\Pi^*)} < T_{S_1^2}$. In order to compare the cycle times of S_2^2 and $S_{12}S_{21}$ we have:

$$T_{S_{12}S_{21}(\Pi^*)} = 5\epsilon + 5\delta + \frac{P + 2P^{M1} + P^{M2}}{2} \leq 5\epsilon + 5\delta + \frac{2\epsilon + 6\delta}{2} = 6\epsilon + 8\delta$$

$$\Rightarrow T_{S_{12}S_{21}(\Pi^*)} \leq T_{S_2^2(\Pi^*)}.$$

2- Otherwise, $T_{S_{12}S_{21}(\Pi^*)}$ is given in equation (6.5b). Since $2P+P^{M1}+P^{M2} > 2\delta$, we have:

$$\begin{aligned} T_{S_1^2} &= 6\epsilon + 6\delta + \tfrac{2P+P^{M1}+P^{M2}}{2} + \tfrac{P^{M1}+P^{M2}}{2} \\ &> 6\epsilon + 6\delta + 2\delta/2 + \tfrac{(P^{M1}+P^{M2})}{2} = T_{S_{12}S_{21}(\Pi^*)} \end{aligned}$$

$$\Rightarrow T_{S_{12}S_{21}(\Pi^*)} < T_{S_1^2}.$$

When we compare the cycle time of S_2^2 with the cycle time of $S_{12}S_{21}$, since $P^{M1} + P^{M2} < 2\delta$, using equation (6.5b) we have:

$$T_{S_{12}S_{21}(\Pi^*)} = 6\epsilon + 7\delta + (P^{M1} + P^{M2})/2 < 6\epsilon + 7\delta + 2\delta/2 = 6\epsilon + 8\delta$$

$$\Rightarrow T_{S_{12}S_{21}(\Pi^*)} < T_{S_2^2(\Pi^*)}.$$ □

Lemma 6.5 *If $P^{M1}+P^{M2} < 2\delta$, $2P+P^{M1}+P^{M2} > 2\delta$ and $P+2P^{M1}+P^{M2} > 2\epsilon + 6\delta$, then either S_2^2 or $S_{12}S_{21}$ gives the minimum cycle time.*

Proof. In this region since we assumed that $P^{M1} + P^{M2} < 2\delta$, we have:

$$2\epsilon + 6\delta \leq P + 2P^{M1} + P^{M2} < P + P^{M1} + 2\delta \Rightarrow P + P^{M1} > 2\epsilon + 4\delta.$$

$T_{S_{12}S_{21}(\Pi^*)}$ is given in equation (6.5a). When we compare this cycle time with $T_{S_1^2}$ given in equation (D.1), we conclude that $T_{S_{12}S_{21}(\Pi^*)} < T_{S_1^2}$. □

As a result of this lemma we showed that S_1^2 is dominated in this region. On the other hand the following example will show that we cannot establish any dominance relation between cycles S_2^2 and $S_{12}S_{21}$.

Example 6.1 Let us suppose that we have 4 operations with a total processing time of 100, $\epsilon = 10$, and $\delta = 10$. Because of the tooling constraints, one of the operations with a processing time of 10 must be processed on the first machine, and another one with a processing time of 5 must be processed on the second machine. Therefore, $P^{M1} = 10$ and $P^{M2} = 5$. The remaining two operations with a total operation time of 85 will be allocated to the machines. When we observe the parameters, we see that Lemma 6.5 is applicable to this case. Since $P + P^{M1} \geq 2\epsilon + 4\delta$, the cycle time for $S_{12}S_{21}$ is given in equation (6.5a) and with given parameters becomes 155.

For S_2^2, the allocation of the operations becomes important. For the first case, assume that we have a total of two operations to be allocated for which, one operation has a processing time of 50 and the other 35 making a total of 85. Calculating the cycle time of S_2^2 given in equation (D.3), we get 140, which is less than the cycle time of $S_{12}S_{21}$. For the second case, assume that we have again two operations to be allocated for which, one operation has a processing time of 75 and the other 10 making a total of 85. Now, the cycle time for S_2^2 is equal to 160. Since this is greater than 155, for this case $S_{12}S_{21}$ is optimal. Therefore we conclude that depending on the allocation of the operations of S_2^2, either S_2^2 or $S_{12}S_{21}$ can be optimal.

Combining the findings in Lemmas 6.2–6.5, we now provide the main result of this chapter.

Theorem 6.5

1- If $P^{M1} + P^{M2} \geq 2\delta$*, then* S_2^2 *gives the minimum cycle time,*

2- Else,

2.1- If $2P + P^{M1} + P^{M2} \leq 2\delta$*, then* S_1^2 *gives the minimum cycle time,*

2.2- Else,

2.2.1- If $2P^{M1} + P^{M2} + P \leq 2\epsilon + 6\delta$*, then* $S_{12}S_{21}$ *gives the minimum cycle time,*

2.2.2- Else, depending on the allocation of the operations for S_2^2*, either* S_2^2 *or* $S_{12}S_{21}$ *gives the minimum cycle time.*

Remember that for this theorem and the proof we assumed that $P^{M1} \geq P^{M2}$. For the reverse case, the results can easily be adapted in analogy with the above analysis resulting in the following corollary:

Corollary 6.1 *If we assume that* $P^{M1} < P^{M2}$ *all cases of Theorem 6.5 are still valid except case 2.2.1 which should be replaced with the following:*

2.2.1 If $P^{M1} + 2P^{M2} + P \leq 2\epsilon + 6\delta$*, then* $S_{12}S_{21}$ *gives the minimum cycle time.*

6.2.3 Sensitivity Analysis

In this subsection we will perform sensitivity analysis on parameters such as the robot transportation time δ, and the loading (or unloading) time of the

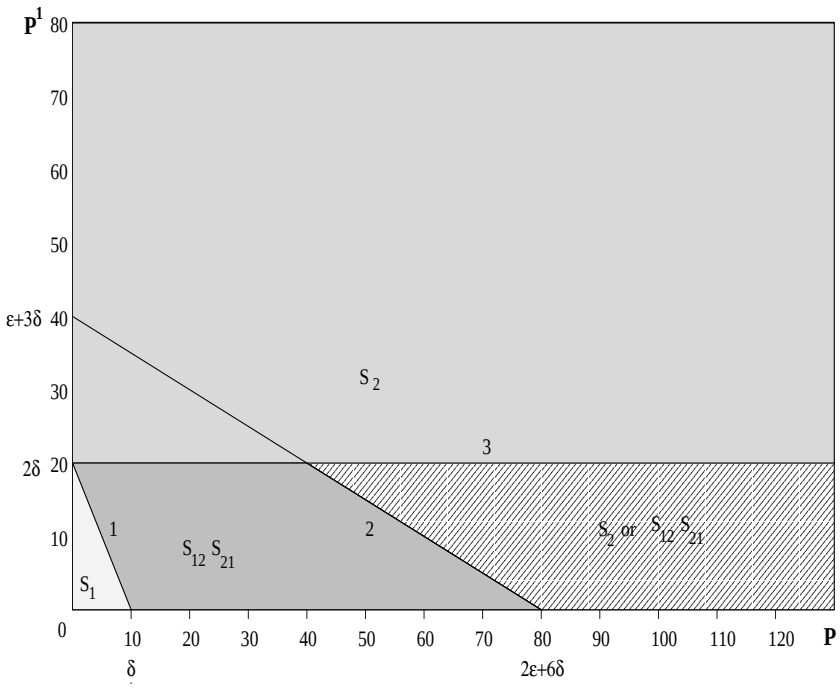

Figure 6.2: Regions of optimality for Example 6.2

machines ϵ, and show how the regions of optimality change with a change in these parameters. We represent P^{M2} as αP^{M1}, where $0 \leq \alpha \leq 1$. The following example will be useful in order to analyze the parameters graphically.

Example 6.2 Assume that $\epsilon = \delta = 10$ and $\alpha = 0$, which means that $P^{M2} = 0$. Figure 6.2 depicts the regions for this case as a graph of P versus P^{M1}.

Consider parameters ϵ, δ, and α one at a time. Theorem 6.5, Case 1 states that if $P^{M1} + P^{M2} \geq 2\delta$, then S_2^2 gives the minimum cycle time. From here we conclude that if the transportation time is zero or negligible, then S_2^2 always gives the minimum cycle time. This is logical since we can consider the robot as a third machine which is the bottleneck one. Then, the main concern is to minimize the waiting time of the robot in front of the machines. In order to achieve this, in cycle S_2^2, while processing of a part continues on one of the machines, the robot makes other activities (such as transportation, loading or

unloading of the other machine, waiting in front of the other machine, etc.) without being late.

The definition of n-unit cycles states that every machine is loaded and unloaded exactly n times. Thus the loading/unloading times are equivalent for all cycles. However, in cycles S_2^2 and $S_{12}S_{21}$, while processing continues on one of the machines, the robot does not wait in front of the machine and performs other activities and when the robot returns back to the machine to unload there is a partial waiting time in front of this machine. The loading/unloading time affects these partial waiting times.

α is defined as P^{M2}/P^{M1}, where $0 \leq \alpha \leq 1$. When we increase P^{M2} from 0 to P^{M1}, this is in favor of S_2^2 because in cycle S_2^2 the optimal allocation is the one which balances the processing times on both machines and when P^{M2} is close to P^{M1}, the ability to balance the processing times increases.

6.3 Conclusion

In this chapter, we studied the 2-machines, identical parts robotic operation allocation problem with tooling constraints. The operation allocation flexibility is a direct consequence of assuming that the machines in the robotic cell are CNC machines as is the case for machining operations. The problem is to find the allocation of the operations to the machines and the corresponding robot move cycle that jointly minimize the cycle time. As a solution to this problem, we proved in Theorem 6.4 that the optimal solution is either a 1-unit or a 2-unit cycle. In Theorem 6.5, we presented the regions of optimality for these robot move cycles. We showed that the study of Sethi et al. [86] becomes a special case of our study. We conducted a sensitivity analysis on parameters. An extended version of this chapter is accepted for publication [38].

Chapter 7

Bicriteria Robotic Cell Scheduling

In this chapter, different from the earlier ones, we will consider a bicriteria optimization problem in the context of robotic cell scheduling. In scheduling theory and practice, two main objectives are time and cost. Minimizing production time (equivalently maximizing throughput) could have the highest priority in "production planning", while minimizing production costs has the highest priority in "process planning". It should also be noted that the former of these objectives is relevant when the demand is assumed to be unlimited. However, in today's highly competitive environment, most industries face a limited demand. Although there is an extensive literature on robotic cell scheduling problems, to the best of our knowledge, none of these considers cost objectives. Furthermore, the trade-offs involved in considering several different criteria provide useful insights to the decision maker. For example, a solution which minimizes the cycle time (long run average time to produce one part) may perform poorly in terms of cost. Thus, in the context of real life scheduling problems it is more relevant to consider problems with such dual

criteria nature. The following chapters consider cost objectives simultaneously with time objectives in the context of robotic cells.

We will consider 2- and 3-machine robotic cells which produce identical parts. Each of the identical parts is assumed to have a number of operations to be performed on the machines. In robotic cells, highly flexible Computer Numerical Control (CNC) machines are used for the metal cutting operations so that the machines and the robot can interact on a real time basis. Machining conditions such as the cutting speed and the feed rate are controllable variables for these machines. Consequently, the processing time of any operation on these machines can be reduced by changing the machining conditions at the expense of incurring extra cost resulting in the opportunity of reducing the cycle time. Due to this reasoning, assuming the processing times to be fixed on each machine is not realistic. In the following chapters, the processing times are taken as decision variables. Different from the current literature, the problem is not only to find the robot move sequence but also to determine the processing times of the operations on the machines that simultaneously minimize the cycle time and the total manufacturing cost. Since we have two criteria, the optimal solution will not be unique but instead a set of nondominated solutions will be identified. A solution is called nondominated if no other feasible solution has smaller objective function values for both performance measures.

The processing time for each operation can be optimized from two different points of view: (i) minimizing cost per unit, or (ii) maximizing production rate. The first criterion is common and basic to all manufacturing. On the contrary, in the current robotic cell scheduling literature only the second criterion (e.g. minimizing the overall cycle time or maximizing throughput) is discussed extensively. This objective is important when the production order must be completed as quickly as possible. When there is limited demand, robotic cells should operate in the interval between these two cases (referred

to as "high-efficiency range") that could be defined by generating a set of nondominated solutions by solving this bicriteria optimization problem.

Most of the studies in the existing literature of controllable processing times assume a linear cost function (Vickson [95], van Wassenhove and Baker [97], Daniels and Sarin [22], Janiak and Kovalyov [54], Cheng et al. [17]). Although this assumption simplifies the problem, it is not realistic because it does not reflect the law of diminishing returns. Thus, in the following chapters we assume a nonlinear, convex, differentiable cost function.

The organization of this chapter is as follows: In the next section we will present some new notation and definitions that will be used from now on. The problem definition and the mathematical formulation will also be presented in the next section. In Section 7.2, 2- and 3-machine cells will be analyzed and the set of nondominated solutions will be determined. In Section 7.3, different cost structures including the cost incurred by the robot will be analyzed. Section 7.4 is devoted to the concluding remarks.

7.1 Problem Definition

In the following chapters, due to the complexity of the problem, as most of the studies of the robotic cell scheduling literature, we will restrict ourselves with 1-unit cycles since they are simple, practical and provably give good results. In this chapter each part is assumed to have a number of operations $o_1, o_2, \ldots, o_p$ in an m-machine robotic cell. In this chapter, we assume one operation to be performed on each machine and the allocation of operations to the machines is predefined. As a consequence, $p = m$ and o_i represents the operation to be performed on machine i with corresponding processing time denoted by $t_i = P_i$. On the other hand, in the next chapter we will also consider the additional

problem of allocating the operations to the machines as well in which case we assume $p \geq m$.

Processing times on the CNC machines can be written as functions of the machine parameters such as the cutting speed and the feed rate. As a consequence, selecting different parameters yields different processing time values. Total manufacturing cost for the CNC machines can be written as the summation of machining and tooling costs. The machining cost can be considered as a function of either the exact working time of the machines or the cycle time which includes some idle time for the machines. The former of these assumes that the machines incur cost only if they perform some operation on the parts. However, the latter one assumes that another job cannot be scheduled during these idle times. We will start with the former of these assumptions and the latter case will be analyzed in Section 7.3 where we consider different cost structures. There is a tradeoff between machining and tooling costs in selecting the processing time values. Reducing the processing time reduces the machining cost but at the same time it reduces the tool life which in turn increases the tooling cost. Conversely, increasing the processing time increases the tool life and thus reduces the tooling cost, but this increases the machining cost.

Kayan and Akturk [63] determined lower and upper bounds for the processing times in order to minimize a convex cost function and any regular scheduling measure. The lower bound of a processing time is derived from constraints such as the limited tool life, machine power and surface roughness. On the other hand, the upper bound of a processing time is the processing time value for which the total manufacturing cost is minimized, so that beyond this value of processing time, both objectives get worse. Note that, these upper and lower bounds are different from time window constraints used in hoist scheduling problems which indicates that any schedule that causes the hoist

not to pick up a part within the time window is infeasible. In this study, a schedule in which the processing times exceed their upper bounds is still feasible but proved to be not optimal. The lower bound corresponds to the minimum processing time-maximum cost case whereas the upper bound corresponds to the maximum processing time-minimum cost case. Let P_i^L and P_i^U denote the lower and upper bounds for the processing time of operation o_i and $f_i(P_i)$ denote the manufacturing cost incurred by the same operation. In this study, we assume $f_i(P_i)$ to be strictly convex and differentiable. As a consequence, from the derivation of the lower and upper bounds of the processing times, it is monotonically decreasing for $P_i^L \leq P_i \leq P_i^U$, $i = 1, 2, \ldots, m$. As a consequence, we can write the total manufacturing cost incurred by all the operations as $\sum_i f_i(P_i)$, which is also a convex, differentiable function for $P_i^L \leq P_i \leq P_i^U$, $\forall i$. Obviously, the total manufacturing cost does not depend on the robot move cycle but depends only on the processing times of the operations whereas the cycle time depends on both. Figure 7.1 depicts the machining, tooling and the total manufacturing costs with respect to the processing time of an operation. P_i^L and P_i^U values and the cost function in between these values are also depicted. It is clear from the determination of the lower and the upper bounds that the portion of the manufacturing cost function lying in between the bounds is monotonically decreasing.

We denote a processing time vector as $\boldsymbol{P} = (P_1, P_2, \ldots, P_m)$. Any processing time violating one of its bounds is called infeasible. As a consequence, we can define the set of feasible processing time vectors as $\mathcal{P}_{feas} = \{(P_1, P_2, \ldots, P_m) \in R^m : P_i^L \leq P_i \leq P_i^U, \ \forall i\}$. On the other hand, feasible robot move cycles are defined by Crama et al. [19] as the cycles in which the robot does not load an already loaded machine and does not unload an already empty machine. For example in a 2-machine robotic cell, there are two feasible 1-unit cycles namely, S_1^2 and S_2^2 where S_i^m represents the i^{th} robot

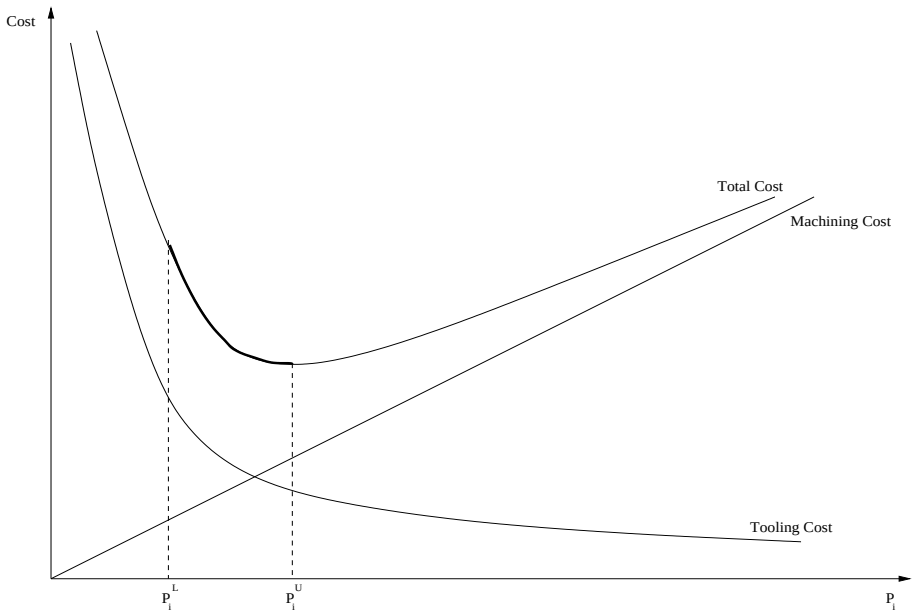

Figure 7.1: Manufacturing cost with respect to processing time

move cycle in an m-machine robotic cell. We denote the set of all feasible robot move cycles in an m-machine robotic cell as $\mathcal{S}^m_{feas}$. Before proceeding, let us present some new definitions and notation that will be in the following chapters.

T : Cycle time, i.e., the long run average time that is required to produce one part.

C_o : Operating cost of the machines. Since we assume the machines to be identical, operating cost is the same for each machine.

K_i : Cost of tool i used (for $i = 1, \ldots, m$, since each operation might require a different tool).

$F_1(S, \boldsymbol{P}) = \sum_{i=1}^{m} f_i(P_i)$: Total manufacturing cost which depends only on the processing times. Note that the individual cost functions for each operation o_i, $f_i(P_i)$ is strictly convex and differentiable and it is monotonically decreasing for $P_i^L \leq P_i \leq P_i^U$, $i = 1, 2, \ldots, m$.

$F_2(S, \boldsymbol{P})$: Cycle time corresponding to robot move cycle S and processing time vector $\boldsymbol{P}$.

As a result of the bounding scheme explained above, we can formulate the bicriteria problem as follows:

$$\begin{aligned} \min \quad & \textit{Total manufacturing cost} \\ \min \quad & \textit{Cycle time} \\ \text{Subject to} \quad & P_i^L \leq P_i \leq P_i^U, \quad \forall i. \end{aligned}$$

This formulation minimizes two conflicting objectives simultaneously. There are different ways to deal with bicriteria problems. We shall adopt the notation summarized in Hoogeveen [50]. Let f and g represent the two performance measures. The first method minimizes a linear composite objective function in f and g with unknown relative weights and is denoted by $G_l(f, g)$. The second way is called the *hierarchical optimization* or the *lexicographical optimization* and is denoted by $Lex(f, g)$. In this approach, performance measure f is assumed to be more important than g. As a result, this problem minimizes g subject to the constraint that the solution value of f is minimum. The third one is called the *epsilon-constraint method* denoted by $\epsilon(f|g)$. In this approach, nondominated points are found by solving a series of problems of the form minimize f given an upper bound on g. The last approach which is the most difficult one and which will be used in this study minimizes a composite objective function in f and g and is denoted by $G(f, g)$. In this approach, the only foreknowledge is that the composite function G is nondecreasing in both arguments.

The decision variables of the bicriteria problem formulated above are the processing times as well as the robot move cycles. In this study, we will consider each 1-unit cycle individually. In other words, for each 1-unit cycle we will

solve the bicriteria problem to determine the processing times and compare these 1-unit cycles with each other. However, in order to be able to find solutions minimizing both objectives simultaneously for 1-unit cycle S, we will first consider the epsilon-constraint formulation of the problem. That is, we will consider $\epsilon(F_1(S, \boldsymbol{P})|F_2(S, \boldsymbol{P}))$ to determine the sufficient conditions for the processing time values minimizing the manufacturing cost for a given level of cycle time. Using these conditions we will be able to write the manufacturing cost as a function of the cycle time, which means we will be able to determine the composite objective function G. As a result, for any given cycle time (manufacturing cost) value we will be able to determine the corresponding manufacturing cost (cycle time) value and the processing times of the parts on the machines.

Epsilon-Constraint Problem (ECP)

$$\begin{aligned} \min \quad & \textit{Total manufacturing cost} \\ \text{Subject to} \quad & \text{Cycle time} \leq T, && (7.1) \\ & P_i^L \leq P_i \leq P_i^U, \quad \forall i. && (7.2) \end{aligned}$$

In this chapter, a solution to the bicriteria problem defines both a feasible robot move cycle and a corresponding feasible processing time vector for the parts. More formally, we can state a solution as follows:

Definition 7.1 *A solution to the bicriteria problem for an m-machine robotic cell is represented as* $\xi = (S^m, \boldsymbol{P})$ *where* $S^m \in \boldsymbol{\mathcal{S}}^m_{feas}$ *and* $\boldsymbol{P} \in \mathcal{P}_{feas}$. *Let* $X = \{\xi = (S^m, \boldsymbol{P}) : S^m \in \boldsymbol{\mathcal{S}}^m_{feas} \text{ and } P \in \mathcal{P}_{feas}\}$ *be the set of all feasible solutions.*

In the context of bicriteria optimization theory, solution ξ_1 dominates solution ξ_2 if it is not worse than ξ_2 under any of the performance measures,

and is strictly better under at least one of the performance measures. Nondominated solutions are classified as Pareto optimal. We can state these more formally as follows:

Definition 7.2 *We say that ξ_1 dominates ξ_2 and denote it as $\xi_1 \preceq \xi_2$ if and only if $F_1(\xi_1) \leq F_1(\xi_2)$ and $F_2(\xi_1) \leq F_2(\xi_2)$, one of which is a strict inequality. A solution $\xi^* \in X$ is called Pareto optimal, if there is no other $\xi \in X$ such that $\xi \preceq \xi^*$. If ξ^* is Pareto optimal, $z^* = (F_1(\xi^*), F_2(\xi^*))$ is called efficient. The set of all efficient points is the efficient frontier.*

Recall that, in this study the problem is twofold. That is, we both try to find the robot move sequence and the processing times of the parts on the machines. In order to achieve this, we will fix the robot move cycles and for each robot move cycle we will determine the set of nondominated processing time vectors. In other words, we will solve the bicriteria problem for each 1-unit cycle. The set of nondominated processing time vectors for an arbitrary 1-unit robot move cycle S_i^m can be defined as follows:

Definition 7.3 *$P^*(S_i^m) = \{\boldsymbol{P} \in \mathcal{P}_{feas}$: There is no other $\overline{\boldsymbol{P}} \in \mathcal{P}_{feas}$ such that $(S_i^m, \overline{\boldsymbol{P}}) \preceq (S_i^m, \boldsymbol{P})\}$.*

We already defined how one solution dominates another solution. However, while comparing robot move cycles with each other we will make use of the following, which defines how one robot move cycle dominates another one in the context of this study.

Definition 7.4 *A cycle S_i^m is said to dominate another cycle S_j^m ($S_i^m \preceq S_j^m$) if there is no $\hat{P} \in P^*(S_j^m)$ such that $(S_j^m, \hat{P}) \preceq (S_i^m, \tilde{P})$, $\forall \tilde{P} \in P^*(S_i^m)$.*

In the current literature, the processing times are assumed to be fixed. A cycle is said to dominate another one if the cycle time of the former is less than that of the latter with the same, fixed processing times used for both cycles. However, in order to find a dominance relation between two cycles as stated in Definition 7.4, the processing times used in the two cycles need not be the same. Hence, a dominance relation between two cycles is found by comparing the minimum cost values of the two cycles corresponding to the same cycle time value. That is, $F_1(S_i^m, \tilde{P})$ is compared with $F_1(S_j^m, \hat{P})$, for all $\tilde{P} \in P^*(S_i^m)$ and $\hat{P} \in P^*(S_j^m)$, where $F_2(S_i^m, \tilde{P}) = F_2(S_j^m, \hat{P})$. Although in such a flexible environment, 1-unit cycles may not be optimal, we will restrict ourselves with these cycles as is frequently done in the literature.

In the next section we will determine the set of nondominated processing time vectors for the 1-unit cycles for 2- and 3-machine cells.

7.2 Solution Procedure

In this section we will consider 2- and 3-machine cells respectively. For each 1-unit cycle, S, we will determine $P^*(S)$, the set of nondominated processing time vectors and then compare these cycles with each other in light of Definition 7.4 to find sufficient conditions under which each of the cycles remains nondominated among all 1-unit cycles.

In the next section we will analyze the 2-machine cells and in Section 7.2.2 we will analyze the 3-machine cells.

7.2.1 2-Machine Case

Let us first analyze the S_1^2 cycle. Recall that the activity sequence of S_1^2 is $A_0A_1A_2$. The cycle time of this cycle can be calculated as $6\epsilon+6\delta+P_1+P_2$. In order to minimize the cost for a given cycle time value, T, the first constraint (7.1) of the ECP must be replaced by:
$6\epsilon + 6\delta + P_1 + P_2 \leq T$.

The following lemma is one of the major contributions of this chapter which determines $P^*(S_1^2)$, the processing times of the parts on each machine under the S_1^2 cycle that simultaneously minimize the cycle time and the total manufacturing cost. Let (P_1^*, P_2^*) be an optimal solution to the ECP formulated for the S_1^2 cycle, where the cycle time is bounded by T. Note that, $(P_1^*, P_2^*) \in P^*(S_1^2)$ according to Definition 7.3.

Lemma 7.1

1. *If* $T = 6\epsilon+6\delta+P_1^L+P_2^L$, *then* $P_1^* = P_1^L$ *and* $P_2^* = P_2^L$. *The corresponding cost is,* $F_1(S_1^2, (P_1^L, P_2^L)) = f_1(P_1^L) + f_2(P_2^L)$.

2. *If* $T = 6\epsilon+6\delta+P_1^U+P_2^U$, *then* $P_1^* = P_1^U$ *and* $P_2^* = P_2^U$. *The corresponding cost is,* $F_1(S_1^2, (P_1^U, P_2^U)) = f_1(P_1^U) + f_2(P_2^U)$.

3. *If* $6\epsilon + 6\delta + P_1^L + P_2^L < T < 6\epsilon + 6\delta + P_1^U + P_2^U$ *then optimal processing times of the ECP are found by solving the following equations:*
   $6\epsilon + 6\delta + P_1^* + P_2^* = T$ *and*
   $\partial f_1(P_1^*) = \partial f_2(P_2^*)$.
   After solving, one may get one of the following cases:

   3.1 *If both processing times satisfy their own bounds then the solution found is optimal.*

3.2 Else if exactly one of the processing times, P_i^*, violates one of its bounds, say P_i^b, then the optimal solution is $P_i^* = P_i^b$ and $P_j^* = T - 6\epsilon - 6\delta - P_i^b$, $i, j = 1, 2$, $i \neq j$.

3.3 Else if one of the processing times (assume P_i^*) violates its lower bound (P_i^L) and the other one (P_j^*) violates its upper bound (P_j^U) then the optimal solution is found by comparing the manufacturing costs of the following two processing time settings:
(i) $P_i^* = P_i^L$, $P_j^* = T - 6\epsilon - 6\delta - P_i^L$ or
(ii) $P_j^* = P_j^U$, $P_i^* = T - 6\epsilon - 6\delta - P_j^U$, $i, j = 1, 2$, $i \neq j$.

Proof. For S_1^2, the cycle time satisfies the following, $6\epsilon + 6\delta + P_1^L + P_2^L \leq T \leq 6\epsilon + 6\delta + P_1^U + P_2^U$. If $T = 6\epsilon + 6\delta + P_1^L + P_2^L$, then there exists a unique solution where $P_1^* = P_1^L$ and $P_2^* = P_2^L$, with corresponding cost $f_1(P_1^L) + f_2(P_2^L)$. In the same way, if $T = 6\epsilon + 6\delta + P_1^U + P_2^U$, then there exists a unique solution where $P_1^* = P_1^U$ and $P_2^* = P_2^U$, with corresponding cost $f_1(P_1^U) + f_2(P_2^U)$. For the remaining case, let (P_1^*, P_2^*) be the optimal solution to our problem. Then, both of the following cannot hold at the same time: $P_i^* = P_i^L$ and $P_i^* = P_i^U$, unless $P_i^L = P_i^U$. Also since $6\epsilon + 6\delta + P_1^L + P_2^L < T < 6\epsilon + 6\delta + P_1^U + P_2^U$, either $P_1^* \neq P_1^L$ or $P_2^* \neq P_2^L$ and either $P_1^* \neq P_1^U$ or $P_2^* \neq P_2^U$. As a result, (P_1^*, P_2^*) is a regular point. Additionally, since the objective function and the constraints are convex, any point satisfying the Karush-Kuhn-Tucker (KKT) conditions is optimal. The Lagrangian function for point P^* is as follows:

$$L(P^*, \mu^*) = f_1(P_1^*) + f_2(P_2^*) + \mu^*(6\epsilon + 6\delta + P_1^* + P_2^* - T).$$

If we set $\nabla_P(L(P^*, \mu^*)) = 0$, we get:
$\partial f_1(P_1^*) + \mu^* = 0$ and $\partial f_2(P_2^*) + \mu^* = 0$,
with the additional constraints, $\mu^* \geq 0$ and $P_i^L \leq P_i^* \leq P_i^U$, $i = 1, 2$. As a result of these equations we have the following:

$$\mu^* = -\partial f_1(P_1^*) = -\partial f_2(P_2^*). \tag{7.3}$$

On the other hand, since $\partial f_i(P_i^*) < 0$ for $P_i^* < P_i^U \Rightarrow \mu^* = -\partial f_i(P_i^*) > 0$, which implies that the corresponding constraint must be satisfied as equality:

$$6\epsilon + 6\delta + P_1^* + P_2^* = T. \tag{7.4}$$

P_i^* can be found by solving equations (7.3) and (7.4) simultaneously. If exactly one of the P_i^* values violates one of its upper or lower bounds, P_i^* is set to the bound which is violated and the remaining processing time is found correspondingly using equation 7.4. Both of the processing times can also violate their own bounds. This can only be the case if one of the processing times violates its lower bound and the other one violates its upper bound. Let $P_i^* < P_i^L$ and $P_j^* > P_j^U$, $i, j = 1, 2$, $i \neq j$. Then there exist two alternative solutions as stated in the statements (3.3.(i)) and (3.3.(ii)) of this lemma and the optimal solution is found by comparing the manufacturing cost values for these two alternatives.

Note that, in order to determine the optimal processing time values, a nonlinear equation system must be solved (equations 7.3 and 7.4) which has a unique root for $P_i^* \geq 0$, $i = 1, 2$. The solution of these equations can be approximated by using either the Newton's method, the golden search algorithm or a bisection algorithm. □

The above solution finds the processing times which give the minimum cost for a given cycle time value. That is, allocating the given resource (in this case the cycle time) to two alternatives (in this case the processing times on the two machines) without violating the bounds. While allocating this resource, priority is given to the alternative (processing time) which has the highest contribution to the cost. That is, the processing time which has the highest contribution to the cost is increased more than the other one without exceeding the corresponding bounds. According to this lemma, for given manufacturing cost functions, $f_i(P_i)$, $\forall i$, the optimal processing times of the

ECP can be written as functions of the cycle time, T. Using this fact, the total manufacturing cost can also be written as a function of T, which determines the efficient frontier of the bicriteria problem. The range of the cycle time can easily be determined by using the lower and the upper bounds of the processing time values. As a result, the minimum manufacturing cost (cycle time) value corresponding to any given cycle time (manufacturing cost) value can be determined easily. The processing times of the parts on the machines can also be determined. The machine parameters such as the speed and the feed rate are determined using these processing times.

Till now we considered the cost function to be any convex, nonlinear, differentiable function. Now let us consider more specifically a single-tool, single pass turning operation on CNC machines. For a more detailed explanation of the cost figures used in this part we refer the reader to Kayan and Akturk [63]. For this operation, the total manufacturing cost can be written as the summation of the machining and the tooling costs. Machining cost is $C_o \cdot (P_1 + P_2)$, where C_o is the operating cost of the CNC machine (\$/minute). Recall that in this section we assume the machining cost to be allocated in terms of the exact working times of the machines (P_1, P_2). Different allocation schemes will be analyzed in Section 7.3. On the other hand, the tooling cost is $K_1U_1P_1^{a_1} + K_2U_2P_2^{a_2}$, where $K_i > 0$ and $a_i < 0$ are specific constants for tool i and $U_i > 0$ is a specific constant for operation i regarding parameters such as the length and the diameter of the operation. We assume that each operation is performed with a corresponding tool. Let us consider a given cycle time value, $T = 6\epsilon + 6\delta + P_1 + P_2 \Rightarrow P_1 + P_2 = T - 6\epsilon - 6\delta$. Then the machining cost can be rewritten as $C_o \cdot (T - 6\epsilon - 6\delta)$, which is constant for a given cycle time, T. In order to find the minimum total cost corresponding to T, the tooling cost will be minimized and summed with the corresponding machining cost. Then using Lemma 7.1 the solution can be found as follows:

1. If $T = 6\epsilon+6\delta+P_1^L+P_2^L$, then $P_1^* = P_1^L$ and $P_2^* = P_2^L$. The corresponding cost is, $C_o \cdot (T - 6\epsilon - 6\delta) + K_1U_1(P_1^L)^{a_1} + K_2U_2(P_2^L)^{a_2}$.

2. If $T = 6\epsilon+6\delta+P_1^U+P_2^U$, then $P_1^* = P_1^U$ and $P_2^* = P_2^U$. The corresponding cost is, $C_o \cdot (T - 6\epsilon - 6\delta) + K_1U_1(P_1^U)^{a_1} + K_2U_2(P_2^U)^{a_2}$.

3. Otherwise, P_i^* is found by solving the following two equations, $6\epsilon + 6\delta + P_1^* + P_2^* = T$ and $P_2^* = (\frac{K_1U_1a_1}{K_2U_2a_2})^{\frac{1}{a_2-1}}(P_1^*)^{\frac{a_1-1}{a_2-1}}$. If any of the processing times violates any of the bounds, update all processing times accordingly so that they each satisfy their bounds and $6\epsilon + 6\delta + P_1^* + P_2^* = T$.

If the operations on both machines are performed with a tool of the same type, then $a_1 = a_2 = a$ and $K_1 = K_2 = K$. In this case the above equations can be solved easily to determine the processing times as follows:

$$P_1^* = \frac{(T - 6\epsilon - 6\delta)U_2^{\frac{1}{a-1}}}{U_1^{\frac{1}{a-1}} + U_2^{\frac{1}{a-1}}} ,$$

and

$$P_2^* = \frac{(T - 6\epsilon - 6\delta)U_1^{\frac{1}{a-1}}}{U_1^{\frac{1}{a-1}} + U_2^{\frac{1}{a-1}}} .$$

As a consequence, the optimal total cost can be written in terms of the cycle time as follows:

$$\begin{aligned} F_1 = \quad & C_o \cdot (T - 6\epsilon - 6\delta) + \frac{KU_1U_2(T-6\epsilon-6\delta)^a}{(U_1^{\frac{1}{a-1}}+U_2^{\frac{1}{a-1}})^{a-1}} , \\ & 6\epsilon + 6\delta + P_1^L + P_2^L \le T \le 6\epsilon + 6\delta + P_1^U + P_2^U . \end{aligned}$$

This identifies the whole set of nondominated solutions and shows the exact tradeoff between the cycle time T and the total manufacturing cost F_1.

Now let us consider the S_2^2 cycle for which the activity sequence can be written as $A_0A_2A_1$. The cycle time of S_2^2 can be calculated to be $\max\{6\epsilon +$

$8\delta, P_1 + 4\epsilon + 4\delta, P_2 + 4\epsilon + 4\delta\}$. Thus, constraint (7.1) of the ECP is replaced by the following:

$\max\{6\epsilon + 8\delta, P_1 + 4\epsilon + 4\delta, P_2 + 4\epsilon + 4\delta\} \leq T$.

It is obvious that under S_2^2, T satisfies $\max\{6\epsilon+8\delta, P_1^L+4\epsilon+4\delta, P_2^L+4\epsilon+4\delta\} \leq T \leq \max\{6\epsilon + 8\delta, P_1^U + 4\epsilon + 4\delta, P_2^U + 4\epsilon + 4\delta\}$. Restricting T to this region, the above constraint can be replaced by the following two linear constraints:

$P_1 + 4\epsilon + 4\delta \leq T$ and

$P_2 + 4\epsilon + 4\delta \leq T$.

Lemma 7.2 *Under cycle S_2^2, for a given cycle time level T, the processing times minimizing the cost are: $P_i^* = \min\{P_i^U, T - 4\epsilon - 4\delta\}$, $i = 1, 2$.*

Proof. Any point (P_1^*, P_2^*) for which $P_1^L < P_1^* < P_1^U$ and $P_2^L < P_2^* < P_2^U$ is a regular point and under these conditions $P_1^* = P_2^* = T - 4\epsilon - 4\delta$ is the point satisfying the KKT conditions. Since the objective function and the constraints are convex, this point is optimal. Including the bounds, the optimal processing times can be rewritten as $P_i^* = \min\{P_i^U, T - 4\epsilon - 4\delta\}$, $i = 1, 2$. As one can observe, for any nonlinear, convex cost function we get the same processing time values. □

As a consequence of this lemma, the total manufacturing cost can be written as a function of the cycle time which defines the efficient frontier of the bicriteria problem. The intuition behind this lemma is the following: Having greater processing times without exceeding the upper bounds of the processing times and the given cycle time level T is better in terms of manufacturing cost and in the above case the processing times are set to their maximum allowable level. Note that in this cycle, after loading a part to one of the machines the robot does not wait in front of the machine but instead performs other activities and returns back to unload the part after finishing these activities. Then, if the

processing of a part finishes before the robot returns back to unload the part, the speed of the machine can be reduced so that the processing time is increased without increasing the cycle time. This means having less cost with the same cycle time value. Furthermore, it is apparent that the optimal processing times on both machines are balanced under the S_2^2 cycle. A numerical example will be helpful for understanding.

Example 7.1 Let us consider S_2^2 cycle for this example and consider a turning operation and assume that both machines use a tool of the same type. Let the parameters be given as follows: $K = 4$, $C_o = 0.5$, $U_1 = 0.2$, $U_2 = 0.03$, $a = -1.43423$, $P_1^L = 0.5$, $P_1^U = 1.4$, $P_2^L = 0.3$, $P_2^U = 0.64$, $\epsilon = 0.1$ and $\delta = 0.2$. Let us first consider the solution where all of the processing times are set to their lower bounds, $(S_2^2, (0.5, 0.3))$. The Gantt chart on top of Figure 7.2 depicts this cycle. For this solution, $F_1(S_2^2, (0.5, 0.3)) = 3.237$ and $F_2(S_2^2, (0.5, 0.3)) = 2.2$. If we analyze this cycle, we observe that the robot never waits and is the bottleneck for this case. Without increasing the cycle time, we can increase the processing time on the first machine from 0.5 to 1 and the processing time on the second machine from 0.3 to 1. Now let us find the optimal processing times on these machines for $T = 2.2$ by using Lemma 7.2. $P_i^* = \min\{P_i^U, T - 4\epsilon - 4\delta\} \Rightarrow P_1^* = 1, P_2^* = 0.64$. That is, the processing time of the first machine is increased up to the end of the idle time period shown in Figure 7.2, but the processing time of the second machine could not be increased because the upper bound of this processing time is less than this value. The Gantt chart for this solution is depicted as the second chart in Figure 7.2. As it is seen, for this case the robot never waits and $F_1(S_2^2, (1, 0.64)) = 1.848$ and $F_2(S_2^2, (1, 0.64)) = 2.2$. From Definition 7.4, we conclude that $(S_2^2, (1, 0.64)) \preceq (S_2^2, (0.5, 0.3))$. Thus, we eliminate $(S_2^2, (0.5, 0.3))$ from further consideration. Let us also consider another solution in which all of the processing times are fixed to their upper bounds, $(S_2^2, (1.4, 0.64))$. The Gantt chart for this solution is depicted as the

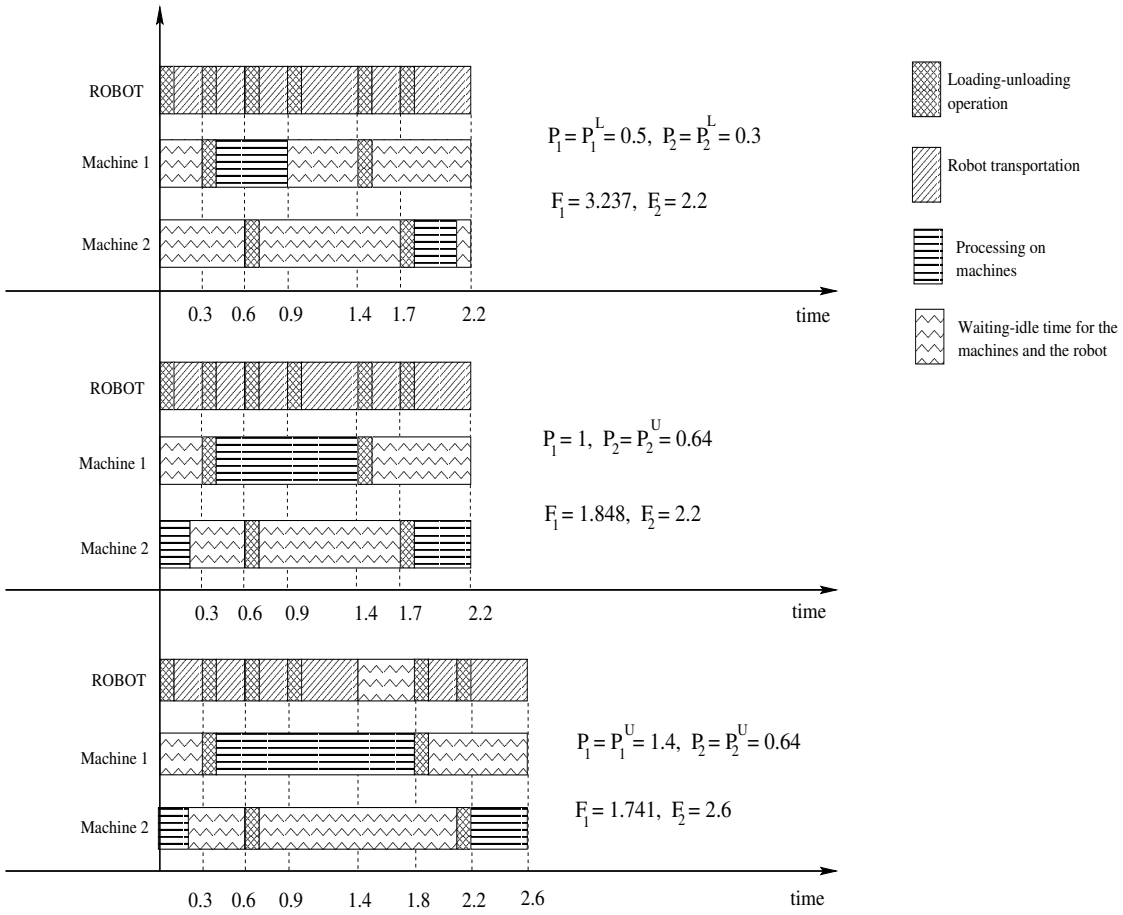

Figure 7.2: Gantt charts for different processing times for Example 7.1

last one in Figure 7.2. Note that in this case the first machine becomes the bottleneck and the robot waits for this machine in order to finish the processing. In this case, $F_1(S_2^2,(1.4,0.64)) = 1.7413$ and $F_2(S_2^2,(1.4,0.64)) = 2.6$. When we compare this solution with $(S_2^2,(1,0.64))$, $F_1(S_2^2,(1.4,0.64)) < F_1(S_2^2,(1,0.64))$ but $F_2(S_2^2,(1.4,0.64)) > F_2(S_2^2,(1,0.64))$. That is, none of these two solutions dominates one another.

After characterizing $P^*(S_1^2)$ and $P^*(S_2^2)$, the following theorem compares the two 1-unit robot move cycles S_1^2 and S_2^2 with each other and finds the sufficient conditions under which one dominates the other.

Theorem 7.1 *Whenever, S_2^2 is feasible ($T \geq 6\epsilon + 8\delta$), it dominates S_1^2.*

Proof. The cycle time of the S_2^2 cycle can be at least $6\epsilon + 8\delta$. Hence, for the cycle time values less than $6\epsilon+8\delta$, S_2^2 cycle is not feasible and we have $S_1^2 \preceq S_2^2$. Now let us consider the region where the cycle time is at least $6\epsilon + 8\delta$ and compare the two cycles for the same cycle time value. Let $(\hat{P}_1, \hat{P}_2) \in P^*(S_1^2)$, which satisfies $T = \hat{P}_1 + \hat{P}_2 + 6\epsilon + 6\delta$. The optimal processing times for S_2^2 with the same cycle time value are the following: $\tilde{P}_i = \min\{P_i^U, \hat{P}_1 + \hat{P}_2 + 2\epsilon + 2\delta\}, \quad i = 1, 2$, where $(\tilde{P}_1, \tilde{P}_2) \in P^*(S_2^2)$. Since $P_i^U \geq \hat{P}_i$ and $\hat{P}_1 + \hat{P}_2 + 2\epsilon + 2\delta \geq \hat{P}_i$, then $\tilde{P}_i \geq \hat{P}_i$. For $P_i^L \leq P_i^* \leq P_i^U$, the total manufacturing cost is monotonically decreasing. Since for the same cycle time value, the optimal processing times for the S_2^2 cycle are greater than that of the S_1^2 cycle, that means the total manufacturing cost of the S_2^2 cycle is less than that of S_1^2 cycle. Consequently, we have $S_2^2 \preceq S_1^2$. □

This theorem is one of the major contributions of this chapter and states that for a given cell data, for the cycle time values that can be attained by the S_2^2 cycle, the minimum cost is also attained by the same cycle. However, for very small cycle time values which cannot be attained by the S_2^2 cycle, although the cost values can be very high, S_1^2 cycle is still an alternative for the decision maker. Note that $T < 6\epsilon+8\delta$ if and only if $\hat{P}_1+\hat{P}_2 < 2\delta$. This fact can be used to rewrite the above theorem. According to the different values of the bounds of the processing times, different versions of this theorem can also be created. For example, if $P_1^L + P_2^L \geq 2\delta$ then all the cycle time values that can be attained by the S_1^2 cycle can also be attained by the S_2^2 cycle. As a result, $S_2^2 \preceq S_1^2$ in the whole region. In a similar way if $P_1^U + P_2^U < 2\delta$ then $S_1^2 \preceq S_2^2$ in the whole region. From these, we can conclude that for greater processing times S_2^2 is preferable to S_1^2 and for smaller processing times vice versa. Observe that $T = 6\epsilon+8\delta$ or $\hat{P}_1+\hat{P}_2 = 2\delta$, is the region of indifference in

the case of Sethi et al. [86]. However, in the settings of this study, in this region S_2^2 dominates S_1^2. That is, previous studies can not handle the cost component and thus state that both cycles perform identically. However, although both cycles give the same cycle time value, S_2^2 has a smaller manufacturing cost value and is preferred to S_1^2. Additionally, if $P_1^U \leq 2\epsilon + 4\delta$ and $P_2^U \leq 2\epsilon + 4\delta$, then the cycle time of S_2^2 can only take one value which is equivalent to $6\epsilon + 8\delta$.

The following example will aid in understanding such special cases.

Example 7.2 Let us consider a turning operation and assume that both machines use a tool of the same type. Let the parameters be given as follows: $K = 4$, $C_o = 0.5$, $U_1 = 0.2$, $U_2 = 0.03$, $a = -1.43423$, $P_1^L = 0.1$, $P_1^U = 1.4$, $P_2^L = 0.08$, $P_2^U = 0.64$ and $\epsilon = 0.02$. In order to present different occurrences of the efficient frontier four different values are used for δ. Using Lemmas 7.1 and 7.2, the efficient frontiers for these two cycles are drawn in Figure 7.3. In the first case, let $\delta = 0.1$. As a result, $P_1^L + P_2^L < 2\delta < P_1^U + P_2^U$. The bold curves show that for $T < 6\epsilon + 8\delta = 0.92$, $S_1^2 \preceq S_2^2$ and otherwise $S_2^2 \preceq S_1^2$. Although the cost of the S_1^2 cycle for $T < 6\epsilon + 8\delta$ is very high, the cycle time is smaller than that of S_2^2 and this region is still an alternative for the decision maker. In the second case, let $\delta = 0.08$, which results in $P_1^L + P_2^L \geq 2\delta$. As it is seen from the figure, for all cycle time and cost combinations, S_2^2 is preferable to S_1^2. In the third case, let $\delta = 1.1$. In this case, $P_1^U + P_2^U \leq 2\delta$ and the only cycle time value that S_2^2 can take is given by $6\epsilon + 8\delta = 8.92$. The minimum cost corresponding to this value of cycle time is found by setting $P_i^* = P_i^U$, $i = 1, 2$. On the other hand, when the same processing time settings are used for the S_1^2 cycle, the cycle time becomes 8.76. Since the same processing time values are used, the cost is the same for both cycles. Thus, we conclude that in this case $S_1^2 \preceq S_2^2$. Lastly, let $\delta = 0.4$. Since $P_1^U \leq 2\epsilon + 4\delta$ and $P_2^U \leq 2\epsilon + 4\delta$, the cycle time of S_2^2 can only be $6\epsilon + 8\delta = 3.32$ and this cycle time value corresponds

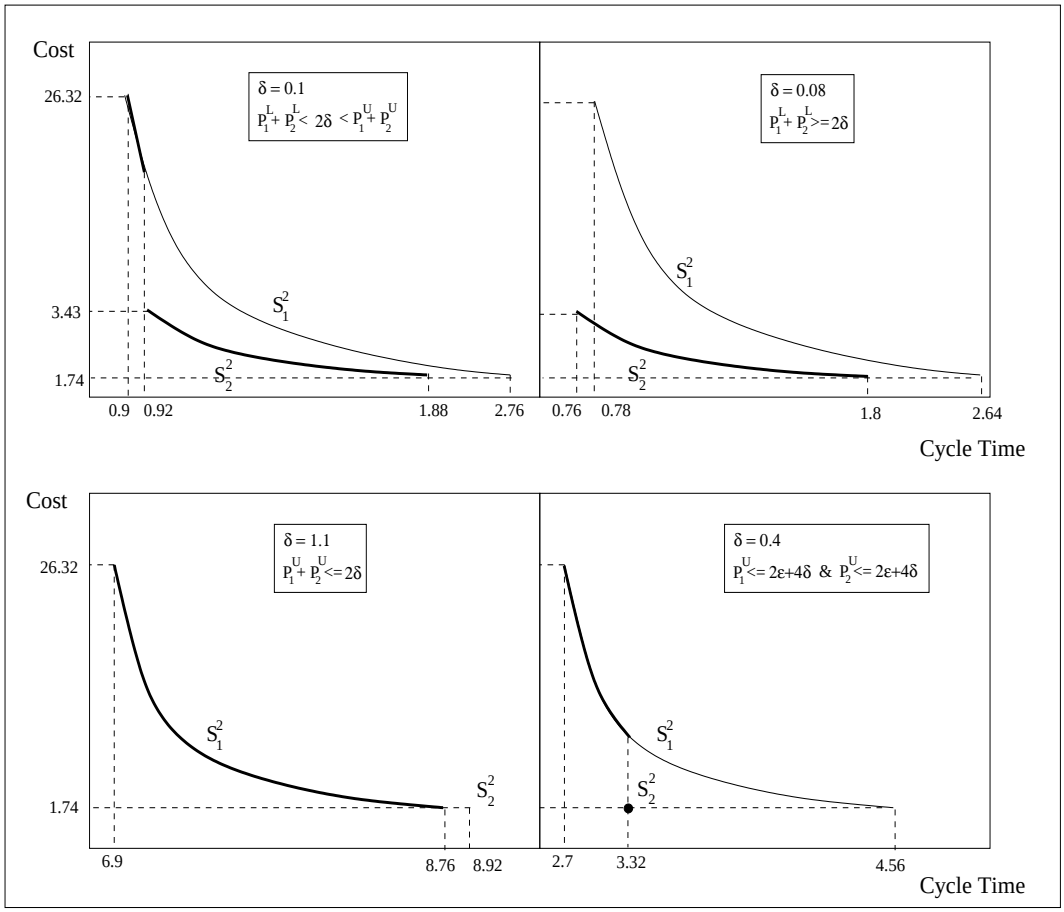

Figure 7.3: Different occurrences of the efficient frontier with respect to given parameters

to a cost of 1.74. S_1^2 cannot take a cost value less than 1.74. As a result, S_2^2 dominates S_1^2 unless the cycle time of $S_1^2 < 3.32$. For cycle time values smaller than 3.32, the only alternative is the S_1^2 cycle.

Akturk et al. [2] proved that 1-unit cycles need not be optimal in the whole region even in 2-machine robotic cells with single objective function when the processing times on the machines are not assumed to be fixed. The following example proves a similar result for this study by finding a processing time setting for the 2-unit cycle $S_{12}S_{21}$, in which for the same cycle time value $S_{12}S_{21}$ gives the minimum cost. One repetition of this cycle produces two parts

and we will determine the processing times of these two parts on the machines which can be different from each other. In order to denote this, let P_{ij} represent the processing time on M_i for part j, $i = 1, 2$ and $j = 1, 2$. The cycle time for this cycle is derived in Equation (D.6) where in this case $P^{M1} = P^{M2} = 0$:

$$\frac{1}{2}\max\{P_{11}+P_{22}+12\epsilon+14\delta, P_{11}+P_{22}+P_{12}+10\epsilon+10\delta, P_{11}+P_{22}+P_{21}+10\epsilon+10\delta\}$$

Example 7.3 Let us again consider the turning operation and use the same parameter values for this example as the one above with $\delta = 0.1$. Let $P_{11} = 0.1$, $P_{12} = 0.44$, $P_{21} = 0.44$ and $P_{22} = 0.08$. With these settings, the cycle time of $S_{12}S_{21}$ is 0.91 and the total manufacturing cost is 6.325. Since the cycle time of S_2^2 cannot take values less than 0.92 with these parameters, $S_{12}S_{21} \preceq S_2^2$. On the other hand, for $T = 0.91$, the minimum cost for S_1^2 is 9.214 $\Rightarrow S_{12}S_{21} \preceq S_1^2$.

This example shows that 1-unit cycles need not be optimal, even for 2-machines, under the assumptions of this study. However, the following theorem determines the regions where they are optimal and their worst case performances for the regions where they may not be optimal.

Theorem 7.2 *S_2^2 dominates all other robot move cycles whenever it is feasible ($T \geq 6\epsilon + 8\delta$) and S_1^2 dominates all other robot move cycles for $T < 6\epsilon + 7\delta$.*

Proof. For the general m-machine flowshop type robotic cells a lower bound for cycle time is presented in Equation (4.3). In this section we have $m = 2$, $P = P_1 + P_2$. As a consequence, this lower bound can be represented as follows for use in this section:

$$max\{6\epsilon + 6\delta + min\{P_1 + P_2, \delta\}, 4\epsilon + 4\delta + max\{P_1, P_2\}\}. \qquad (7.5)$$

Observing this equation we can state that for any given cycle time T, if $T < 6\epsilon + 7\delta$, then $P_1 + P_2 < \delta$. As a consequence, $T = 6\epsilon + 6\delta + P_1 + P_2$ which

is equivalent to the cycle time of the S_1^2 cycle. This concludes that for any given cycle time $T < 6\epsilon + 7\delta$, S_1^2 cycle dominates all other cycles. On the other hand, if $T \geq 6\epsilon + 7\delta$, the processing times can at most be increased (the cost can at most be reduced) so that $4\epsilon + 4\delta + max\{P_1, P_2\} \leq T$. From here, since the processing times have upper bounds, the processing times satisfying the minimum cost for a given cycle time T is as follows, $P_i = max\{P_i^U, T - 4\epsilon - 4\delta\}$, $i = 1, 2$. Using this processing time setting, we get a lower bound for the cost for given cycle time value T. From Lemma 7.2, this is equivalent to the processing time setting that minimizes the cost for given T under the S_2^2 cycle. However, S_2^2 cycle is feasible for $T \geq 6\epsilon + 8\delta$. This completes the proof. □

This theorem determines the regions of optimality for the two 1-unit cycles. It is also shown that for $6\epsilon + 7\delta \leq T < 6\epsilon + 8\delta$, 1-unit cycles need not be optimal. Note that, in this region S_2^2 is not feasible. In order to determine the worst case performance of the S_1^2 cycle inside this region one can calculate the processing times for the S_1^2 cycle according to Lemma 7.1 and compare the cost corresponding to this setting of processing times with the cost corresponding to setting $P_i = max\{P_i^U, T - 4\epsilon - 4\delta\}$, $i = 1, 2$, to get a lower bound of the cost according to the theorem above.

The next section is devoted to the three-machine robotic cells.

7.2.2 3-Machine Case

Increasing the number of machines in a robotic cell increases the number of feasible robot move cycles drastically. More specifically, Sethi et al. [86] proved that the number of feasible 1-unit cycles for an m-machine robotic cell is $m!$. For a 3-machine robotic cell there are a total of six feasible 1-unit cycles. The robot activity sequences and the corresponding cycle times for these cycles are

presented in Appendix F. Note that S_1^3 cycle is very similar, with respect to robot activity sequence and the cycle time formula, to S_1^2 cycle which may both be classified as the forward cycles and the S_6^3 cycle is very similar, again for similar reasons, to S_2^2 cycle both of which may be classified as the backward cycles. Let us first consider the forward move cycle, S_1^3. Proceeding just as in S_1^2, we can solve the single criterion problem. But this time we have three variables to determine, P_1, P_2 and P_3. The following lemma determines the optimal processing times for the ECP of the S_1^3 cycle.

Lemma 7.3 *Let (P_1^*, P_2^*, P_3^*) be the optimal processing times for the ECP formulated for the S_1^3 cycle for a given cycle time, T. Then, this point satisfies the following set of nonlinear equations:*
$\partial f_1(P_1^*) = \partial f_2(P_2^*) = \partial f_3(P_3^*)$,
$P_1^* + P_2^* + P_3^* = T - 8\epsilon - 8\delta$.
After solving,

1. *If all of the processing times satisfy their lower and upper bounds, then the solution found is optimal.*

2. *Else, if for exactly one index i, $i = 1, 2, 3$, P_i^* violates its bounds, set it to the bound which is violated. Let P_i^b represent the bound which is violated. Update T such that $\hat{T} = T - P_i^b$. In order to determine the remaining two processing times, proceed just as solving the S_1^2 cycle case with cycle time set to $\hat{T}$.*

3. *Else, if exactly two processing times violate their bounds and if both violate their lower or both violate their upper bounds then set them to their own violated bound. That is, for $i, j, k = 1, 2, 3$, $i \neq j$, $i \neq k$, $j \neq k$, if $P_i^* < P_i^L$ and $P_j^* < P_j^L$ (or $P_i^* > P_i^U$ and $P_j^* > P_j^U$), set $P_i^* = P_i^L$ and $P_j^* = P_j^L$ ($P_i^* = P_i^U$ and $P_j^* = P_j^U$). The last processing time is found as,*

$P_k^* = T - P_i^L - P_j^L - 8\epsilon - 8\delta$ ($P_k^* = T - P_i^U - P_j^U - 8\epsilon - 8\delta$). *Else, if one of the processing times violates its lower (assume w.l.o.g* $P_i^* < P_i^L$*) and the other one violates its upper bound (w.l.o.g* $P_j^* > P_j^U$, $i \neq j$*) then compare the manufacturing costs found by the following two alternatives:*
(i) Set $P_i^* = P_i^L$ *and solve for the remaining two processing times similar to the* S_1^2 *cycle,*
(ii) Set $P_j^* = P_j^U$ *and solve for the remaining two processing times similar to the* S_1^2 *cycle.*

4. *Else, if all processing times violate their own bounds, let* P_i^b *represent the violated bound for* P_i^*, $i = 1, 2, 3$. *Compare the manufacturing costs for the following three alternative solutions:*
(i) Set $P_1^* = P_1^b$ *and solve for the remaining two processing times similar to* S_1^2 *case,*
(ii) Set $P_2^* = P_2^b$ *and solve for the remaining two processing times similar to* S_1^2 *case,*
(iii) Set $P_3^* = P_3^b$ *and solve for the remaining two processing times similar to* S_1^2 *case.*

Proof. The proof is very similar to that for cycle S_1^2 and is omitted. □

Now let us consider the backward cycle, S_6^3. For this case, the cycle time T can take values between $\max\{8\epsilon+12\delta, P_1^L+4\epsilon+4\delta, P_2^L+4\epsilon+4\delta, P_3^L+4\epsilon+4\delta\} < T < \max\{8\epsilon + 12\delta, P_1^U + 4\epsilon + 4\delta, P_2^U + 4\epsilon + 4\delta, P_3^U + 4\epsilon + 4\delta\}$.

Lemma 7.4 *Under cycle* S_6^3*, the optimal processing times for the ECP for the* S_6^3 *cycle are* $P_i^* = \min\{P_i^U, T - 4\epsilon - 4\delta\}$, $i = 1, 2, 3$.

Proof. The proof is very similar to the S_2^2 case and is omitted. □

In the following two theorems, we will prove some dominance relations of the type stated in Definition 7.4. Considering only the 1-unit cycles, Crama and Van de Klundert [19] proved that the set of pyramidal permutations necessarily contains an optimal solution of the problem. Let $A_0, A_{i_1}, \ldots, A_{i_k}, A_{i_{k+1}}, \ldots, A_{i_m}$ denote the activity sequence of a 1-unit cycle in an m-machine cell. Then, Crama and Van de Klundert [19] defines this cycle to be pyramidal if $1 \leq i_1 < \ldots < i_k = m$ and $m > i_{k+1} > \ldots > i_m \geq 1$. The following is an important theorem which proves that the classical dominance of pyramidal cycles is valid with the assumptions of this study also.

Theorem 7.3 *The set of pyramidal cycles is dominating among 1-unit cycles.*

Proof. According to Theorem 3 of Crama and Van de Klundert [19], for any processing time setting there exists at least one pyramidal cycle which gives a smaller cycle time than any nonpyramidal cycle. This means that, for any processing time setting there exists at least one pyramidal cycle which has the same cost value with the nonpyramidal cycle but with a smaller cycle time value meaning that the nonpyramidal cycle is dominated. Note that, this processing time setting need not be optimal for the pyramidal cycle for this cost value. That is, with another processing time setting for the pyramidal cycle a smaller cycle time value can be found which corresponds to the same cost value. This completes the proof. □

In three-machine cells, the S_2^3 and S_4^3 cycles are nonpyramidal and the remaining ones are pyramidal. According to the above theorem these two cycles are dominated and can be eliminated from further consideration. In the following theorem we will compare the remaining cycles with each other and determine the regions where S_6^3 dominates the remaining cycles. In order to prove these, we will select an arbitrary T and find the optimal processing times for the ECP formulation for each cycle and compare them with each other. Let

$P_i^*(S_j^3)$ denote the nondominated processing times on machine i for the robot move cycle S_j^3.

Theorem 7.4 *Whenever S_6^3 is feasible ($T \geq 8\epsilon + 12\delta$), it dominates all of the remaining cycles.*

Proof. S_6^3 cannot take cycle time values less than $8\epsilon + 12\delta$. Thus, for an arbitrary selection of $T \geq 8\epsilon + 12\delta$, we will compare S_6^3 with S_1^3, S_3^3 and S_5^3 in the following cases respectively:

1. The optimal solution of ECP for S_1^3 satisfies, $T = 8\epsilon + 8\delta + P_1^*(S_1^3) + P_2^*(S_1^3) + P_3^*(S_1^3)$. Optimal processing time values of ECP for the S_6^3 corresponding to this cycle time value can be found by using Lemma 7.4 as $P_i^*(S_6^3) = \min\{P_i^U, 4\epsilon + 4\delta + P_1^*(S_1^3) + P_2^*(S_1^3) + P_3^*(S_1^3)\} \geq P_i^*(S_1^3)$. Therefore, $S_6^3 \preceq S_1^3$ in this region.

2. The optimal solution of ECP for S_3^3 satisfies, $T = 4\epsilon + 4\delta + \max\{P_3^*(S_3^3), P_1^*(S_3^3) + 4\epsilon + 6\delta, P_1^*(S_3^3) + P_2^*(S_3^3) + 2\epsilon + 2\delta\}$. Optimal processing time values of ECP for the S_6^3 corresponding to this cycle time value can be found by using Lemma 7.4 as, $P_i^*(S_6^3) = \min\{P_i^U, T - 4\epsilon - 4\delta\} = \min\{P_i^U, \max\{P_3^*(S_3^3), P_1^*(S_3^3)+4\epsilon+6\delta, P_1^*(S_3^3)+P_2^*(S_3^3)+2\epsilon+2\delta\} \geq P_i^*(S_3^3)$. Thus, $S_6^3 \preceq S_3^3$ for $T \geq 8\epsilon + 12\delta$.

3. As one can observe from the cycle time functions presented in Appendix F the cycle time function of S_5^3 is very similar to that of S_3^3. The only difference is the places of $P_1^*(S_3^3)$ and $P_3^*(S_3^3)$. When we swap the places of these two in the cycle time function of S_3^3, we get the cycle time function of S_5^3. Therefore, this case is identical with case 2, the only difference being the places of $P_1^*(S_3^3)$ and $P_3^*(S_3^3)$.

□

This theorem derives a similar result to Theorem 7.1 for 3-machine cells. According to this theorem, for a given cell data such as the loading/unloading time and robot transportation time, the backward cycle gives the minimum cost values for the cycle time values that can be attained by this cycle. The remaining three cycles, S_1^3, S_3^3 and S_5^3 can only be optimal for the cycle time values that cannot be attained by S_6^3. Sethi et al. [86] provided a decision tree on conditions for the robot move cycles to be optimal with any given cell data considering only the cycle time. However, the above theorem shows that earlier results are not valid anymore when the manufacturing cost is considered besides the cycle time. That is, considering the cycle time as the only objective hinders the additional insights provided by the cost of the suggested settings for the cell.

Let us now consider the region for $T < 8\epsilon + 12\delta$. In this region three cycles remain nondominated. According to the cycle times of these cycles presented in Appendix F, one can easily verify that under the cycle S_1^3, $T \geq 8\epsilon + 8\delta + P_1^L + P_2^L + P_3^L$. Similarly, for the cycle S_3^3, $T \geq \max\{P_1^L + 8\epsilon + 10\delta, P_1^L + P_2^L + 6\epsilon + 6\delta, P_3^L + 4\epsilon + 4\delta\}$. For the cycle S_5^3, $T \geq \max\{P_1^L + 4\epsilon + 4\delta, P_2^L + P_3^L + 6\epsilon + 6\delta, P_3^L + 8\epsilon + 10\delta\}$. In Lemma 7.3, we determined the optimal processing time values of the ECP for S_1^3. In the sequel we will prove similar results for the cycles S_3^3 and S_5^3, respectively.

Lemma 7.5 *Under the cycle S_3^3, the optimal processing times of the ECP are found as follows:*

1. *If $P_1^U + P_2^U < T - 6\epsilon - 6\delta$ or $P_2^U < 2\epsilon + 4\delta$, then $P_1^* = \min\{P_1^U, T - 8\epsilon - 10\delta\}$, $P_2^* = P_2^U$ and $P_3^* = \min\{P_3^U, T - 4\epsilon - 4\delta\}$,*

2. *Otherwise, $P_3^* = \min\{P_3^U, T - 4\epsilon - 4\delta\}$ and P_1^* and P_2^* are found by solving the following two equations simultaneously: $P_1^* + P_2^* = T - 6\epsilon + 6\delta$*

and $\partial f_1(P_1^*) = \partial f_2(P_2^*)$. *After solving, one may get one of the following cases:*

2.1 If both processing times satisfy their own bounds then the solution found is optimal.

2.2 Else if exactly one of the processing times, P_i^*, *violates one of its bounds,* P_i^b, *then the optimal solution is* $P_i^* = P_i^b$ *and* $P_j^* = T - 6\epsilon - 6\delta - P_i^b$, $i, j = 1, 2$, $i \neq j$.

2.3 Else if one of the processing times (assume P_i^**) violates its lower bound (*P_i^L*) and the other one (*P_j^**) violates its upper bound (*P_j^U*) then the optimal solution is found by comparing the manufacturing costs of the following two processing time settings:*
(i) $P_i^* = P_i^L$, $P_j^* = T - 6\epsilon - 6\delta - P_i^L$ *or*
(ii) $P_j^* = P_j^U$, $P_i^* = T - 6\epsilon - 6\delta - P_j^U$, $i, j = 1, 2$, $i \neq j$.

Proof. In order to find the optimal processing times, the objective function of the ECP must be replaced by $f_1(P_1) + f_2(P_2) + f_3(P_3)$, and the constraint (7.1) must be written as $\max\{P_1 + 8\epsilon + 10\delta, P_1 + P_2 + 6\epsilon + 6\delta, P_3 + 4\epsilon + 4\delta\} \leq T$.

Under this cycle, the cycle time is bounded as follows:

$$\begin{aligned} &\max\{P_1^L + 8\epsilon + 10\delta, P_1^L + P_2^L + 6\epsilon + 6\delta, P_3^L + 4\epsilon + 4\delta\} \leq T \\ &\leq \max\{P_1^U + 8\epsilon + 10\delta, P_1^U + P_2^U + 6\epsilon + 6\delta, P_3^U + 4\epsilon + 4\delta\}. \end{aligned} \tag{7.6}$$

As a result, the above constraint can be rewritten as the union of three constraints as follows:

$$P_1 + 8\epsilon + 10\delta \leq T, \tag{7.7}$$

$$P_1 + P_2 + 6\epsilon + 6\delta \leq T, \tag{7.8}$$

$$P_3 + 4\epsilon + 4\delta \leq T. \tag{7.9}$$

The Lagrangian for this formulation is the following:

$$L(P^*, \lambda^*, \mu^*) = f_1(P_1^*) + f_2(P_2^*) + f_3(P_3^*) + \mu_1^*(P_1^* - T + 8\epsilon + 10\delta) + \mu_2^*(P_1^* + P_2^* - T + 6\epsilon + 6\delta) + \mu_3^*(P_3^* - T + 4\epsilon + 4\delta).$$

If we set $\nabla_P(L(P^*, \mu^*)) = 0$, we get:
$\partial f_1(P_1^*) + \mu_1^* + \mu_2^* = 0$,
$\partial f_2(P_2^*) + \mu_2^* = 0$ and
$\partial f_3(P_3^*) + \mu_3^* = 0$.
We also have $\mu_i \geq 0$ and $P_i^L \leq P_i \leq P_i^U$, $\forall i$.

From the last equation we get $\mu_3^* = -\partial f_3(P_3^*)$. Since the objective function is strictly convex and decreasing, $-\partial f_3(P_3^*) > 0$ unless $P_3^* = P_3^U$. Thus, constraint (7.9) must be satisfied as equality. However, P_3^* cannot violate its bounds. From equation (7.6), $P_3^L \leq T - 4\epsilon - 4\delta$. As a result, $P_3^* = \min\{P_3^U, T - 4\epsilon - 4\delta\}$. Now let us consider the following cases:

1. If $P_1^U + P_2^U < T - 6\epsilon - 6\delta$ or $P_2^U < 2\epsilon + 4\delta$, then constraint (7.8) cannot be satisfied as equality. As a result, $\mu_2^* = -\partial f_2(P_2^*) = 0 \Rightarrow P_2^* = P_2^U$. Also since $\mu_2^* = 0$, $\mu_1^* = -\partial f_1(P_1^*) > 0$ unless $P_1^* = P_1^U$. Thus, constraint (7.7) is satisfied with equality. However, P_3^* cannot violate its bounds. From equation (7.6), $P_1^L \leq T - 8\epsilon - 10\delta$. As a result, $P_1^* = \min\{P_1^U, T - 8\epsilon - 10\delta\}$.

2. Otherwise, $\mu_2^* = -\partial f_2(P_2^*) > 0$ unless $P_2^* = P_2^U$. Thus, in this case constraint (7.8) is satisfied with equality. As a result $\mu_1^* = \partial f_2(P_2^*) - \partial f_1(P_1^*) \geq 0$. Thus, we have the following cases:

   2.1. If $\partial f_1(T - 8\epsilon - 10\delta) \leq \partial f_2(2\epsilon + 4\delta)$, then $P_1^* = T - 8\epsilon - 10\delta$ and $P_2^* = 2\epsilon + 4\delta$.

   2.2. Else, solve $\partial f_1(P_1^*) = \partial f_2(P_2^*)$ and $P_1^* + P_2^* = T - 6\epsilon - 6\delta$ simultaneously to find P_1^* and P_2^*.

Instead of these two cases we can simply represent the solution as follows: Solve $\partial f_1(P_1^*) = \partial f_2(P_2^*)$ and $P_1^* + P_2^* = T - 6\epsilon - 6\delta$ simultaneously to find P_1^* and P_2^*. If any of them violates its bounds, set that processing time to the violated bound and find the other one accordingly. The upper bound for P_1^* in this case is $\min\{P_1^U, T - 8\epsilon - 10\delta\}$.

□

When we compare the cycle time of this cycle with the S_5^3 cycle, we easily see that when we replace P_1 with P_3 in one of the cycle times, we get the cycle time of the other one. Thus, the analysis for these two cycles are identical. Consequently, the proof of the following lemma is very similar to the one above and will not be presented here.

Lemma 7.6 *Under the cycle S_5^3, the optimal processing times of ECP are found as follows:*

1. *If $P_3^U + P_2^U < T - 6\epsilon - 6\delta$ or $P_2^U < 2\epsilon + 4\delta$, then $P_1^* = \min\{P_1^U, T - 8\epsilon - 10\delta\}$, $P_2^* = P_2^U$ and $P_3^* = \min\{P_3^U, T - 4\epsilon - 4\delta\}$,*

2. *Otherwise, $P_1^* = \min\{P_1^U, T - 4\epsilon - 4\delta\}$ and P_3^* and P_2^* are found by solving the following two equations simultaneously: $P_3^* + P_2^* = T - 6\epsilon - 6\delta$ and $\partial f_3(P_3^*) = \partial f_2(P_2^*)$. After solving, one may get one of the following cases:*

   2.1 If both processing times satisfy their own bounds then the solution found is optimal.

   2.2 Else if exactly one of the processing times, P_i^*, violates one of its bounds, P_i^b, then the optimal solution is $P_i^* = P_i^b$ and $P_j^* = T - 6\epsilon - 6\delta - P_i^b$, $i, j = 2, 3$, $i \neq j$.

2.3 Else if one of the processing times (assume P_i^*) violates its lower bound (P_i^L) and the other one (P_j^*) violates its upper bound (P_j^U) then the optimal solution is found by comparing the manufacturing costs of the following two processing time settings:
(i) $P_i^* = P_i^L$, $P_j^* = T - 6\epsilon - 6\delta - P_i^L$ or
(ii) $P_j^* = P_j^U$, $P_i^* = T - 6\epsilon - 6\delta - P_j^U$, $i, j = 2, 3$, $i \neq j$.

The following is a good example to illustrate the differences of this study from the earlier ones.

Example 7.4 Let us consider a three-machine cell and CNC turning operations with following parameters: $\epsilon = 0.02$, $\delta = 0.1$, $K = 4$, $C_o = 0.5$, $a = -1.43423$, $U_1 = 0.2$, $U_2 = 0.03$, $U_3 = 0.75$, $P_1^L = 0.1$, $P_1^U = 1.4$, $P_2^L = 0.08$, $P_2^U = 0.64$, $P_3^L = 1.1$, $P_3^U = 2.42$. Let us determine the optimal processing times for the S_3^3 cycle with cycle time given as $T(S_3^3) = 1.8$. According to Lemma 7.5, the optimal processing times for this cycle can be determined to be as follows: $P_1^*(S_3^3) = 0.74$, $P_2^*(S_3^3) = 0.34$, $P_3^*(S_3^3) = 1.32$. The corresponding cost for this setting of processing time is 5.012. When we calculate the cycle time of the S_6^3 cycle with this same setting of processing times, as it is the case in the current literature, we also get $T(S_6^3) = 1.8$. This means that, both cycles have the same cycle time and cost values and hence we are indifferent between these two cycles. However, if we determine the optimal processing times for the S_6^3 cycle with $T = 1.8$ according to Lemma 7.4, we get $P_1^*(S_6^3) = 1.4$, $P_2^*(S_6^3) = 0.64$, $P_3^*(S_6^3) = 1.32$. The corresponding cost for this case is 4.416. This means that both cycles have the same cycle time value but the minimum cost corresponding this cycle time value for S_6^3 is less than that of S_3^3. Hence, S_3^3 is dominated.

This example shows that, under the assumptions of this study, when comparing the cycles with each other the processing time settings can be

different for each cycle. Additionally, if the only criteria was the cycle time, we would conclude that both S_3^3 and S_6^3 perform equally well. However, this example proves that the cost of S_6^3 is less than S_3^3 and hence they can not be considered as having equal performance.

Analyzing the remaining three cycles, S_1^3, S_3^3 and S_5^3 in the remaining region ($T < 8\epsilon + 12\delta$), we conclude that there is no general dominance relation among these cycles, but instead according to the parameters such as P_i^L, P_i^U, ϵ and δ, the regions where each of them dominates the others can be determined. This is another result that differentiates this study from the earlier ones since the decision tree provided by Sethi et al. [86] compares all of the 1-unit cycles with each other and presents the sufficient conditions for each of them to be optimal with any given cell data, where the only objective is the minimization of the cycle time. In other words, the decision tree spans the whole feasible region. However, with the assumptions of this study, only for $T < 8\epsilon + 12\delta$ the dominance relations among the remaining three cycles depend on the cell data.

7.3 Different Cost Structures

In this section we will show how different assumptions on cost structures for the machining cost and the cost of the robot can be handled. We will present the analysis for the 2-machine cells which can be extended to 3-machine cells in a similar manner. The machining cost can be assumed to be either a function of the exact working time of the machines or a function of the cycle time. Till now we assumed the former of these to hold. Additionally, the cost of the robot could also be considered as another cost component. Although the cost incurred by the robot is relatively small in comparison with the cost incurred

by the machines and the structure for the cost of the robot cannot be defined easily, we will consider two different cost structures for the robot in order to show how to handle additional cost components. In the sequel, we will give insights for handling different cost structures for the machining and robot costs.

7.3.1 Machining Cost as a Function of the Cycle Time

In this section, we assume the machining cost to be a function of the cycle time, $C_o \cdot T$, where $T = 6\epsilon+6\delta+P_1+P_2$ for the S_1^2 cycle and $T = 6\epsilon+8\delta+\max\{0, P_1-2\epsilon-4\delta, P_2-2\epsilon-4\delta\}$ for the S_2^2 cycle. The lower bounds for the processing times are determined by the constraints dictated by the limited tool life, the machine power, and the surface roughness. These constraints are independent of the machining cost. As a result, the lower bounds of the processing times remain unchanged. On the other hand, the upper bounds arise from the total manufacturing cost, which is different from the previous case. Furthermore, as opposed to the previous case, since the machining costs for the S_1^2 and S_2^2 cycles are different from each other, the total manufacturing costs are also different leading to different upper bounds of the processing times for identical operations under these two cycles. As we mentioned earlier, the upper bound for processing time P_i is the point satisfying $\frac{\partial f_i(P_i^U)}{\partial P_i} = 0$. Note that the total manufacturing cost is assumed to be a convex function, the machining cost is a nondecreasing function, and the tooling cost is a nonincreasing function. We also have the following:

$$\frac{\partial(C_o(6\epsilon+6\delta+P_1+P_2))}{\partial P_i} \geq \frac{\partial(C_o(6\epsilon+8\delta+\max\{0,P_1-2\epsilon-4\delta,P_2-2\epsilon-4\epsilon\}))}{\partial P_i}, \quad \forall i.$$

As a consequence, $P_i^U(S_2^2) \geq P_i^U(S_1^2)$, $\forall i$, where $P_i^U(S)$ is the upper bound of the processing time under robot move cycle S. Since the total manufacturing cost is a monotonically decreasing function of the processing time P_i for $P_i^L \leq P_i \leq P_i^U$, Lemmas 7.1 and 7.2 are still valid. Additionally, since $P_i^U(S_2^2) \geq P_i^U(S_1^2)$, $\forall i$, Theorem 7.1 is also valid which determines the regions where the

S_1^2 and S_2^2 cycles dominate each other.

7.3.2 Robot Cost as a Function of the Cycle Time

In this case the total cost function is assumed to consist of the machining cost, tooling cost, and robot cost. Similar to the machining cost, the cost of the robot can also be considered as a function of the cycle time. That is, a cost is incurred for each unit of time the cell works. The analysis in this case is very similar to that of the machining cost considered as a function of the cycle time. The new cost terms in this case are; $R \cdot (6\epsilon + 6\delta + P_1 + P_2)$ and $R \cdot (6\epsilon + 8\delta + \max\{0, P_1 - 2\epsilon - 4\delta, P_2 - 2\epsilon - 4\delta)\}$ for S_1^2 and S_2^2 respectively where R represents the cost for the robot for each unit of time the cell works ($/min). These new terms only affect the upper bounds of the processing times as shown in Section 7.3.1 and similarly, $P_i^U(S_2^2) \geq P_i^U(S_1^2)$, $\forall i$. Lemmas 7.1 and 7.2, which determine the set of nondominated solutions for the S_1^2 and S_2^2 cycles respectively and Theorem 7.1, which compares the two cycles with each other and determines the regions of optimality for these cycles, are still valid in this case as well.

7.3.3 Robot Cost as a Function of Exact Working Time

In this section we assume that the cost of the robot is computed with respect to the exact robot activity time. That is, if R represents the unit cost for the robot activity time, then the cost incurred by the robot is $R \cdot (6\epsilon + 6\delta)$ and $R \cdot (6\epsilon + 8\delta)$ under S_1^2 and S_2^2 cycles, respectively. Note that, during an n-unit cycle, each machine is loaded and unloaded exactly n times. As a result, the total load/unload times under all cycles to produce one part are equivalent to each other. On the other hand, robot travel times differ among

cycles. Comparing 1-unit cycles with each other, the robot travel time is greater under S_2^2 than under S_1^2. As a result, the cost incurred by the robot is greater under S_2^2 than S_1^2. Remember that, Lemmas 7.1 and 7.2 determined the set of nondominated solutions for S_1^2 and S_2^2 cycles respectively where the total cost function did not include the cost of the robot. By assuming the robot cost to be a function of the loading/unloading time and the robot transportation time but not the processing times, it becomes a constant for the two cycles S_1^2 and S_2^2. As a consequence, Lemmas 7.1 and 7.2 are still valid.

On the other hand, Theorem 7.1, which compares the two cycles with each other, is proved without the robot cost. The total cost functions for S_1^2 and S_2^2 cycles are identical without the robot cost, which means that for the same processing time values under both cycles, the total cost is equivalent for these two cycles. Additionally, the total cost function is assumed to be monotonically decreasing for the region under consideration which means that the cost will not increase with a greater processing time. In the proof of Theorem 7.1, we used these properties. However, as we include the robot cost, the total cost functions for S_1^2 and S_2^2 cycles become different from each other and Theorem 7.1 is no longer valid. Let $f_m(\boldsymbol{P})$ represent the machining cost and $f_t(\boldsymbol{P})$ represent the tooling cost with the processing time vector $\boldsymbol{P}$. The new breakpoint for the region of dominance satisfies the following:
$\bar{T} = 6\epsilon + 6\delta + \hat{P}_1 + \hat{P}_2 = 6\epsilon + 8\delta + \max\{0, \tilde{P}_1 - 2\epsilon - 4\delta, \tilde{P}_2 - 2\epsilon - 4\delta\}$ and
$f_m(\hat{P}_1, \hat{P}_2) + f_t(\hat{P}_1, \hat{P}_2) + R \cdot (6\epsilon + 6\delta) = f_m(\tilde{P}_1, \tilde{P}_2) + f_t(\tilde{P}_1, \tilde{P}_2) + R \cdot (6\epsilon + 8\delta)$
where $\hat{P}_i \in P^*(S_1^2)$ and $\tilde{P}_i \in P^*(S_2^2)$, $i = 1, 2$. If $T \leq \bar{T}$ then, $S_1^2 \preceq S_2^2$, otherwise $S_2^2 \preceq S_1^2$.

The following is an example showing that the breakpoint found in Theorem 7.1 is not valid for this new situation.

Example 7.5 Let us consider the turning operation for which the total cost

function for the S_1^2 cycle can be written as $C_o \cdot (P_1+P_2)+K_1U_1P_1^{a_1}+K_2U_2P_2^{a_2}+R \cdot (6\epsilon+6\delta)$. The only difference in the cost function for the S_2^2 cycle is the robot cost which is $R \cdot (6\epsilon+8\delta)$ for S_2^2. Let $K_1 = 6$, $K_2 = 5$, $C_o = 0.9$, $R = 2$, $\epsilon = 0.1$, $\delta = 0.2$, $U_1 = 0.03$, $U_2 = 0.8$, $a_1 = -0.6$ and $a_2 = -1$. According to the given parameters, the upper bounds for the processing times are found to be $P_1^U = 0.266$ and $P_2^U = 2.108$. Let $T = 3.3 > 6\epsilon + 8\delta = 2.2$. Then, if the robot cost is ignored, for $T = 3.3$, according to Theorem 7.1, $S_2^2 \preceq S_1^2$. With the inclusion of the robot cost, let $\hat{P}_1 = 0.152$, $\hat{P}_2 = 1.348$, where $\hat{P}_i \in P^*(S_1^2)$ and $\tilde{P}_1 = 0.266$, $\tilde{P}_2 = 2.1$, where $\tilde{P}_i \in P^*(S_2^2)$. As a consequence, $F_2(S_1^2,(0.152,1.348)) = F_2(S_2^2,(0.237,2.1)) = 3.3$. On the other hand, $F_1(S_1^2,(0.152,1.348)) = 8.475$ and $F_1(S_2^2,(0.237,2.1)) = 8.832$. As a result, in contrast to Theorem 7.1, with the presence of robot cost we have $S_1^2 \preceq S_2^2$, even though $T > 6\epsilon + 8\delta$.

This example shows that considering the cycle time as the only performance measure hinders the other characteristics of the solutions. Although a solution may have a small cycle time value, it may be dominated because of its poor cost performance. Even the basic results of Sethi et al. [86] regarding the 2-machine identical parts robotic cell scheduling problem are not valid when the cost is considered as a performance measure simultaneously with the cycle time. This brings additional insights to the problem and provides flexibility for the decision maker by determining the set of efficient solutions.

7.4 Conclusion

In this chapter, we considered a robotic cell with highly flexible CNC machines. The processing times of the parts on these machines can be controlled by adjusting the machining conditions such as the speed and the

feed rate. However, adjusting these parameters also affects the tool life which consequently affects the total manufacturing cost. Hence, in this chapter we considered a bicriteria robotic cell scheduling problem in which the robot move sequence as well as the processing times on the machines are the decision variables and the cycle time and the total manufacturing cost are the performance measures. Since there are two competing performance measures, instead of a unique optimal solution a set of nondominated solutions exists for such problems.

We determined the set of nondominated solutions for the two 1-unit cycles of 2-machine robotic cells in Lemmas 7.1 and 7.2, and compared these two cycles with each other in Theorem 7.1. A similar analysis is performed for 3-machine cells also. Theorem 7.3 proves that two of the six 1-unit cycles of a 3-machine cell are dominated and need not be considered any more. Lemmas 7.3, 7.4, 7.5 and 7.6 determine the nondominated set of solutions for the remaining four cycles. By comparing these with each other, Theorem 7.4 determines the regions where S_6^3 dominates the rest. Note that no dominance relations exist between the remaining three cycles for the remaining very small region. We carried out our analysis for any strictly convex, differentiable cost function. In Section 7.3, we showed how different assumptions on cost structures can be handled.

Chapter 8

Bicriteria Robotic Operation Allocation

In this chapter we will consider a more generalized bicriteria optimization problem in the context of robotic cell scheduling. We assume a robotic cell with 2-machines producing identical parts. Each of the identical parts has a set of operations $O = \{1, 2, \ldots, p\}$ to be performed. Recall that the processing time of operation l is represented by t_l. In the previous chapter we assumed $p = m$ for an m-machine robotic cell where the operations are preassigned to the m machines so that each machine performs only one specific operation of each part. However, as we proved in Chapter 6, by considering the allocation of the operations to the machines as a decision variable we can improve the efficiency of the cell in terms of the cycle time. As a consequence, in this chapter we assume that the number of operations is greater than or equal to the number of machines in the cell, i.e., $p \geq 2$ and the allocation of the operations to the machines is not known in advance. The problem considered in this chapter is threefold: (i) to determine the robot move cycle, (ii) to determine the allocation of the operations to the machines, and (iii) to determine the

processing times of the operations all at the same time. The objective is to minimize the cycle time together with the total manufacturing cost. Justified with the complexity of the problem and consistent with most of the studies of the robotic cell scheduling literature, we will restrict ourselves with 1-unit cycles.

The organization of this chapter is as follows: In the next section we will present the mathematical formulation of the problem. The solution procedure for the S_1^2 cycle will be developed in Section 8.2. In Section 8.3 we will present a heuristic procedure to determine a set of points on the efficient frontier for the S_2^2 cycle and in Section 8.4 we will evaluate the efficiency of the heuristic procedure with a computational study. Section 8.5 is devoted to the concluding remarks.

8.1 Problem Formulation

In this section the necessary mathematical notation for the problem will be presented. Each 1-unit cycle will be considered individually and for each cycle the allocation of the operations to the machines and the processing times of these operations on the machines will be determined. Similar to the previous chapter, we consider the cycle time and manufacturing cost objectives simultaneously. Since both of these objectives can not be improved at the same time, there is not a unique optimal solution but a set of nondominated solutions. We assume that both machines are capable of performing all of the operations that are in set O. The processing time of a part on a machine is equal to the summation of the processing times of the operations performed by that machine. Let x_{li} be the binary variable that indicates whether operation l is allocated to machine i or not. The total processing time on machine i, P_i,

can be written as $\sum_{l=1}^{p} x_{li} t_l$.

In this section, due to the complexity of the problem, a finite number of nondominated points will be determined using the epsilon-constraint method. In other words, $\epsilon(F_1(S, \boldsymbol{P})|F_2(S, \boldsymbol{P}))$ will be solved for a number of specific $F_2(S, \boldsymbol{P})$ values which are used to estimate the entire efficient frontier. Estimating the entire efficient frontier means that the cycle time values for which we solve the epsilon-constraint problem are uniformly spread over the range of all feasible cycle time values.

The epsilon-constraint problem with cycle time bound T can be formulated as follows:

Epsilon-Constraint Problem: $\boldsymbol{\epsilon(F_1|T)}$

$$\begin{aligned}
\min \quad & \textstyle\sum_{l=1}^{p} f_l(t_l) && && (8.1)\\
\text{Subject to} \quad & \text{Cycle time} \leq T, && && (8.2)\\
& x_{l1} + x_{l2} = 1, && \forall l \in [1, \ldots, p], && (8.3)\\
& t_l \geq t_l^L, && \forall l \in [1, \ldots, p], && (8.4)\\
& x_{li} \in \{0, 1\}, && \forall l \in [1, \ldots, p], \forall i \in [1, 2], && (8.5)
\end{aligned}$$

In the formulation above, the objective function is the total manufacturing cost which is assumed to be a strictly convex and differentiable function of the processing times of the operations. Constraint (8.2) puts an upper bound on the cycle time. Constraint set (8.3) forces each operation to be allocated to exactly one of the machines. Constraint set (8.4) sets the processing time lower bounds for each operation. Note that, in Constraint (8.4) the upper bounds of the processing times could also be used. However, from the derivation of these bounds we know that $\partial f_l(t_l^U) = 0$ and $\partial f_l(\hat{t}_l) > 0$ for $\hat{t}_l > t_l^U$. Hence, $t_l^* \leq t_l^U$ always holds for optimal t_l^* and is not required as an additional constraint.

In the following sections we will solve $\epsilon(F_1|T)$ by considering both cycles individually.

8.2 Solution Procedure for the S_1^2 Cycle

In this section we will develop a solution procedure for the $\epsilon(F_1|T)$ for the S_1^2 cycle. In particular, in order to solve the problem for the S_1^2 cycle we will use the following as Constraint (8.2) of $\epsilon(F_1|T)$:

$$6\epsilon + 6\delta + \sum_{l=1}^{p} x_{l1}t_l + \sum_{l=1}^{p} x_{l2}t_l \leq T. \tag{8.6}$$

From this equation it is obvious that regardless of the allocation of the operations, the cycle time of the S_1^2 cycle is the same. Hence, the binary allocation variables, x_{li}, can be eliminated from the formulation and we get the following as the cycle time bound constraint:

$6\epsilon + 6\delta + \sum_{l=1}^{p} t_l \leq T$.

Since $6\epsilon + 6\delta$ is constant for a given problem setting, letting $\hat{T} = T - 6\epsilon - 6\delta$ we have the following formulation for the S_1^2 cycle:

$$\boldsymbol{\epsilon(F_1|\hat{T})^{S_1^2}}: \qquad \min \quad \sum_{l=1}^{p} f_l(t_l) \tag{8.7}$$

$$\text{Subject to} \quad \sum_{l=1}^{p} t_l \leq \hat{T}, \tag{8.8}$$

$$t_l \geq t_l^L, \qquad \forall l \in [1, \ldots, p]. \tag{8.9}$$

This formulation is the same as a single machine makespan minimization problem with p jobs and controllable processing times which is a nonlinear knapsack problem. In the formulation above we have separable, convex continuous objective function and constraints for which different solution approaches can be reviewed in [12]. However, in the sequel we will develop a more detailed problem specific solution procedure. Since $\boldsymbol{\epsilon(F_1|\hat{T})^{S_1^2}}$ minimizes

a strictly convex function over a convex closed set, a local minimum of F_1 is a global minimum and there exists exactly one global minimum (See proposition 2.1.1 of Bertsekas [7]). Let $\boldsymbol{t^*} = (t_1^*, t_2^*, \ldots, t_p^*)$ be the optimal solution to $\epsilon(F_1|\hat{T})^{S_1^2}$ throughout this section. Let $T_{S_1^2}^L$ and $T_{S_1^2}^U$ be the lower and upper bounds of the cycle time of the S_1^2 cycle, respectively. These can be calculated by setting the processing times of all operations to their lower and upper bounds, respectively, as follows:

$$T_{S_1^2}^L = 6\epsilon + 6\delta + \sum_{l=1}^{p} t_l^L, \tag{8.10}$$

$$T_{S_1^2}^U = 6\epsilon + 6\delta + \sum_{l=1}^{p} t_l^U. \tag{8.11}$$

Also let $\hat{T}^L = T_{S_1^2}^L - 6\epsilon - 6\delta$ and $\hat{T}^U = T_{S_1^2}^U - 6\epsilon - 6\delta$. Note that, $\epsilon(F_1|\hat{T})^{S_1^2}$ is infeasible if the cycle time bound in constraint (8.8) satisfies $\hat{T} < \hat{T}^L$ and all solutions are dominated if $\hat{T} > \hat{T}^U$. Theorem 7.1 in the previous chapter gives a solution for the $\epsilon(F_1|\hat{T})^{S_1^2}$ when the number of operations is limited to 2. However, in this chapter the number of operations can be any positive integer value. Hence, a more general solution procedure for the S_1^2 cycle will be developed here. The following lemma is helpful in characterizing the optimal solution.

Lemma 8.1 *In the optimal solution to $\epsilon(F_1|\hat{T})^{S_1^2}$ for $\hat{T}^L \leq \hat{T} \leq \hat{T}^U$, constraint (8.8) is satisfied as equality.*

Proof. Let $F_1^* = \sum_{l=1}^{p} f_l(t_l^*)$ be the optimal objective function value of $\epsilon(F_1|\hat{T})^{S_1^2}$ with optimal processing time vector $\boldsymbol{t^*}$. Assume to the contrary that $\sum_{l=1}^{p} t_l^* < \hat{T}$. Then consider another solution with, $\hat{t}_l^* = t_l^*$, $\forall l \neq \hat{l}$ for an arbitrary index $\hat{l}$ such that $t_{\hat{l}}^* < t_{\hat{l}}^U$. Let $\hat{t}_{\hat{l}}^* = t_{\hat{l}}^* + \beta$, $0 < \beta \leq min\{t_{\hat{l}}^U - t_{\hat{l}}^*, \hat{T} - (\sum_{l=1}^{p} t_l^*)\}$. As a consequence, this new solution has identical processing times for all operations except $\hat{l}$ and $\hat{t}_{\hat{l}}^* > t_{\hat{l}}^*$. Since the cost function

is decreasing with respect to processing times, the objective function of the new solution, $\hat{F}_1^*$, satisfies: $\hat{F}_1^* < F_1^*$. However, this contradicts with $\boldsymbol{t^*}$ being the optimal solution to $\epsilon(F_1|\hat{T})^{S_1^2}$. □

As a consequence of the lemma above, we know that the sum of the optimal processing times is equal to the cycle time bound. Consider the partition induced by $\boldsymbol{t^*}$, i.e., $J = \{l : t_l^* > t_l^L\}$ and $\bar{J} = \{h : t_h^* = t_h^L\}$. We know that if $\hat{T} > \hat{T}^L$, then $J \neq \emptyset$. The following lemma determines the properties of the operations of these two sets.

Lemma 8.2 *In the optimal solution to $\epsilon(F_1|\hat{T})^{S_1^2}$, where $\hat{T} > \hat{T}^L$ the following conditions hold:*

i. $\partial f_l(t_l^*) = \partial f_k(t_k^*)$, $\forall l, k \in J$,

ii. $\partial f_l(t_l^*) \leq \partial f_h(t_h^*)$, $\forall h \in \bar{J}$ *and* $\forall l \in J$.

Proof. Since we assume $\hat{T} > \hat{T}^L$, then there exists at least one l such that $t_l^* > t_l^L$. Therefore, the vector $\boldsymbol{t^*} = (t_1^*, t_2^*, \ldots, t_p^*)$ is a regular point. A point is regular if the gradients of the active inequality constraints and the gradients of the equality constraints are linearly independent at that point. Such a point must satisfy the Karush-Kuhn-Tucker (KKT) conditions. From Lemma 8.1, Constraint (8.8) is satisfied as equality. As a consequence, the Lagrangian function for point $\boldsymbol{t^*}$ can be written as follows:

$$L(t^*, \lambda^*, \mu^*) = \sum_{l=1}^{p} f_l(t_l^*) + \lambda^* \left(\sum_{l=1}^{p} t_l^* - \hat{T} \right) + \sum_{l=1}^{p} \mu_l^* (t_l^L - t_l^*).$$

If we set $\nabla_t(L(t^*, \lambda^*, \mu^*)) = 0$, we get:

$$\partial f_l(t_l^*) + \lambda^* - \mu_l^* = 0, \quad \forall l.$$

If $l \in J$, then $\mu_l^* = 0$. Thus, $\partial f_l(t_l^*) = -\lambda^*$, $\forall l \in J$, which proves (i). On the other hand, if $h \in \bar{J}$, then $\partial f_h(t_h^*) = -\lambda^* + \mu_h^*$. Since $\mu_h^* \geq 0$, $\partial f_h(t_h^*) \geq -\lambda^* = \partial f_l(t_l^*)$, $\forall h \in \bar{J}$ and $\forall l \in J$, which proves (ii). □

Up to know, we know that the operations are partitioned into two sets with respect to their processing time values in the optimal solution to $\epsilon(F_1|\hat{T})^{S_1^2}$. Additionally, the lemma above, identifies some properties of the elements of these two sets regarding their processing time values. Let $\alpha_l = \partial f_l(t_l^L)$, $l = 1, 2, \ldots, p$. For each operation we can calculate a critical value of cycle time from the given problem data as follows:

$$M_l = \sum_{h \in O} max\{t_h^L, \partial {f_h}^{-1}(\alpha_l)\}. \tag{8.12}$$

The following lemma uses these values to determine the elements of the J and $\bar{J}$ sets easily, without determining the optimal processing times.

Lemma 8.3 *In the optimal solution to $\epsilon(F_1|\hat{T})^{S_1^2}$, $l \in J$ if and only if $\hat{T} > M_l$.*

Proof. (proof by contradiction). Let us first prove the necessity: Assume that $\hat{T} > M_h$ but to the contrary $h \in \bar{J}$, for at least one operation h. From the definition of set $\bar{J}$, $t_h^* = t_h^L$. But from condition (ii) of Lemma 8.2, the following must hold:

$$\alpha_h = \partial f_h(t_h^L) \geq \partial f_l(t_l^*), \quad \forall l \in J.$$

Since f_l is convex and invertible, we have the following:

$$t_l^* \leq \partial {f_l}^{-1}(\alpha_h), \quad \forall l \in J. \tag{8.13}$$

$\hat{T}$ can be written as:

$$\hat{T} = \sum_{l \in \bar{J}} t_l^L + \sum_{l \in J} t_l^*.$$

Using this together with inequality (8.13) we get:

$$\hat{T} = \sum_{l \in \bar{J}} t_l^L + \sum_{l \in J} t_l^* \leq \sum_{l \in \bar{J}} t_l^L + \sum_{l \in J} \partial {f_l}^{-1}(\alpha_h).$$

Since $\hat{T} > M_h$ is assumed, rewriting the right hand side of the above inequality we get the following:

$$\sum_{l=1}^{p} max\{t_l^L, \partial f_l{}^{-1}(\alpha_h)\} \geq \hat{T} > \sum_{l=1}^{p} max\{t_l^L, \partial f_l{}^{-1}(\alpha_h)\},$$

which is a contradiction.

Now let us prove the sufficiency: Assume $h \in J$ but to the contrary $\hat{T} \leq M_h$, for at least one operation h. Hence, $t_h^* > t_h^L$, which implies $\alpha_h = \partial f_h(t_h^L) < \partial f_h(t_h^*)$. From condition (i) of Lemma 8.2, $\partial f_h(t_h^*) = \partial f_l(t_l^*), \quad \forall h, l \in J$. Hence, the following must hold:

$$\alpha_h = \partial f_h(t_h^L) < \partial f_l(t_l^*), \ \ \forall l \in J.$$

Since f_l is convex and, it satisfies,

$$t_l^* > \partial f_l{}^{-1}(\alpha_h). \tag{8.14}$$

$\hat{T}$ can be written as:

$$\hat{T} = \sum_{l \in \bar{J}} t_l^L + \sum_{l \in J} t_l^*.$$

Using this with inequality 8.14 we get:

$$\hat{T} = \sum_{l \in \bar{J}} t_l^L + \sum_{l \in J} t_l^* > \sum_{l \in \bar{J}} t_l^L + \sum_{l \in J} \partial f_l{}^{-1}(\alpha_h).$$

Since $\hat{T} \leq M_h$ is assumed, rewriting the left hand side of the above inequality we get the following:

$$\sum_{l=1}^{p} max\{t_l^L, \partial f_l{}^{-1}(\alpha_h)\} < \hat{T} \leq \sum_{l=1}^{p} max\{t_l^L, \partial f_l{}^{-1}(\alpha_h)\},$$

which is a contradiction. This completes the proof. □

This lemma suggests that there is a breakpoint M_l for each operation l that can be calculated easily from the given cost functions and processing time lower bounds. For any operation, we can determine whether this operation is

an element of set J or set $\bar{J}$ by comparing the breakpoint of this operation with the given cycle time value $\hat{T}$. The processing times of the operations that are in set $\bar{J}$ are set to their lower bounds in the optimal solution. After this preprocessing, we need to determine the optimal processing times of the remaining operations that are in set J. Following lemma will be used for this purpose. Let $\Gamma = \sum_{h \in \bar{J}} t_h^L$.

Lemma 8.4 *In the optimal solution to the* $\epsilon(F_1|\hat{T})^{S_1^2}$*, the processing times of the operations in set* J *satisfy the following system of nonlinear equations:*

1. $t_l^* = \partial {f_l}^{-1}(\partial f_k(t_k^*)), \quad \forall l, k \in J,$
2. $\sum_{l \in J} t_l^* = \hat{T} - \Gamma.$

Proof. As a consequence of Lemma 8.3, for a given value of $\hat{T}$, we can determine which operations are in J and which are in $\bar{J}$ in the optimal solution. Additionally, Lemma 8.1 suggests that the cycle time bound constraint is satisfied as equality in the optimal solution. As a result, $\epsilon(F_1|\hat{T})^{S_1^2}$ reduces to the following:

$$\begin{aligned} \min \quad & \textstyle\sum_{l \in J} f_l(t_l) & & \\ \text{Subject to} \quad & \textstyle\sum_{l \in J} t_l = \hat{T} - \Gamma, & & (8.15) \\ & t_l \geq t_l^L, & \forall l \in J. & (8.16) \end{aligned}$$

Note that any feasible vector $\boldsymbol{t^*}$ is regular for $\hat{T} > \hat{T}^L$. Since the objective function and the constraints are convex, any point satisfying the KKT conditions is optimal. Hence, we have $\partial f_l(t_l^*) = -\lambda$, $\forall l \in J$. From here we get, $t_l^* = \partial {f_l}^{-1}(-\lambda)$. Hence, the processing time of any operation $l \in J$ can be represented in terms of another operation $k \in J$ as follows:

$$t_l^* = \partial {f_l}^{-1}(\partial f_k(t_k^*)).$$

The processing times of all operations can be found using this equation and Equation (8.15) jointly. □

Note that the system of nonlinear equations mentioned in the above lemma can be solved using some search methods such as the bisection algorithm, the Newton's method or the golden search algorithm within an error bound. As a consequence of lemmas 8.3 and 8.4, we can solve $\epsilon(F_1|\hat{T})^{S_1^2}$ easily. This solution corresponds to one of the nondominated solutions on the efficient frontier. We can state an algorithm that determines a total of r nondominated solutions, where the c^{th} solution has the following cycle time value:

$$\hat{T}^L + (c-1)\frac{\hat{T}^U - \hat{T}^L}{r-1} \ . \tag{8.17}$$

In this way it is guaranteed that the set of nondominated solutions that we find is uniformly spread over the entire efficient frontier.

Algorithm Efficient Frontier for S_1^2 (EFFRONT-S_1^2)
INPUT: r, O, t_l^L, $f_l(\cdot)$, $\forall l \in O$.
OUTPUT: t_{lc}^*, $l \in O$ with corresponding cycle time $\hat{T}_c$ and cost C_c values for $c = 1, 2, \ldots, r$.

1. Set solution counter $c = 1$.
2. Calculate t_l^U satisfying $\partial f_l(t_l^U) = 0$, $\forall l \in O$.
3. Using Equation (8.12) determine M_l, $\forall l \in O$.
4. $\hat{T}_c \leftarrow \hat{T}^L + (c-1)\frac{\hat{T}^U - \hat{T}^L}{r-1}$, ($\hat{T}^L$ and $\hat{T}^U$ are determined using Equations (8.10) and (8.11), respectively).
5. Call **SIMAM($O, \hat{T}_c, M_l, f_l(\cdot)$)**. Let t_l^* be the output of SIMAM, $l \in O$.
6. $t_{lc}^* \leftarrow t_l^*$, $l \in O$. Calculate $C_c = \sum_{l \in O} f_l(t_l^*)$. Output t_{lc}^*, $\hat{T}_c$ and C_c, $l \in O$,

7. $c \leftarrow c + 1$,

8. If $c \leq r$, go to Step 4. Else, STOP.

Subroutine SIMAM

INPUT: $O' \subseteq O$, T, M_l, $f_l(\cdot)$, $l \in O'$.

OUTPUT: t_l^*, $l \in O'$.

1. Determine sets J and $\bar{J}$ using T and M_l, $l \in O'$ according to Lemma 8.3. Set $t_h^* = t_h^L$, $\forall h \in \bar{J}$,

2. Calculate $\Gamma = \sum_{h \in \bar{J}} t_h^L$,

3. Solve the following nonlinear equation as stated in Lemma 8.4 to determine $t_{\hat{k}}^*$ for an arbitrary $\hat{k} \in J$:

$$T - \Gamma = \sum_{l \in J} \partial {f_l}^{-1}(\partial f_{\hat{k}}(t_{\hat{k}})),$$

4. Determine $t_l^* = \partial {f_l}^{-1}(\partial f_{\hat{k}}(t_{\hat{k}}^*))$, $\forall l \in J$, $l \neq \hat{k}$,

5. Output t_l^*, $l \in O'$ and return.

This algorithm generates a total of r nondominated solutions that are uniformly spread on the efficient frontier. The SIMAM subroutine is called to determine the optimal processing times of each of the r solutions. This subroutine will also be used while determining the efficient frontier of the S_2^2 cycle.

Till now the manufacturing cost function is considered to be any strictly convex function. The following example considers the CNC turning operations, which possess a strictly convex nonlinear cost function.

Example 8.1 Let us consider a 2-machine robotic cell with CNC turning machines. As stated in the previous chapter, manufacturing cost for CNC turning operations can be written as: $f_l(t_l) = C_o t_l + K_l U_l t_l^{a_l}$. The optimal processing time of an operation $\hat{k} \in J$ can be determined by solving the following nonlinear equation for $t_{\hat{k}}$:

$$\hat{T} - \Gamma = \sum_{l \in J} t_{\hat{k}}^{\frac{a_{\hat{k}}-1}{a_l-1}} \left(\frac{K_{\hat{k}} U_{\hat{k}} a_{\hat{k}}}{K_l U_l a_l} \right)^{\frac{1}{a_l-1}}.$$

Then t_l^* can be determined using $t_{\hat{k}}^*$ by solving the following equation:

$$t_l^* = t_{\hat{k}}^{*\left(\frac{a_{\hat{k}}-1}{a_l-1}\right)} \left(\frac{K_{\hat{k}} U_{\hat{k}} a_{\hat{k}}}{K_l U_l a_l} \right)^{\frac{1}{a_l-1}}, \quad \forall l \in J.$$

If all of the operations use the same tool type, then $K_l = K_k = K$ and $a_k = a_l = a$, $\forall l, k$. As a consequence, the optimal processing times of the operations that are in set J can be determined using the following closed form expression:

$$t_k^* = \frac{(\hat{T} - \Gamma)(U_k)^{\frac{1}{1-a}}}{\sum_{l \in J} U_l^{\frac{1}{1-a}}}, \quad \forall k \in J.$$

In the next section we will consider the S_2^2 cycle.

8.3 Heuristic Procedure for the S_2^2 Cycle

In this section we will consider the S_2^2 cycle for which, unlike in the previous section, we also have to deal with the allocation problem. In this section, the allocation of the operations to the two machines means partitioning set O into two subsets O_1, O_2 such that $O_1 \cup O_2 = O$ and $O_1 \cap O_2 = \emptyset$. O_i denotes the set of operations that are allocated to machine i, $i = 1, 2$. The total processing time of the part on machine i is $P_i = \sum_{l \in O_i} t_l$, $i = 1, 2$. Recall that in Theorem 6.2 we proved that the operation allocation problem for the S_2^2 cycle itself is

NP-complete. Hence, we will develop a heuristic procedure that approximates the efficient frontier and perform a computational study to verify its solution quality. We will compare the results of the heuristic procedure by solving the epsilon-constraint problem using nonlinear mixed integer problem solvers GAMS-DICOPT2x-C and GAMS-BARON 7.2.3. Before proceeding with the heuristic procedure let us first consider the mathematical formulation of the problem. For the S_2^2 cycle we will use the following as Constraint (8.2) of $\epsilon(F_1|T)$:

$$max\{6\epsilon + 8\delta, 4\epsilon + 4\delta + \sum_{l=1}^{p} x_{l1}t_l, 4\epsilon + 4\delta + \sum_{l=1}^{p} x_{l2}t_l\} \leq T,$$

which can be replaced by the following constraints:

$$6\epsilon + 8\delta \leq T,$$

$$4\epsilon + 4\delta + \textstyle\sum_{l=1}^{p} x_{li}t_l \leq T, \quad i = 1, 2.$$

Note that the second constraint is nonlinear. Let N_l denote a big number. By replacing $x_{li}t_l$ with w_{li}, we can properly linearize the above constraints as follows:

$$w_{li} \geq t_l - N_l(1 - x_{li}),$$

$$w_{li} \leq t_l + N_l(1 - x_{li}),$$

$$w_{li} \leq N_l x_{li},$$

$$w_{li} \geq 0.$$

Note that, N_l must be greater than t_l. Hence, we will use $N_l = t_l^U$. As a result, the epsilon-constraint problem for the S_2^2 cycle can be formulated as follows:

$$\begin{aligned}
\boldsymbol{\epsilon(F_1|T)^{S_2^2}}\text{:} \quad \min \quad & \sum_{l=1}^{p} f_l(t_l) \\
\text{s.t.} \quad & 6\epsilon + 8\delta \leq T, && (8.18)\\
& 4\epsilon + 4\delta + \sum_{l=1}^{p} w_{li} \leq T, \quad \forall i \in [1, 2], && (8.19)\\
& w_{li} \geq t_l - t_l^U(1 - x_{li}), \quad \forall l \in [1, \ldots, p], \forall i \in [1, 2], && (8.20)
\end{aligned}$$

$$w_{li} \leq t_l + t_l^U(1 - x_{li}), \quad \forall l \in [1, \ldots, p], \forall i \in [1, 2], \quad (8.21)$$

$$w_{li} \leq t_l^U x_{li}, \quad \forall l \in [1, \ldots, p], \forall i \in [1, 2], \quad (8.22)$$

$$x_{l1} + x_{l2} = 1, \quad \forall l \in [1, \ldots, p], \quad (8.23)$$

$$t_l \geq t_l^L, \quad \forall l \in [1, \ldots, p], \quad (8.24)$$

$$x_{li} \in \{0, 1\}, \quad \forall l \in [1, \ldots, p], \forall i \in [1, 2], \quad (8.25)$$

$$w_{li} \geq 0, \quad \forall l \in [1, \ldots, p], \forall i \in [1, 2]. \quad (8.26)$$

This mathematical problem is a Mixed Integer Nonlinear Problem (MINLP) which allocates the operations to both machines and determines processing time values of all operations satisfying a given cycle time value while minimizing the total manufacturing cost.

Let O_i^*, $i = 1, 2$ denote the set of operations that are allocated to machine i in the optimal solution to $\epsilon(F_1|T)^{S_2^2}$. The following lemma characterizes some properties of the optimal solution. Let $\boldsymbol{t^*} = (t_1^*, \ldots, t_p^*)$ denote the vector of optimal processing times to $\epsilon(F_1|T)^{S_2^2}$ throughout this section.

Lemma 8.5 *Either one of the following two properties holds:*

1. $\sum_{l \in O_1^*} t_l^* = \sum_{l \in O_2^*} t_l^*$ *or,*
2. *If* $\sum_{l \in O_1^*} t_l^* < \sum_{l \in O_2^*} t_l^*$, *then* $t_l^* = t_l^U$, $\forall l \in O_1^*$. *Else* $t_l^* = t_l^U$, $\forall l \in O_2^*$.

Proof. Let $\boldsymbol{t^*}$ be the optimal processing time vector with optimal objective function value F_1^*. Assume to the contrary that $\sum_{l \in O_1^*} t_l^* < \sum_{l \in O_2^*} t_l^*$ and $t_{\hat{l}}^* < t_{\hat{l}}^U$, for at least one operation $\hat{l} \in O_1^*$. Then we have another feasible solution such that $\hat{t}_l^* = t_l^*$, $\forall l \neq \hat{l}$ and $\hat{t}_{\hat{l}}^* = t_{\hat{l}}^* + \beta$ for $0 < \beta \leq min\{\sum_{l \in O_2^*} t_l^* - \sum_{l \in O_1^*} t_l^*, t_{\hat{l}}^U - t_{\hat{l}}^*\}$. Since the cost function is decreasing, $\hat{F}_1^* < F_1^*$. This contradicts with $\boldsymbol{t^*}$ being an optimal solution. □

The cycle time of the S_2^2 cycle is determined by the machine which has the greatest processing time. The other machine has no effect on the cycle time. Hence, the cycle time does not change by increasing the processing times of the operations on the other machine as long as the total processing time of this machine does not exceed the other machine. However, such a change reduces the cost. Hence, the lemma above states that in the optimal solution to $\epsilon(F_1|T)^{S_2^2}$, the operations are allocated such that the total processing times on both machines are equal to each other. However, since the processing times have upper bounds, the processing times on both machines may not be optimal in some cases. Assume without loss of generality that $\sum_{l\in O_1^*} t_l^* < \sum_{l\in O_2^*} t_l^*$. Then we know that, in the optimal solution $\sum_{l\in O_2^*} t_l^* - \sum_{l\in O_1^*} t_l^*$ is minimum and $t_l^* = t_l^U, \forall l \in O_1^*$. As a consequence, if the processing times are not decision variables but predetermined parameters, the optimal allocation of operations to the machines can be determined by solving the following Allocation Problem which is a Mixed Integer Problem (MIP) formulation:

$$
\begin{aligned}
\textbf{(AP)} \quad & \min \quad && T \\
& \text{Subject to} \quad && 6\epsilon + 8\delta \leq T, && (8.27)\\
& && 4\epsilon + 4\delta + \sum_{l=1}^{p} x_{li}\hat{t}_l \leq T, \quad i = 1, 2, && (8.28)\\
& && x_{l1} + x_{l2} = 1, \quad \forall l, && (8.29)\\
& && x_{l1}, x_{l2} \in \{0, 1\}, \quad \forall l. && (8.30)
\end{aligned}
$$

In this formulation, $\hat{t}_l$ is not a decision variable but a parameter. We can make use of this formulation in order to determine the upper and lower bounds of the cycle time of the S_2^2 cycle. Let $T_{S_2^2}^U$ and $T_{S_2^2}^L$ denote the upper and lower bounds for the cycle time of S_2^2, respectively. Furthermore, let $t^L = (t_1^L, t_2^L, \ldots, t_p^L)$ and $t^U = (t_1^U, t_2^U, \ldots, t_p^U)$ be the vectors of lower and upper bounds of the processing times of operations, respectively. Let T_L^* denote the optimal objective function value of the AP where $\hat{t}_l = t_l^L, \forall l$. Then,

$T^L_{S_2^2} = T^*_L$. Furthermore, if the processing times on both of the machines are equal to each other in the optimal solution, then from Lemma 8.5, the point $(F_1(\boldsymbol{t^L}), F_2(\boldsymbol{t^L}))$, where $F_2(\boldsymbol{t^L}) = T^*_L$ is one of the nondominated solutions of the bicriteria formulation for S_2^2. This is the minimum cycle time-maximum cost solution. On the other hand, if the processing times on both machines are not equal to each other, according to Lemma 8.5 the point $(F_1(\boldsymbol{t^L}), F_2(\boldsymbol{t^L}))$ is dominated. Hence, we can conclude that solving the above AP provides the lower bound of the cycle time. Additionally, if the processing times on both machines are equal to each other, we get the nondominated solution corresponding to the minimum cycle time value. However, if the processing times are not equal to each other, in order to get the nondominated solution, we have to solve the epsilon-constraint problem by setting the cycle time bound to $T^L_{S_2^2}$, that is $\epsilon(F_1|T^L_{S_2^2})^{S_2^2}$. Similarly the upper bound of the cycle time can be found by solving the AP by setting $\hat{t}_l = t^U_l$, $\forall l$. The optimal objective function value of this formulation is the upper bound of the cycle time, $F_2(\boldsymbol{t^U})$. Note that, according to Case (2) of Lemma 8.5, the point $(F_1(\boldsymbol{t^U}), F_2(\boldsymbol{t^U}))$ is a nondominated solution for the bicriteria problem, which corresponds to the maximum cycle-time minimum cost pair.

From Constraint (8.27) of the AP, the cycle time can not be less than $6\epsilon+8\delta$. As a consequence, from Constraint (8.28) of the AP, the total processing times on any one of the machines is not less than $2\epsilon + 4\delta$ in the optimal allocation unless the processing times of the operations allocated to the same machine are not set to their upper bounds. We can write this more formally as in the following lemma. Let x^*_{li} be the optimal solution to the AP where all processing times are set to their upper bounds, $\hat{t}_l = t^U_l$, $\forall l \in O$.

Lemma 8.6 *If* $\sum_{l=1}^{p} x^*_{li} t^U_l \leq 2\epsilon + 4\delta$, $i = 1, 2$, *the point* $(F_1(\boldsymbol{t^U}), F_2(\boldsymbol{t^U}))$, *where* $F_1(\boldsymbol{t^U}) = \sum_{l=1}^{p} f_l(t^U_l)$ *and* $F_2(\boldsymbol{t^U}) = 6\epsilon + 8\delta$ *dominates all other feasible*

solutions.

Proof. If $\sum_{l=1}^{p} x_{li}^{*} t_l^U \leq 2\epsilon+4\delta$, $i = 1, 2$, then Constraint (8.27) of AP is binding in the optimal solution. Hence, the cycle time can only be equal to $6\epsilon + 8\delta$. Furthermore, $\sum_{l=1}^{p} f_l(t_l^U)$ is the lower bound for the cost. Any other solution will have a greater cost value, which means that it is dominated. □

The cycle time of the S_2^2 cycle can not be less than $6\epsilon + 8\delta$. This is the time required for the robot to perform all the necessary loading/unloading and transportation operations. In this cycle, while the processing on one machine continues, the robot does not wait in front of this machine, but loads/unloads the other machine. Hence the processing time of the operations on this machine can be increased so that the processing of all operations are completed when the robot arrives to unload this machine. Such a change does not increase the cycle time but reduces the cost. However, if the upper bounds of the processing times of the operations are so small such that the total processing times on both machines are completed before the robot arrives in front of the machines even if all processing times are set to their upper bounds, then there is a unique nondominated solution, as stated in the lemma above.

In order to estimate the efficient frontier we will determine r uniformly spread nondominated solutions. The quality of the estimation depends on the magnitude of r. Since the allocation problem is NP-complete, solving even only a single problem to optimality using a nonlinear mixed integer solver will require a significant amount of CPU time. Because of this, we will present a heuristic algorithm. This algorithm generates a new nondominated solution using the solution at hand. Starting from $T_{S_2^2}^L$, the cycle time value is incremented at each step until we reach $T_{S_2^2}^U$. Instead of using a fixed increment amount, a new increment amount is determined dynamically at each step depending on problem characteristics. As a result, the number of

nondominated solutions generated is unknown prior to running the algorithm, but it is guaranteed that the generated solutions are spread in the range between $(F_1(\boldsymbol{t^L}), F_2(\boldsymbol{t^L}))$ and $(F_1(\boldsymbol{t^U}), F_2(\boldsymbol{t^U}))$.

The problem is twofold: finding the allocation of the operations to the machines and determining the processing time values of the operations. According to Lemma 8.5, for fixed processing times the former of these problems is identical to set partitioning problem, for which we will make use of the *Difference Method* (DM) developed by Karmarkar and Karp [58]. Let $D = \{a_1, a_2, \ldots a_p\}$ be a set of numbers to be partitioned. Let

$$D' = D \setminus \{a_j, a_k\} \cup \{|a_j - a_k|\}.$$

From a partition (A', B') of D' a partition (A, B) of D can be constructed easily so that both partitions have identical differences. Suppose $a_j > a_k$ and $|a_j - a_k| \in A'$. Then

$$\begin{aligned} A &= A' \setminus \{|a_j - a_k|\} \cup \{a_j\}, \\ B &= B' \cup \{a_k\}, \end{aligned} \tag{8.31}$$

gives the desired partition. Then, the Difference Method does the following:

Difference Method (DM))

INPUT: A set of numbers to be partitioned: D.

OUTPUT: A partition (A, B) of set D.

While $|D| > 1$ do begin

pick the largest two numbers a_j, $a_k \in D$.

$D \leftarrow D \setminus \{a_j, a_k\} \cup \{|a_j - a_k|\}$.

end

Do the backtracking operations as in Equation (8.31).

The algorithm terminates when $|D| = 1$. It is trivial to construct the actual partition (A, B) by backtracking through the sequence of differencing operations as in Equation (8.31). The following example will be helpful in understanding.

Example 8.2 Let us consider partitioning set $D = \{7, 4, 8, 10, 3\}$. We can present the DM algorithm as follows:

Step	Ordered Set	$a_j - a_k$
1	$\{10, 8, 7, 4, 3\}$	10-8=2
2	$\{7, 4, 3, 2\}$	7-4=3
3	$\{3, 3, 2\}$	3-3=0
4	$\{2, 0\}$	2-0=2
5	$\{2\}$	STOP

The resulting set with only 2 as the element in Step 5 of the algorithm is found by differencing operation (2-0=2). Hence, the partition in Step 4 which has identical difference as Step 5 is ($\{0\}$, $\{2\}$). In a similar way we can backtrack all the differencing operations as follows:

Step	Partition	
5	$\emptyset$	$\{2\}$
4	$\{0\}$	$\{2\}$
3	$\{3\}$	$\{2, 3\}$
2	$\{3, 4\}$	$\{2, 7\}$
1	$\{3, 4, 8\}$	$\{7, 10\}$

As already mentioned, we will use a different increment value at each step of the algorithm in order to generate a new nondominated solution. More specifically, let T_c be the cycle time value of the current nondominated solution. Then a new solution will be determined with a cycle time value of $T_c + inc$, where inc is the amount of increment calculated for that step. We will determine two candidates and select the minimum of these as the the increment value. One of these candidates will be calculated considering the DM algorithm. Recall that this algorithm works with a set of fixed numbers. However, in our case the processing times which are to be partitioned are not fixed but are decision variables. In such a case, we have to determine the sensitivity of the algorithm. That is, for each processing time t_l, let β_l be calculated such that adjusting the processing time as $t_l + \lambda$, does not change the allocation resulting from the DM algorithm for $0 < \lambda \leq \beta_l$, but it does for $\lambda > \beta_l$. These breakpoints are determined considering the ordering of the numbers at each step. The following example illustrates the procedure for determining the first breakpoint.

Example 8.3 Let us consider the same example above with $a_1 = 10$, $a_2 = 8$, $a_3 = 7$, $a_4 = 4$ and $a_5 = 3$. In Step 1 we have the following descending order of numbers: (10-8-7-4-3). Incrementing $a_1 = 10$ does not yield a change in the ordering of the operations. However, incrementing a_2 more than $a_1 - a_2 = 10 - 8 = 2$ yields a new order and probably a new partition. Thus, $\beta_{21} = 2$, where β_{lj} is the bound for a_l at step j. At this step we have $\beta_{11} = \infty$, $\beta_{21} = 2$, $\beta_{31} = 1$, $\beta_{41} = 3$ and $\beta_{51} = 1$. In Step 2, a_1 and a_2 are removed from the set but a new element $(a_1 - a_2)$ is included. The new ordering of the numbers at this step is: (7-4-3-2). The bounds in this step are as follows: $\beta_{32} = \infty$, $\beta_{42} = 3$, $\beta_{52} = 1$. Note that, the last element, 2, is found as $a_1 - a_2 = 10 - 8 = 2$. Incrementing this number by 1 leads to a new partition. Hence, 1 is a bound for $a_1 \Rightarrow \beta_{12} = 1$. On the other hand, decrementing this by 2 leads to a

negative number. Decrementing this number means incrementing a_2. Hence, 2 is a bound for $a_2 \Rightarrow \beta_{22} = 2$. Proceeding similarly the following bounds can be determined at each step:

Step	Bounds				
1	$\beta_{11} = \infty$	$\beta_{21} = 2$	$\beta_{31} = 1$	$\beta_{41} = 3$	$\beta_{51} = 1$
2	$\beta_{12} = 1$	$\beta_{22} = 2$	$\beta_{32} = \infty$	$\beta_{42} = 3$	$\beta_{52} = 1$
3	$\beta_{13} = 1$	$\beta_{23} = 2$	$\beta_{33} = 0$	$\beta_{43} = 0$	$\beta_{53} = 0$
4	$\beta_{14} = \infty$	$\beta_{24} = 2$	$\beta_{34} = 2$	$\beta_{44} = 0$	$\beta_{54} = 2$

Then the overall bound for element l is $\beta_l = min_{j,\beta_{lj}\neq 0}\{\beta_{lj}\}$. Note that, in some steps, the bounds may be equal to 0. A bound may be 0 if there are alternative partitions with identical differences. These are not considered while calculating the overall bounds for the numbers. Furthermore, the processing times to be partitioned have upper bounds. Let a_l^U be the upper bound of a_l. As a result we have: $0 < \beta_l \leq a_l^U - a_l$. Finally, the candidate for the increment value is $min_l\{\beta_l\}$. This value is selected as a candidate because an increment less than this value does not yield a different partition but a greater increment yields a new partition according to the DM algorithm. Hence, this candidate for the increment value determines whether we need to run the DM algorithm or not in order to generate the new solution.

The second candidate for the increment comes from Lemma 8.3. This lemma determines whether the processing time of an operation is equal to its lower bound or greater than it. M_l values, calculated for each operation l, are used for this decision. However, the calculation of the M_l values for the S_2^2 cycle differs from the calculation of those for the S_1^2 cycle since allocation does not matter for the S_1^2 cycle. However, we can calculate M_l values for a given allocation as follows: $M_l = \sum_{h\in O_i} max[t_h^L, \partial {f_h}^{-1}(\partial f_l(t_l^L))]$, $\forall l \in O_i, i = 1, 2$.

From Constraint (8.19) of $\epsilon(F_1|T)^{S_2^2}$ the sum of the processing times allocated to the same machine must be less than or equal to $T_c - 4\epsilon - 4\delta$. Then if $M_l > T_c - 4\epsilon - 4\delta$ for an arbitrary operation l, then $t_l^* = t_l^L$. Assume that the cycle time is incremented so that $T_{new} = T_c + \lambda$. If $\lambda \leq M_l - T_c - 4\epsilon - 4\delta$, then $t_l^* = t_l^L$. Otherwise, $t_l^* > t_l^L$. As a consequence, the second candidate is calculated as $\Delta_T = T_c - 4\epsilon - 4\delta - max_{l\in\bar{J}}\{M_l\}$. As a result, the increment value for the cycle time to determine a new nondominated solution is determined as: $min\{\Delta_T, min_l\{\beta_l\}\}$. However, in some cases this increment value becomes very small especially when the number of operations is high. As a result, a set of unnecessarily large number of solutions is computed by the heuristic, resulting in a relatively large CPU time. In order to avoid this, the increment value is bounded below by $\omega = E * 10^{\ln \frac{p}{10}}$. By this bound, the number of points generated by the algorithm and the CPU time requirements are balanced for problems with small number of operations and large number of operations. The bound can be adjusted by altering the constant E.

The following heuristic algorithm generates a number of solutions on the efficient frontier of the S_2^2 cycle.

Algorithm Efficient Frontier for S_2^2 (EFFRONT-S_2^2)

INPUT: O, ω, t_l^L, $f_l(\cdot)$, $l \in O$.

OUTPUT: t_{lc}^*, $l \in O$ with corresponding cycle time T_c and cost C_c values and allocation of operations to machines (O_{1c}, O_{2c}), for each nondominated solution $c = 1, 2, \ldots, r$, and the total number of nondominated solutions r.

1. Set solution counter $c = 1$.
2. Calculate t_l^U satisfying $\partial f_l(t_l^U) = 0$, $l \in O$.
3. Call $\mathbf{DM(\{t_1^U, t_2^U, \ldots, t_p^U\})}$. Let (O_1, O_2) be the output.

   3.1. If $\sum_{l\in O_i} t_l^U \leq 2\epsilon + 4\delta$, $i = 1, 2$, then according to Lemma 8.6, there

is a unique nondominated solution with $T^U = 6\epsilon + 8\delta$ and $C^L = \sum_{l \in O} f_l(t_l^U)$. Output t_l^U, T^U, C^L, (O_1, O_2), c. STOP.

3.2. Else, $T^U = max_i\{\sum_{l \in O_i} t_l^U + 4\epsilon + 4\delta\}$, $C^L = \sum_{l=1}^{p} f_l(t_l^U)$.

4. Call $\mathbf{DM(t_1^L, t_2^L, \ldots, t_p^L\})$. Let (O_1, O_2) be the output. Set $O_{ic} = O_i$, $i = 1, 2$ and $t_{lc} = t_l^L$, $\forall l \in O$.

4.1. Let $P_{ic} = \sum_{l \in O_{ic}} t_{lc}$, $i = 1, 2$. If $P_{ic} < 2\epsilon + 4\delta$, $i = 1, 2$, then go to Step 5.

4.2. Else,

4.2.1. Calculate M_l for $l \in O_1$ and $l \in O_2$ independently using Equation (8.12).

4.2.2. Call **Calculate-t**$((\boldsymbol{O_{1c}, O_{2c}}), \boldsymbol{t_{lc}}, \boldsymbol{M_l}, \boldsymbol{f_l(\cdot)})$. Let t_l be the output.

4.2.3. Set $t_{lc} = t_l$, $\forall l \in O$. Calculate $T_c = max_i\{P_{ic} + 4\epsilon + 4\delta\}$, $C_c = \sum_{l \in O} f_l(t_{lc})$. Output t_{lc}, T_c, C_c, (O_{1c}, O_{2c}). Let $c \leftarrow c + 1$.

5. Calculate M_l, for $l \in O_1$ and $l \in O_2$ independently using Equation (8.12).

6. If $P_{ic} < 2\epsilon + 4\delta$, $i = 1, 2$, then set $\Delta_T = min_i\{2\epsilon + 4\delta - P_{ic}\}$, else $\Delta_T = min_i\{P_{ic} - max_{l \in \bar{J}}\{M_l\}\}$, $i = 1, 2$. Determine breakpoints β_l, $l \in O$ as explained in Example 8.3.

6.1. If $\Delta_T < min_l\{\beta_l\}$,

6.1.1. $inc = min\{\Delta_T, \omega\}$.

6.1.2. Set $T_i = min\{\sum_{l \in O_{ic}} t_{lc}, P_{ic} + inc\}$, $i = 1, 2$.

6.1.3. Call **SIMAM(** $\boldsymbol{O_{1c}, T_1, M_l, f_l(\cdot)}$**)** and **SIMAM(** $\boldsymbol{O_{2c}, T_2, M_l, f_l(\cdot)}$**)**. Let t_l be the output of these. Set $t_{lc} = t_l$, $\forall l \in O$.

6.2. Otherwise,

6.2.1. $inc = min\{min_l\{\beta_l\}, \omega\}$.

6.2.2. Let $l^* \in O_{i^*} = argmin_l\{\beta_l\}$. Set $T_{i^*} = P_{i^*} + inc$. Call **SIMAM(O_{i^*}, T_{i^*}, M_l, $f_l(\cdot)$)**.

6.2.3. Call **DM($\{t_{1c}, t_{2c}, \ldots t_{pc}\}$)**. Let (O_1, O_2) be the output. Set $O_{ic} = O_i$, $i = 1, 2$.

6.2.3.1. If $P_{ic} < 2\epsilon + 4\delta$, $i = 1, 2$, then go to Step 5.

6.2.3.2. Else, Call **Calculate-t**((O_{1c}, O_{2c}), t_{lc}, M_l, $f_l(\cdot)$). Let t_l be the output. Set $t_{lc} = t_l$, $\forall l \in O$.

7. Calculate $T_c = max_i\{P_{ic} + 4\epsilon + 4\delta\}$, $C_c = \sum_{l \in O} f_l(t_{lc})$. Output t_{lc}, T_c, C_c, (O_1, O_2).

8. If $T_c = T^U$, output c. STOP. Else, let $c \leftarrow c + 1$, go to Step 6.

Subroutine Calculate-t

INPUT: (O_1, O_2), t_l, M_l, $f_l(\cdot)$, $l \in O$.

OUTPUT: t_l, $\forall l \in O$.

1. If $\sum_{l \in O_1} t_l = \sum_{l \in O_2} t_l$, output t_l, $\forall l \in O$, return.

2. Else if $\sum_{l \in O_1} t_l > \sum_{l \in O_2} t_l$, then Call **SIMAM(O_2, $\sum_{l \in O_1} t_l$, M_l, $f_l(\cdot)$)**, output t_l, $\forall l \in O$, return.

3. Else if $\sum_{l \in O_1} t_l < \sum_{l \in O_2} t_l$, then Call **SIMAM(O_1, $\sum_{l \in O_2} t_l$, M_l, $f_l(\cdot)$)**, output t_l, $\forall l \in O$, return.

This heuristic algorithm determines a set of nondominated solutions by generating a new point from an initial point. An increment value for the cycle time is determined and the corresponding allocation of the operations and the processing time values are determined. In Step 2, the largest cycle time-smallest cost solution is determined by setting all processing times to their upper bounds and allocating them to the machines. Similarly, in Step

3, the smallest cycle time-largest cost alternative is determined by setting all processing times to their lower bounds and allocating the operations to the machines. If the processing times on both machines are not equal to each other, the processing times of the operations are updated using the SIMAM algorithm. In Step 4 breakpoints are determined in order to determine sets J and $\bar{J}$ using Lemma 8.3. In Steps 5, 6 and 7 the increment value of the cycle time to find the next efficient point is determined. The increment value is bounded by $E * 10^{\ln \frac{p}{10}}$. The reason for using such a bound is that, the increment values start to be too small after some iterations of the algorithm and especially when the number of operations is high. As a result, a set of unnecessarily large number of solutions is determined by the heuristic, resulting in a relatively large CPU time. By this bound, the number of points generated by the algorithm and the CPU time requirements are balanced for problems with small number of operations and large number of operations. The bound can be adjusted by altering the constant E. The difference between the current cycle time value and the next largest breakpoint value determined in Step 4 and the bounds from the DM algorithm are calculated. The minimum of these values is selected as the increment value. If the increment is equal to the difference between the current cycle time value and the next largest breakpoint value, then the allocation is not changed and the new processing time values are determined using the SIMAM algorithm. Otherwise, the processing times on the pivot machine are updated using the SIMAM algorithm, operations are reallocated to the machines and using the SIMAM algorithm again the processing times are updated. The heuristic continues until the cycle time value for the next point to be determined is equal to the upper bound of the cycle time.

This heuristic procedure generates a set of nondominated solutions which are not necessarily equally spaced. However, since a large number of points

are generated, the distance between two consecutive points is very small and the points are spread to the entire efficient frontier. Additionally, the following lemma presents an important property of the EFFRONT-S_2^2 algorithm.

Lemma 8.7 *Any two solutions generated by the EFFRONT-S_2^2 algorithm can not dominate each other.*

Proof. The EFFRONT-S_2^2 algorithm generates a new solution starting with an initial solution. Let $T_c < T^U$ and C_c be the cycle time and cost values of a solution generated by the algorithm. Using this solution, a new solution is generated by setting $T_{(c+1)} = T_c + inc$, where $inc > 0$. Then, the EFFRONT-S_2^2 algorithm allocates this increment in cycle time to the individual processing times using the SIMAM algorithm as a subroutine, which guarantees the allocation to be the most cost reducing alternative (since the cost function is decreasing with respect to the processing times). As a result, $C_{(c+1)} < C_c$. This guarantees that $T_c < T_d$ and $C_c > C_d$ for any solution generated by the EFFRONT-S_2^2 algorithm, which means that the solutions generated by the EFFRONT-S_2^2 algorithm can not dominate each other and a new nondominated solution is generated at each iteration of the algorithm. □

The following small example illustrates the results generated after some iterations of the algorithm.

Example 8.4 Let us consider a CNC turning operation for which the manufacturing cost is presented in Example 8.1 and consider a problem with 5 operations. Let the parameters be given as follows:

l	t_l^L	t_l^U	$T_l * U_l$	a_l
1	1.2	4.7	15.87	-1.49
2	2	2.8	4.48	-1.56
3	1.8	5.6	23.72	-1.46
4	3.5	4.2	14.13	-1.70
5	2.2	3.4	6.66	-1.38

Table 8.1 presents the results of both the heuristic and MINLP solver GAMS-DICOPT2x-C after the first 5 iterations. (1^*) in this table, denotes the first point generated by allocating the operations to the machines with the processing times set to their lower bounds. Note that the second solution dominates this one since the two solutions have identical cycle time values whereas the second solution has a smaller manufacturing cost than the first one. Note that, both the allocation of the operations to the machines and the processing time values may change from one solution to another. Furthermore, as T increases, the processing times of some operations may decrease depending on the allocated machine. These properties show the complexity of the problem.

In the next section we will test the efficiency of our heuristic algorithm with an experimental design on problem parameters.

8.4 Computational Study

In this section we will test the EFFRONT-S_2^2 algorithm. This algorithm works for any strictly convex and differentiable function. In order to evaluate the algorithm we will consider CNC turning operations for which the cost function is presented earlier. We will compare the results of the EFFRONT-S_2^2 algorithm with the results of MINLP solvers GAMS-DICOPT2X-c and

			EFFRONT-S_2^2			DICOPT		
Iteration	T	l	Machine	t_l	Cost	Machine	t_l	Cost
1*	5.4	1	1	3.5	32.94	1	3.5	32.94
		2	2	2.2		2	2.2	
		3	2	2		2	2	
		4	1	1.8		1	1.8	
		5	2	1.2		2	1.2	
2	5.4	1	1	3.5	32.23	1	3.5	32.23
		2	2	2.2		2	2.2	
		3	2	2		2	2	
		4	1	1.9		1	1.9	
		5	2	1.2		2	1.2	
3	5.5	1	1	3.5	29.67	1	3.5	29.67
		2	2	2.2		2	2.2	
		3	1	2		1	2	
		4	2	1.8		2	1.8	
		5	2	1.5		2	1.5	
4	5.534	1	1	3.5	29.37	1	3.5	29.37
		2	2	2.2		2	2.2	
		3	1	2.034		1	2.034	
		4	2	1.8		2	1.8	
		5	2	1.534		2	1.534	
5	5.7	1	1	3.5	28.10	1	3.5	26.96
		2	2	2.2		1	2.2	
		3	1	2.2		2	2	
		4	2	1.89		2	2	
		5	2	1.61		2	1.7	

Table 8.1: Results of the first 5 iterations of EFFRONT-S_2^2 and DICOPT for Example 8.4

GAMS-BARON 7.2.3. DICOPT was developed at the Engineering Design Research Center at the Carnegie Mellon University. The MINLP algorithm inside DICOPT solves a series of NLP and MIP subproblems. Despite the speed of the algorithm, DICOPT is unable to prove the optimality and the quality of the solution provided. On the other hand, the Branch-And-Reduce Optimization Navigator (BARON) is a GAMS solver for the global solution of nonlinear (NLP) and mixed-integer nonlinear programs (MINLP). BARON implements deterministic global optimization algorithms of the branch-and-bound type that are guaranteed to provide global optima under fairly mild assumptions. However, BARON requires much more CPU time in comparison with DICOPT. We coded the EFFRONT-S_2^2 algorithm in C language and compiled with Gnu C compiler. The DICOPT and EFFRONT-S_2^2 methods were run on a computer with 512 MB memory and Pentium IV 3.00GHz CPU. Whereas due to licensing issues, the BARON method is run on a computer with 1294 MB memory and Pentium III 1133 MHz CPU.

There are four experimental factors that can affect the efficiency of the methods. These factors are presented in Table 8.4. The number of operations affects the problem size and thus the computational requirements. The most important parameters that affect the efficiency of the methods are t^L and t^U. Using factors B, C and D, t^L and t^U parameters are generated as follows:

$$t_l^U = U[C * D, D] \text{ and } t_l^L = B * t_l^U,$$

where $U[a, b]$ is a Uniform distribution on the interval $[a, b]$. Factors C and D affect the shape of the manufacturing cost function and this in turn affects the running times of the MINLP solvers. A small value for Factor B means a greater range between t^L and t^U and a small value for Factor C means greater variability for both t^L and t^U values in which case the MINLP solvers are expected to work better. Additionally, the t^U level is another important parameter that increases the importance of the allocation of the operations to

the machines. If the t^U level is greater, then the penalty of a poor allocation is higher in which case the performance of the DM algorithm used inside the EFFRONT-S_2^2 algorithm increases. In addition to these experimental design parameters, we assume identical CNC machines with operating cost $C_o = 0.5$. Consistent with earlier studies [63], a_l is selected from $U[-1.7, -1.3]$ and given these parameters, the required values for $K_l * U_l$ can be calculated using, $K_l * U_l = -\frac{C_o}{a_l (t_l^U)^{(a_l - 1)}}$, $\forall l \in \{1, \ldots, p\}$.

Factor	Definition	Level 1	Level 2	Level 3
A	Number of operations, (p)	20	50	80
B	$t^L - t^U$ range	0.5	0.8	
C	t^U variability	0.7	0.3	
D	t^U level	5	10	

Table 8.2: Experimental design factors

Also note that the robot transportation time δ and load/unload time ϵ do not have an effect on either the allocation of the operations to the machines or the processing times. However, as mentioned in the previous chapter, if these parameters are too large with respect to the processing times, this affects the values that the cycle time can attain. For example, after setting all processing times to their upper bounds and allocating them to both machines, if the processing times on both machines are less than $2\epsilon + 4\delta$, then the only value that the cycle time can take is $6\epsilon + 8\delta$. As a consequence of this, in order to test the efficiency of the EFFRONT-S_2^2 algorithm we will assume that $\epsilon = 0$ and $\delta = 0$. In this way, we guarantee that the heuristic will not stop in STEP 2 and a comprehensive test can be made.

Five replications are taken for each of the experimental settings, $(3*2*2*2)$, which makes a total of 120 different problem settings. This means, by this experimental design we will approximate a total of 120 efficient frontiers. In

order to approximate these, we will use 20 different cycle time bounds that are spread over the entire efficient frontier. Hence, a total of 2400 problems will be solved. Additionally, as already mentioned, the minimum cycle time-maximum cost solution is found by solving AP formulation with commercial MIP solver GAMS-CPLEX 9.1 by setting all processing times to their lower bounds. Let T^* be the optimal objective function value of the AP. The solution found by CPLEX is a nondominated solution if the processing times on both machines are equal to each other. However, if the processing times are not equal to each other, in order to determine the minimum cycle time-maximum cost solution, the ECP formulation is solved by using T^* as the cycle time bound. Similarly, the maximum cycle time-minimum cost solution is found by solving the AP formulation with CPLEX by setting all processing times to their upper bounds. However, unlike the previous case, the generated solution is a nondominated solution regardless of the processing times of the optimal solution. The remaining 18 problems are solved using a MINLP solver. Due to CPU restrictions, in order to find a good solution in a reasonable time we set $optcr = 0.05$, which is the relative optimality gap. That is, when $optcr = 0.05$, the MINLP model stops as soon as $\frac{|Best\ Found - Best\ Possible|}{Best\ Possible} \leq 0.05$. In such a case, even for very small problem instances, the model stops with an optimality gap. In order to catch these instances, we run the DICOPT model with $optcr = 0$ with a time limit of 900 seconds. If the run is not completed until this time limit, then we set $optcr = 0.05$, input the best integer solution found till that time as the starting solution and run the model again with a time limit of 3600 seconds this time. As a consequence, the solver either makes a normal completion before 900 seconds or stops with $optcr = 0.05$ between somewhere in [900, 4500] seconds after starting the run or stops due to time limit when it reaches to 4500 seconds. On the other hand, the BARON model is used only for $p = 20$ case. However, since we want to compare the results of the heuristic procedure with high quality results, in contrast with the

	DICOPT			BARON		
p	Normal	optcr = 0.05	Time Limit	Normal	optcr < 0.05	optcr = 0.05
20	800	-	-	65	594	141
50	154	612	34			
80	104	653	43			

Table 8.3: Completion statistics for DICOPT and BARON

DICOPT case, the BARON model is run for 1800 seconds initially. If the run is not completed but $optcr \leq 0.05$, then it is stopped immediately. Otherwise, the model is run until $optcr = 0.05$. Table 8.3 lists the number of instances with each stopping type for both MINLP methods. Note that all of the normal completions listed for the BARON method are actually achieved by the CPLEX solver for the AP formulations to determine the minimum cost-maximum cycle time and maximum cost-minimum cycle time solutions. None of the BARON runs are normally completed within the time limit. As expected, the number of normal completions decreases as the number of operations increases. For the DICOPT model, for $p = 20$ all 800 instances including the CPLEX runs for the AP formulations completed within the 900 seconds time limit, whereas this number reduced to 154 and 104 for $p = 50$ and $p = 80$, respectively.

The number of approximate efficient points generated by the EFFRONT-S_2^2 algorithm depends on the experimental design parameters as well as the manufacturing cost parameters and the step size constant E. In this study we will use $E = 0.0001$. Since the number of points to be generated by the EFFRONT-S_2^2 algorithm is not known in advance, in order to compare the results of different methods we first run the EFFRONT-S_2^2 algorithm and generate a set of points. Then we choose 18 of these other than the minimum cycle time-maximum cost and maximum cycle time-minimum cost solutions such that each successive point pair has (almost) equal separation

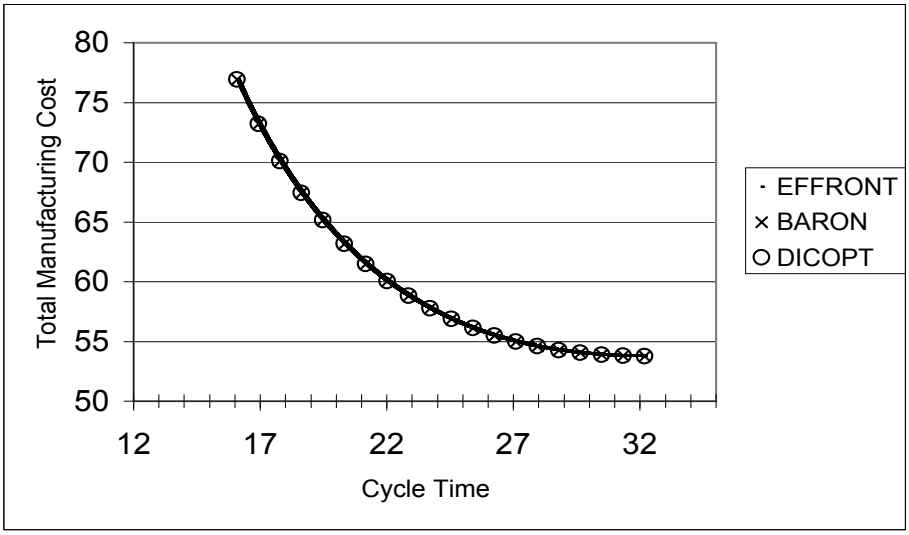

Figure 8.1: Comparison of points generated by the three methods

and run the MINLP models for the corresponding cycle time values of the 20 points. Figure 8.1 depicts the points generated by the EFFRONT-S_2^2 algorithm and the 20 points generated by the DICOPT and BARON solvers for 20 operations with the factor combination $(B, C, D) = (0.5, 0.3, 5)$. For this example, the EFFRONT-S_2^2 algorithm generated a total of 17327 approximate efficient points in 15.156 seconds. 18 points generated by DICOPT in addition to 2 points generated by CPLEX used a total of 1089.74 seconds, where for all 20 points the models made normal completions. Note that, normal completion with DICOPT does not necessarily mean the global optima is achieved but in most cases a local optimal solution is presented. On the other hand, BARON used 29574.17 seconds to generate the 20 points, 2 of which are generated by CPLEX. In 12 of the 18 points generated by BARON, the model stopped immediately after reaching 1800 seconds since $optcr \leq 0.05$ is achieved within this time. In the remaining 6 problems, the model ran until $optcr = 0.05$.

Let us use F_1 and F_2 instead of $F_1(S, \boldsymbol{P})$ and $F_2(S, \boldsymbol{P})$, respectively, for notational simplicity. We measured the relative difference between F_1 values of different methods for the same given F_2 values. Let

		R_1			R_2			R_3		
p		> 0	$= 0$	< 0	> 0	$= 0$	< 0	> 0	$= 0$	< 0
20	Number	90	6	704	387	2	411	699	101	0
	Min ($\times 10^{-6}$)	0,009	-	-0,006	0,007	-	-0,009	0,007	-	-
	Avg ($\times 10^{-6}$)	0,436	-	-31,402	0,785	-	-49,361	2,982	-	-
	Max ($\times 10^{-6}$)	6,721	-	-5722,0	8,337	-	-5722,0	67,585	-	-
50	Number	154	5	641						
	Min ($\times 10^{-6}$)	0,003	-	-0,003						
	Avg ($\times 10^{-6}$)	0,099	-	-2,435						
	Max ($\times 10^{-6}$)	0,341	-	-59,319						
80	Number	214	5	581						
	Min ($\times 10^{-6}$)	0,002	-	-0,002						
	Avg ($\times 10^{-6}$)	0,073	-	-26,095						
	Max ($\times 10^{-6}$)	0,245	-	-10653,6						

Table 8.4: Summary of results

$R_1 = (F_1(EFFRONT\text{-}S_2^2) - F_1(DICOPT))/F_1(DICOPT)$,
$R_2 = (F_1(EFFRONT\text{-}S_2^2) - F_1(BARON))/F_1(BARON)$,
$R_3 = (F_1(DICOPT) - F_1(BARON))/F_1(BARON)$.

The detailed results of all runs for each problem factor are presented in Tables G.1-G.5 in Appendix G. Tables 8.4 and 8.5 aggregate and summarize these results. Table 8.4 compares the methods in terms of their average, minimum and maximum R_1, R_2 and R_3 values. Comparisons with BARON can only be made for 20 operations case due to CPU time restrictions. The R_1 statistics show that, although the average difference is very small (0.203×10^{-6}), in most of the cases the EFFRONT-S_2^2 algorithm finds better solutions than DICOPT. In 704, 641 and 581 out of 800 instances for 20, 50 and 80 operations, respectively, the EFFRONT-S_2^2 algorithm found a better solution than DICOPT. As the number of operations increases, this performance seems to slightly decrease. This is due to the usage of the step size limit. This limit is 0.000069 for 20 operations case whereas it is 0.012 for 80 operations case with $E = 0.0001$. Using a smaller step size limit increases the number

and the quality of the generated points. On the other hand, a smaller step size means greater CPU time requirements. Comparing the EFFRONT-S_2^2 algorithm with BARON, we see that these two methods perform similar to each other. The number of instances where each of these methods performed better than the other is nearly equal (387 vs 411). However, the average value of R_2 is greater (in absolute magnitude) for the instances where EFFRONT-S_2^2 performed better than the instances where BARON performed better (-49.361 vs 0.785 ($\times 10^{-6}$), respectively). Finally, comparing BARON with DICOPT, we can conclude that in 699 out of 800 cases, BARON performed better than DICOPT and in the remaining ones they found the same solution. DICOPT could not find a better solution than BARON. However, the average R_3 value is very small (2.982×10^{-6}).

Paired t-tests presented in Table 8.5 compare the three methods in pairs (EFFRONT-S_2^2, DICOPT), (EFFRONT-S_2^2, BARON) and (DICOPT, BARON) with respect to their F_1 values at each instance. The tests prove that the differences of F_1 values are statistically significant for the three methods except for $p = 80$. Considering the 95% confidence interval of the difference, although the lower and upper limits of the confidence interval are very small, EFFRONT-S_2^2 performs better than both DICOPT and BARON for $p = 20$ and better than DICOPT for $p = 50$. Also BARON performs better than DICOPT for $p = 20$. The reason for the probability being greater for $p = 80$ is the high standard deviation. That is, the performances of EFFRONT-S_2^2 and DICOPT have a high variability from instance to instance for $p = 80$.

Another factor for evaluating the quality of an algorithm is the CPU time requirements. The total number of points generated by the EFFRONT-S_2^2 algorithm and the corresponding CPU times for each factor combination for all methods are presented in Table 8.6. The CPU times listed for DICOPT and BARON are only for generating 20 points on the efficient frontier. The

p	Pair	Paired Differences					t	df	Sig.
		Mean	Std. Deviation	Std. Error Mean	95% Confidence Interval of the Difference				
					Lower	Upper			
20	E-D	-3.75E-03	3.72E-02	1.31E-03	-6.33E-03	-1.17E-03	-2,855	799	0,004
	E-B	-3.47E-03	3.72E-02	1.31E-03	-6.04E-03	-8.87E-04	-2,638	799	0,008
	D-B	2.86E-04	7.02E-04	2.48E-05	2.38E-04	3.35E-04	11,540	799	0,000
50	E-D	-5.38E-04	1.77E-03	6.26E-05	-6.61E-04	-4.15E-04	-8,586	799	0,000
80	E-D	-6.99E-03	0,126013	4.46E-03	-1.57E-02	1.75E-03	-1,570	799	0,117

Table 8.5: Paired t-tests

results indicate that the EFFRONT-S_2^2 algorithm generates at least 600 times more points than DICOPT and BARON generate in a very small CPU time. The increase in the CPU time as the number of operations increases is very small for the EFFRONT-S_2^2 algorithm (from 18.6 for 20 operations to 203.7 for 80 operations). On the other hand, for DICOPT even for 50 operations, the average CPU time requirements (892 seconds) reaches time limit of normal completion (900 seconds) and for BARON we are not able to generate solutions for $p = 50$ and $p = 80$ cases even with setting $optcr = 0.05$ after the first 1800 seconds.

Now let us evaluate the effects of the experimental design parameters on the performance of the methods. Table 8.7 aggregates the results for each design parameter. The results indicate that having a high $t^L - t^U$ range (Factor B set to 0.5) is in favor of the EFFRONT-S_2^2 algorithm. Considering the number of points generated by the EFFRONT-S_2^2 algorithm presented in Table 8.6, we can determine the reasoning. When the $t^L - t^U$ range is high, as expected, the EFFRONT-S_2^2 algorithm generates 2 to 3 times more points than the other case. This is because, the two solutions, one of which is generated from the other one, are consecutive and the difference between two consecutive points decreases when the generated number of points increases. As a result as the number of generated points increases, the solution quality also increases. But as

					EFFRONT-S_2^2		DICOPT	BARON
Factors				N	CPU	Number of	CPU	CPU
p	B	C	D		Time (sec)	Points	Time (sec)	Time (sec)
20	0	0	0	100	27,15	29207,6	52,26	1983,08
	0	0	1	100	42,71	44938,8	45,28	2006,21
	0	1	0	100	16,49	18416,2	50,66	1577,26
	0	1	1	100	20,84	21975,0	69,83	1636,31
	1	0	0	100	9,03	9716,8	129,15	1692,18
	1	0	1	100	20,08	19871,2	121,41	1692,31
	1	1	0	100	4,83	4745,6	118,50	1696,02
	1	1	1	100	7,69	7324,0	93,58	1684,24
Average					18,60	19524,4	85,08	1745,95
50	0	0	0	100	95,99	25351,4	838,32	
	0	0	1	100	188,87	50500,0	885,13	
	0	1	0	100	73,03	19884,4	852,56	
	0	1	1	100	135,81	36095,0	1060,18	
	1	0	0	100	38,11	9896,8	900,28	
	1	0	1	100	73,69	19727,2	846,28	
	1	1	0	100	28,00	7763,4	873,18	
	1	1	1	100	53,38	14440,4	882,10	
Average					85,86	22957,3	892,27	
80	0	0	0	100	112,03	14020,0	790,47	
	0	0	1	100	225,34	27860,4	786,07	
	0	1	0	100	86,48	10930,0	736,98	
	0	1	1	100	175,32	21791,0	1857,62	
	1	0	0	100	43,39	5594,6	882,28	
	1	0	1	100	88,71	11072,6	880,17	
	1	1	0	100	32,48	4240,6	882,43	
	1	1	1	100	66,21	8415,0	891,28	
Average					103,75	12990,5	963,41	

Table 8.6: Number of points generated by the EFFRONT-S_2^2 and the CPU times

already mentioned and as listed in the same table, the CPU time requirements increase in this case. However, DICOPT requires greater CPU time when the $t^L - t^U$ range is smaller. When the range is smaller, this means a lot of alternatives with similar penalty costs for the MINLP solvers to evaluate, where the marginal contribution is very small. As a result, proving the optimality of a solution requires more CPU time in such a case.

				R_1				
A	Factor	Level	N	$= 0$	> 0	Avg ($\times 10^{-6}$)	< 0	Avg ($\times 10^{-6}$)
20	B	0	400	3	43	0,567	354	-9,160
		1	400	3	47	0,315	350	-5,240
	C	0	400	2	41	0,627	357	-5,576
		1	400	4	49	0,348	347	-9,352
	D	0	400	1	60	0,475	339	-7,579
		1	400	5	30	0,465	365	-6,820
50	B	0	400	2	47	0,080	351	-3,855
		1	400	3	107	0,108	290	-0,715
	C	0	400	2	95	0,088	303	-2,553
		1	400	3	59	0,118	338	-2,320
	D	0	400	3	76	0,132	321	-2,210
		1	400	2	78	0,067	320	-2,650
80	B	0	400	1	158	0,612	331	-45,532
		1	400	4	300	0,166	250	-0,360
	C	0	400	3	101	0,064	296	-46,978
		1	400	2	113	0,081	285	-4,405
	D	0	400	1	253	0,189	282	-39,433
		1	400	4	205	0,108	299	-13,515

Table 8.7: Analysis of factors

Tables H.1 and H.2 placed in Appendix H present the ANOVA results for R_1, R_2 and R_3 statistics. These results prove that Factor B is significant with respect to R_1, R_2 and R_3 for $p = 20$ and $p = 50$ cases. Furthermore, t^U variability (Factor C) and t^U level (Factor D) are significant with respect to R_1 and R_2 for only $p = 20$ case. From Table 8.6 we observe that the number of points generated by the EFFRONT-S_2^2 algorithm and the required CPU

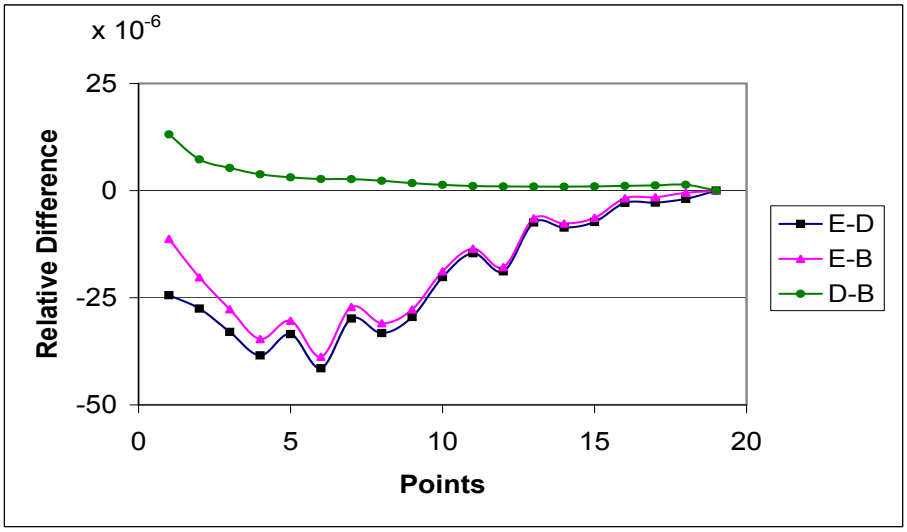

Figure 8.2: Relative differences for 20 points with $p = 20$

times are higher when the t^U level is higher and t^U variability is lower. This is reasonable since as the t^U variability reduces, the step size also reduces and more points are generated by the EFFRONT-S_2^2 algorithm. On the other hand, by considering F_1 values instead of R_1, R_2 and R_3 values in the ANOVA analysis, all the factors appear to be statistically significant.

It is also important to analyze the performance of the EFFRONT-S_2^2 algorithm on different parts of the efficient frontier. We generated a total of 20 points on the efficient frontier using DICOPT and BARON. Let these points be numbered from 1 to 20 where Point 1 corresponds to the minimum cycle time-maximum cost solution and Point 20 corresponds to the maximum cycle time-minimum cost solution. Figure 8.2 depicts the average R_1, R_2 and R_3 values with respect to points for the $p = 20$ case. We can conclude that the EFFRONT-S_2^2 algorithm performs better in the middle parts of the efficient frontier. As we go from Point 1 to Point 20, the performances of all methods get closer. This is due to the behavior of the manufacturing cost functions. The derivatives of the cost functions for all operations become almost equal at the tails of the cost functions. This means that the marginal contribution

of changing the processing times to the cost is almost equal for all operations. As a result, different allocations with different processing time settings yield very close cost values. However, this increases the CPU time requirements for the MINLP Solvers. Figures 8.3 and 8.4 depict the average CPU time requirements with respect to the 20 points for $p = 20$ operations. Figure 8.5 depicts a similar figure for $p = 50$ and $p = 80$ cases on the same chart. These figures support the above ideas. As we move from Point 1 to Point 20, the CPU requirements increase and in most of the cases the MINLP solvers stop by the time limit through Point 20. Note that, Point 20 is generated by solving the

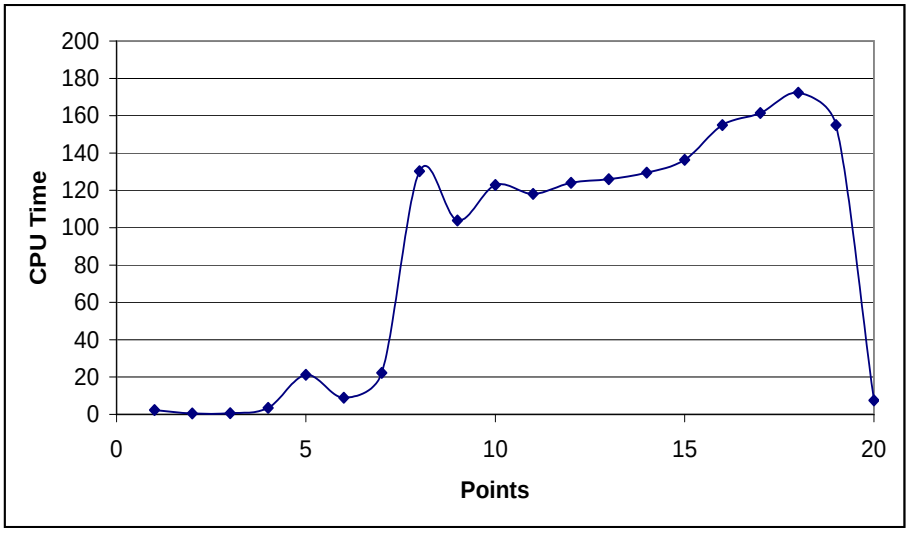

Figure 8.3: CPU times for $p = 20$ operations with DICOPT

AP formulation using CPLEX. As a consequence, the CPU requirements to generate this point is smaller with respect to the points generated by DICOPT and BARON.

Since we considered a bicriteria problem, after comparing the methods with respect to solution quality and the corresponding CPU time, we also have to compare the solution methods with respect to some multicriteria comparison criteria. In the literature there are different metrics used to compare the approximation quality of solution sets generated by different

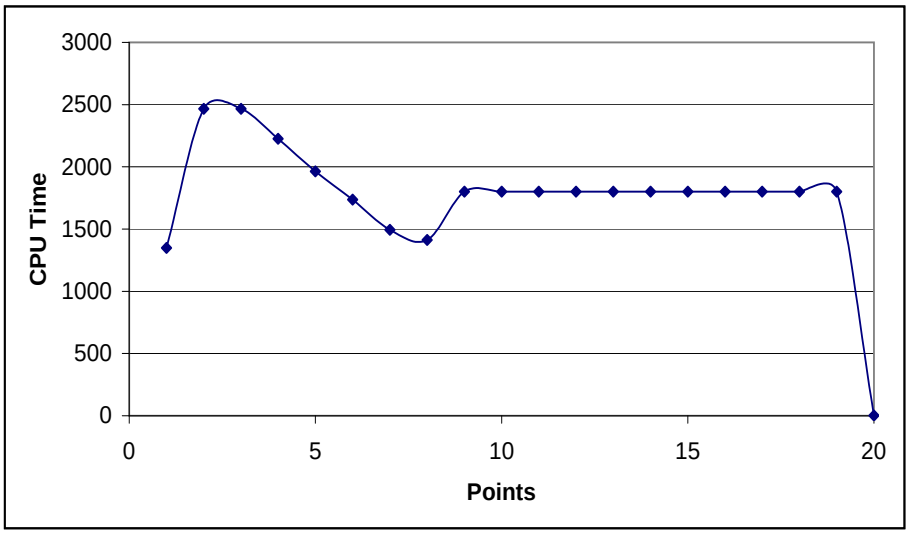

Figure 8.4: CPU times for $p = 20$ operations with BARON

methods. We will use the metric proposed by Hansen and Jaszkiewicz [48], which consists of measuring the probability $P(A, B)$ that an algorithm A gives a better solution than another algorithm B. It is calculated as $P(A, B) = \int_{u\in[0,1]} C(A(u), B(u))du$, where

$$C(A(u), B(u)) = \begin{cases} 1, & f(A(u)) < f(B(u)), \\ 1/2, & f(A(u)) = f(B(u)), \\ 0, & f(A(u)) > f(B(u)) \end{cases}$$

and $f(A(u)) = min_{x\in A}\{max(uF_1'(x), (1-u)F_2'(x))\}$, where $F_1'(x) = (F_1(x) - F_1(Z_2))/(F_1(Z_1) - F_1(Z_2))$ which is a normalization of F_1 using the minimum cycle time-maximum cost solution (Z_1) and maximum cycle time-minimum cost solution (Z_2). This method estimates the decision maker's probability to choose a solution generated by method A over method B. Table 8.8 presents the P(EFFRONT-S_2^2, DICOPT) results for each factor combination. The results indicate that the probability for the decision maker to select a solution generated by the EFFRONT-S_2^2 algorithm never falls below 99.3%. These results are due to the incomparably large number of solutions generated by

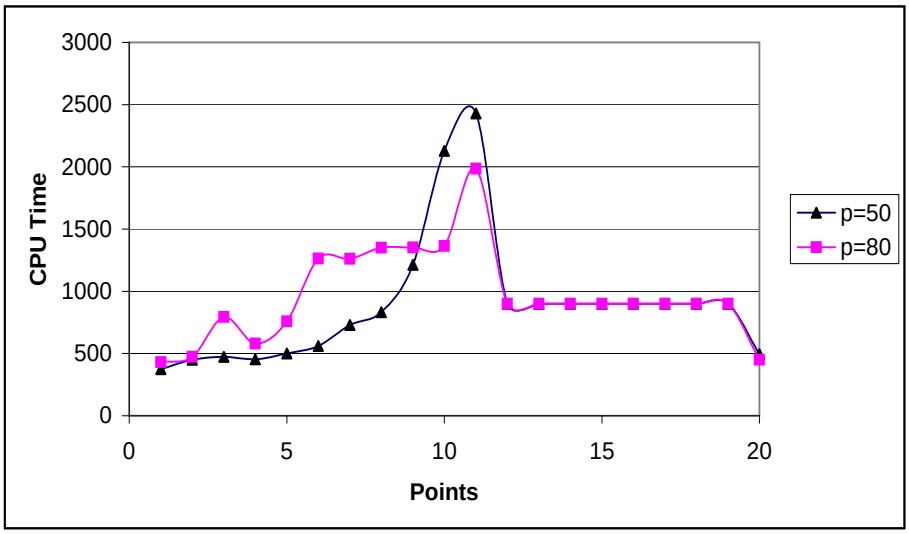

Figure 8.5: CPU times for $p = 50$ and $p = 80$ operations with DICOPT

the EFFRONT-S_2^2 algorithm with respect to DICOPT and the quality of the solutions. In conclusion, the EFFRONT-S_2^2 algorithm appears to be a very efficient method to generate nondominated solutions from the multicriteria optimization perspective as well.

In this section we presented the computational results of the EFFRONT-S_2^2 algorithm. The next section is devoted to the concluding remarks.

8.5 Conclusion

In this chapter, we considered bicriteria robotic cell scheduling problem in a 2-machine robotic cell. The machines are assumed to be CNC machines which are highly flexible. As a result, instead of assuming the processing times to be fixed on each machine, we assumed the allocation of the operations as well as the processing time values of each operation to be decision variables. We presented the mathematical formulation of the problem which appeared to be a Mixed Integer Nonlinear Programming (MINLP) formulation. Considering

p	A	B	C	$P(EFFRONT\text{-}S_2^2, DICOPT)$
20	0	0	0	1,00000
	0	0	1	1,00000
	0	1	0	0,99400
	0	1	1	1,00000
	1	0	0	1,00000
	1	0	1	1,00000
	1	1	0	0,99300
	1	1	1	0,99800
Average				0,99812
50	0	0	0	1,00000
	0	0	1	1,00000
	0	1	0	1,00000
	0	1	1	1,00000
	1	0	0	1,00000
	1	0	1	1,00000
	1	1	0	1,00000
	1	1	1	1,00000
Average				1,00000
80	0	0	0	1,00000
	0	0	1	1,00000
	0	1	0	1,00000
	0	1	1	0,99900
	1	0	0	1,00000
	1	0	1	1,00000
	1	1	0	1,00000
	1	1	1	1,00000
Average				0,99987

Table 8.8: Comparison of EFFRONT with DICOPT with a multi-objective criteria

each of the 1-unit cycles one at a time, we first developed a solution procedure for the S_1^2 cycle. Since the allocation problem for the S_2^2 cycle is proved to be NP-Complete, we presented a heuristic algorithm that generates a set of approximate efficient solutions. We compared the results of the algorithm with commercial MINLP solvers DICOPT and BARON. The computational study proved that the proposed algorithm is very efficient in terms of the number and the quality of the generated efficient solutions, the computational requirements and from the multicriteria point of view.

Chapter 9

Conclusion

This thesis considers scheduling problems arising in flexible robotic manufacturing cells. The machines used in these systems for metal cutting operations are predominantly CNC machines which possess many different types of flexibilities. Although flexibility is the key term that affects the performance of these systems, the current literature of robotic cell scheduling problems ignores this. As a consequence, the results of the earlier studies are either suboptimal or valid under some limiting assumptions. Furthermore, the earlier studies considered only the operational problems such as finding the part input sequence and the robot move sequence. However, the efficiency of the cells depends on the design of the cells as well as the operation of the cells. This is the first study that considers flexibility issues and some design problems in the robotic cell scheduling literature.

In the next section the contributions of this study will be explained and in Section 9.2, some future research directions will be provided.

9.1 Contributions

In this thesis, a manufacturing cell composed of a number of highly flexible CNC machines and a material handling robot which produces identical parts is considered. The CNC machines can perform different operations as long as the required cutting tools are loaded in their tool magazines. As a consequence, each part is assumed to have a set of operations to be performed on these machines. The current literature assumes the allocation of the operations to the machines to be fixed for each part. Hence, the processing time of each part on each machine is assumed to be a known parameter. However, this assumption is unrealistic for cells consisting of CNC machines and limits the number of alternatives unnecessarily. In the first part of the thesis (Chapters 4-6), the allocation of the operations to the machines is assumed to be a decision variable. Since different allocations yield different processing times which in turn yield a different cycle time value, considering the allocation of the operations as a decision variable is an option to increase the efficiency of such cells.

In Chapter 4, we define a new class of robot move cycles, namely, the class of *pure cycles*, which is a direct consequence of the flexibility of the machines. Despite the fact that these cycles are used extensively in industry because of their simplicity, there were no studies considering these cycles in the robotic cell scheduling literature until this study. For 2-machine cells we prove that the set of pure cycles dominates all the traditional robot move cycles considered in the literature. Furthermore, we determine the regions of optimality for each of the 6 feasible pure cycles in a 2-machine cell. Since the number of pure cycles increases drastically as the number of machines increases, we select the most widely used pure cycle for 3- and m-machine cells as the proposed cycle. The regions where this cycle dominates the traditional robot move cycles are determined. For the remaining region, a worst case performance bound is

derived for the proposed cycle. Since 3-machine cells are common in industry we analyze these cells in detail and prove that the proposed cycle dominates all 2-unit cycles and all 1-unit cycles except S_6^3. The results indicate that the pure cycles are not only simple and practical but also perform effectively.

The efficiency of the cells measured in terms of the throughput of the cells depends on the operations of the cells as well as the design of the cells. Despite this fact, to our knowledge, there are no studies considering the design of the cells. In Chapter 5, two design problems are considered. First, we consider the layout of the cells as a decision variable and prove that changing the layout from in-line robotic cell layout to robot centered cell layout reduces the cycle time for pure cycles. This is an important result since the robot centered cells are preferred to in-line robotic cells in industry because they require less physical space and because the rotational movements of a robot are more convenient than linear movements. The second design problem considered in this chapter is the determination of the optimal number of machines to be placed in a cell. Considering the proposed cycle, we determine the optimal number of machines that a robot serves for given parameters such as the robot move time and loading/unloading time. The results can be used to determine the machine, robot and other equipment requirements, to redesign the production facility, to determine the physical space requirements, etc., in order to increase the efficiency of the system.

The machines considered in this study are highly flexible CNC machines which can perform any operation as long as the required cutting tool is loaded on the tool magazine of the machine. However, the tool magazines of these machines have limited capacity. Additionally, duplicating and loading a copy of each of the required tools to the tool magazines of all of the machines may not be economically justifiable due to high tool investment costs. Such a situation is analyzed in Chapter 6. A 2-machine cell producing identical parts

is considered where each of the identical parts has three sets of operations. The operations that are in the first set can only be processed on the first machine, the operations that are in the second set can only be processed on the second machine, and the operations of the last set can be processed on either machine due to tooling constraints. Then the problem is not only sequencing the robot's activities but also partitioning the last set of operations into two machines. We prove that the optimal solution to this problem is not necessarily a 1-unit cycle but a 2-unit cycle can also be optimal in some regions. We reduce the number of potentially optimal robot move cycles to three and present the regions of optimality for each of these cycles. We show that the earlier problems considered in the literature are special cases of this one.

In the second part of the thesis, we consider a bicriteria approach to the robotic cell scheduling problem, an approach undertaken for the first time in this literature. Time and cost related objectives are two main objectives considered in the scheduling literature. Although minimizing production costs has the highest priority in process planning, there were no studies considering cost objectives in the robotic cell scheduling literature until this study. The processing times of the parts on the CNC machines can be controlled by adjusting the machining conditions such as the speed and the feed rate. However, adjusting these parameters also affects the tool life which consequently affects the total manufacturing cost. Hence, a bicriteria problem is considered in which the cycle time and the total manufacturing cost are the performance measures. Since there are two competing performance measures, instead of a unique optimal solution, a set of nondominated solutions exists for such problems.

In Chapter 7, 2- and 3-machine cells, where each part has one operation to be performed on each machine, are considered. The robot move sequence as well as the processing times on the machines are the decision variables.

The analysis is carried out for any non-linear, convex, nonincreasing cost function. We determine a set of nondominated solutions for the two 1-unit cycles of 2-machine robotic cells and compare these two cycles with each other to determine the regions of optimality for each. A similar analysis is also performed for 3-machine cells. We prove that two of the six 1-unit cycles of a 3-machine cell are dominated and need not be considered any further. We determine a nondominated set of solutions for the remaining four cycles. We prove that S_6^3 dominates the rest of the cycles for the cycle time values that can be attained by this cycle. For the remaining very small region, it is shown that no dominance relations exist between the remaining three cycles. These results suggest that considering only the cycle time as the unique objective function hinders the additional information provided by the cost objective and considering several different criteria provides useful insights to the decision maker. We also consider how different assumptions on cost structures can be handled using a 2-machine cell. These assumptions include how we can allocate the machining cost and how we can take the cost incurred by the robot into account. We show that if the machining cost and the cost of the robot are allocated with respect to the cycle time, earlier results found in this study are still valid. However, if the robot cost is allocated with respect to the exact working time of the robot, the regions of optimality for the S_1^2 and S_2^2 cycles change. This change is in favor of the S_1^2 cycle under which the number of robot moves is less than the number of robot moves under S_2^2 cycle.

In the last chapter, we consider a more general bicriteria model by including the decision of the allocation of the operations to the problem definition as well. We first consider the S_1^2 cycle for which the cycle time is independent of the allocation of the operations. We show that the problem is identical to a single machine makespan minimization problem with controllable processing times and propose a new solution procedure to solve the problem in polynomial time

by solving a nonlinear equation system with an approximation algorithm. On the other hand, the cycle time of the S_2^2 cycle depends on the allocation of the operations and the allocation problem is proved to be NP-Complete. The mathematical formulation of the problem is a mixed integer nonlinear (MINLP) program. We develop a heuristic algorithm which efficiently generates a large number of approximate efficient points. The results of the experimental design on the problem parameters prove that the proposed algorithm is a powerful method in terms of the number and the quality of the generated efficient solutions and the computational requirements as well as in terms of several multicriteria evaluation metrics.

9.2 Future Research Directions

This study considers a new set of problems for the first time in the robotic cell scheduling literature which initiates some new research directions. These problems arise from: (i) the assumption that the cell is a flexible manufacturing cell in which highly flexible machines are used, (ii) considering bicriteria approaches on robotic cell scheduling problems, and (iii) considering some design problems.

Throughout this study we assume the parts to be identical. However, since CNC machines are used to increase flexibility and reduce the response time to meet the customer demand, the assumption of having identical parts may be limiting in such systems. Considering multiple parts is a challenging open problem in which case the additional problem of finding the part input sequence must be solved. Another possible topic may be the case of non-identical machines. Such a situation is commonly faced in industry, where for example, some of the machines may be new and may technologically dominate

the other ones in terms of power, speed etc. As a consequence, the processing times of the same operation on different machines will be different. Hence, the allocation problem of the operations to the machines and determining the processing times for the controllable processing times get complicated even further. The assumptions of multiple parts and nonidentical machines can simultaneously or independently be adapted to all problems considered in this study leading to new challenging problems.

In Chapter 4, we define a new class of robot move cycles called the pure cycles and show that these cycles perform efficiently in comparison with the traditional robot move cycles. However, finding the best pure cycle in an m-machine cell is an open problem. Note that this is not an easy problem since there are $(2m-1)!$ pure cycles in an m-machine cell. Deriving the cycle times and comparing them with each other for that many cycles is a very challenging task.

In Chapter 5, we consider some design problems including the layout of the cells and the optimal number of machines to be placed in a cell. These design problems can be combined with operational problems to have a more general problem. For example, the optimal number of machines and the optimal layouts can be determined for each pure cycle and the best pure cycle can be determined by comparing the cycle times of these cycles. Additionally, there are some studies considering determination of the number of robots in a robotic cell with no-wait constraints. However, there are no studies which relax the no-wait constraints. Combining such problems with the ones suggested in this study will lead to more general design problems.

The number of machines in a robotic cell is an important parameter affecting the problem complexity. Most analytical results could be derived for the cells where the number of machines is 2 or 3. In this study also, in

Chapters 6 and 8 we consider 2-machine cells and in Chapter 7 we consider 2- and 3-machine cells together. Solving similar problems for cells with larger number of machines will be an important contribution to the literature.

A newly emerging area of research in the robotic cell scheduling literature considers dual gripper robotic cells. Such robots can unload and then load the same machine without having to move to another machine which leads to an increase in the cell efficiency. However, the number of feasible robot move cycles increases drastically in such cells. Considering dual gripper robotic cells with operation allocation as well as with bicriteria objectives developed in this study is a new research direction.

In Chapters 7 and 8, for the sake of simplicity, we restrict our analysis to the 1-unit cycles. 1-unit cycles are important because they are simple, practical and provide efficient results. However, as it is shown, in some cases these cycles may yield suboptimal results. Hence, another future research direction is to extend the analysis of these chapters so that not only 1-unit cycles but all feasible cycles are considered as alternatives.

Bibliography

[1] A. Agnetis. No-wait flow shop scheduling with large lot sizes. *Annals of Operations Research*, 70(3):415–438, 1997.

[2] M.S. Akturk, H. Gultekin, and O.E. Karasan. Robotic cell scheduling with operational flexibility. *Discrete Applied Mathematics*, 145(3):334–348, 2005.

[3] O. Alagoz and M. Azizoglu. Rescheduling of identical parallel machines under machine eligibility constraint. *European Journal of Operational Research*, 149:523–532, 2003.

[4] B. Alidaee and A. Ahmadian. Two parallel machine sequencing problems involving controllable processing times. *European Journal of Operational Research*, 70:335–341, 1993.

[5] Y.P. Aneja and H. Kamoun. Scheduling of parts and robot activities in two-machine robotic cell. *Computers & Operations Research*, 26:297–312, 1999.

[6] J.J. Bernardo and K.-S. Lin. An interactive procedure for bi-criteria production scheduling. *Computers & Operations Research*, 21(6):677–688, 1994.

[7] D.P. Bertsekas. *Nonlinear programming*. Athena Scientific, Belmont, Massachusetts, 1999.

[8] J. Blazewicz, G. Pawlak, and B. Walter. Scheduling production tasks in a two-stage fms. *International Journal of Production Research*, 40(17):4341–4352, 2002.

[9] N. Brauner and G. Finke. On the conjecture in robotic cells: new simplified proof for the three-machine case. *INFOR*, 37:20–36, 1999a.

[10] N. Brauner and G. Finke. On cycles and permutations in robotic cells. *Mathematical and Computer Modeling*, 34:565–591, 2001.

[11] N. Brauner, G. Finke, and W. Kubiak. Complexity of one-cycle robotic flow-shops. *Journal of Scheduling*, 6(4):355–371, 2003.

[12] K.M. Bretthauer and B. Shetty. The nonlinear knapsack problem–algorithms and applications. *European Journal of Operational Research*, 138:459–472, 2002.

[13] J. Browne, J. Harhen, and J. Shivnan. *Production Management Systems*. Addison-Wesley, New York, NY, 1996.

[14] D. Cao, M. Chen, and G. Wan. Parallel machine selection and job scheduling to minimize machine cost and job tardiness. *Computers & Operations Research*, 32:1995–2012, 2005.

[15] A. Che and C. Chuand E. Levner. A polynomial algorithm for 2-degree cyclic robot scheduling. *European Journal of Operational Research*, 145:31–44, 2003.

[16] T.C.E. Cheng, Z.-L. Chen, and C.-L. Li. Parallel machine scheduling with controllable processing times. *IIE Transactions*, 28:177–180, 1996.

[17] T.C.E. Cheng, A. Janiak, and M.Y. Kovalyov. Bicriterion single machine scheduling with resource dependent processing times. *SIAM Journal on Optimization*, 8:617–630, 1998.

[18] Y. Crama. Combinatorial optimization models for production scheduling in automated manufacturing systems. *European Journal of Operational Research*, 99:136–153, 1997.

[19] Y. Crama and J. Van de Klundert. Cyclic scheduling of identical parts in a robotic cell. *Operations Research*, 45(6):952–965, 1997.

[20] Y. Crama and J. Van de Klundert. Cyclic scheduling in 3-machine robotic flow shops. *Journal of Scheduling*, 4:35–54, 1999.

[21] Y. Crama, V. Kats, J. Van de Klundert, and E. Levner. Cyclic scheduling in robotic flowshops. *Annals of Operations Research*, 96:97–124, 2000.

[22] R.L. Daniels and R.K. Sarin. Single machine scheduling with controllable processing times and number of jobs tardy. *Operations Research*, 37:981–984, 1989.

[23] M. Dawande, N. Geismar, and S.P. Sethi. Dominance of cyclic solutions and challenges in the scheduling of robotic cells. *SIAM Review*, 47(4):709–721, 2005.

[24] M. Dawande, C. Sriskandarajah, and S.P. Sethi. On throughput maximization in constant travel-time robotic cells. *Manufacturing and Service Operations Management*, 4(4):296–312, 2002.

[25] I.G. Drobouchevitch, S. Sethi, J. Sidney, and C. Sriskandarajah. Scheduling multiple parts in two-machine dual gripper robotic cell: Heuristic algorithm and performance guarantee. *International Journal of Operations and Quantitative Management*, 10(4):297–314, 2004.

[26] I.G. Drobouchevitch, S. Sethi, J. Sidney, and C. Sriskandarajah. Scheduling dual gripper robotic cell: One-unit cycles. *European Journal of Operational Research*, 171:598–631, 2006.

[27] G. Finke, C. Gueguen, and N. Brauner. *ECOCO IX Conference Proceedings*, chapter Robotic cells with buffer space. Dublin City University, 1996.

[28] M.G. Garey and D.S. Johnson. *Computers and Intractability : A Guide to the Theory of NP-Completeness*. W.H. Freeman, 1976.

[29] H.N. Geismar, M. Dawande, S.P. Sethi, and C. Sriskandarajah. A note on productivity gains in flexible robotic cells. *International Journal of Flexible Manufacturing Systems*, 17(1):5–21, 2005.

[30] H.N. Geismar, M. Dawande, S.P. Sethi, and C. Sriskandarajah. Sequencing and scheduling in robotics cells: Recent developments. *Journal of Scheduling*, 8:387–426, 2005.

[31] H.N. Geismar, M. Dawande, and C. Sriskandarajah. Robotic cells with parallel machines: Throughput maximization in constant travel-time cells. *Journal of Scheduling*, 7(5):375–395, 2004.

[32] H.N. Geismar, M. Dawande, and C. Sriskandarajah. Throughput optimization in constant travel-time dual gripper robotic cells with parallel machines. *Production and Operations Management*, 2006. to appear.

[33] H.N. Geismar, C. Sriskandarajah, and N.N. Ramanan. Increasing throughput for robotic cells with parallel machines and multiple robots. *IEEE Transactions on Automation Science & Engineering*, 1(1):84–89, 2004.

[34] P.C. Gilmore and R.E. Gomory. Sequencing in one state-variable machine: a solvable case of the travelling salesman problem. *Operations Research*, 12:655–679, 1964.

[35] P.C. Gilmore, E.L. Lawler, and D.B. Shmoys. *Well solved special cases in: The Travelling Salesman Problem: A Guided Tour of Combinatorial Optimization*. Wiley-Interscience Series in Discrete Mathematics. Wiley, Chichester, 1985.

[36] R.L. Graham, E.L. Lawler, J.K. Lenstra, and A.H.G. Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals Discrete Mathematics*, 5:287–326, 1979.

[37] A.E. Gray, A. Seidmann, and K.E. Stecke. A synthesis of decision models for tool management in automated manufacturing. *Management Science*, 39:549–567, 1993.

[38] H. Gultekin, M.S. Akturk, and O.E. Karasan. Robotic cell scheduling with tooling constraints. *European Journal of Operational Research*, 174:777–796, 2006.

[39] H. Gultekin, M.S. Akturk, and O.E. Karasan. Scheduling in a three-machine robotic flexible manufacturing cell. *Computers & Operations Research*, 2006. In press.

[40] H. Gultekin, M.S. Akturk, and O.E. Karasan. Scheduling in robotic cells: process flexibility and cell layout. *International Journal of Production Research*, 2006. Accepted for publication.

[41] J.N.D. Gupta and J.C. Ho. Minimizing makespan subject to minimum flow time on two identical parallel machines. *Computers & Operations Research*, 28:705–717, 2001.

[42] J.N.D. Gupta and A.J. Ruiz-Torres. Generating efficient schedules for identical parallel machines involving flow-time and tardy jobs. *European Journal of Operational Research*, 167:679–695, 2005.

[43] N.G. Hall, H. Kamoun, and C. Sriskandarajah. Scheduling in robotic cells: classification, two and three machine cells. *Operations Research*, 45(3):421–439, 1997.

[44] N.G. Hall, H. Kamoun, and C. Sriskandarajah. Scheduling in robotic cells: complexity and steady state analysis. *European Journal of Operational Research*, 109:43–65, 1998.

[45] N.G. Hall, C.N. Potts, and C. Sriskandarajah. Parallel machine scheduling with a common server. *Discrete Applied Mathematics*, 102:223–243, 2000.

[46] N.G. Hall and C. Sriskandarajah. A survey of machine scheduling problems with blocking and no-wait in process. *Operations Research*, 44(3):510–525, 1196.

[47] B.T. Han and J.S. Cook. An efficient heuristic for robot acquisition and cell formation. *Annals of Operations Research*, 77:229–252, 1998.

[48] M.P. Hansen and A. Jaszkiewicz. Evaluating the quality of approximations to the non-dominated set. Technical Report IMM-REP-1998-7, Technical University of Denmark, 1998.

[49] K. Hitomi and M. Yoshimura. *Robotics and Material Flow*, chapter Operations Scheduling for Work Transportation by Industrial Robots in Automated Manufacturing Systems, pages 131–139. Elsevier Science Publishers, New York, NY, 1986.

[50] H. Hoogeveen. Multicriteria scheduling. *European Journal of Operational Research*, 167:592–623, 2005.

[51] J. Hurink and S. Knust. Makespan minimization for flow-shop problems with transportation times and a single robot. *Discrete Applied Mathematics*, 112:199–216, 2001.

[52] J. Hurink and S. Knust. A tabu search algorithm for scheduling a single robot in a job-shop environment. *Discrete Applied Mathematics*, 119:181–203, 2002.

[53] H. Ishii, C. Martel, T. Masuda, and T. Nishida. A generalized uniform processor system. *Operations Research*, 33:346–362, 1985.

[54] A. Janiak and M.Y. Kovalyov. Single machine scheduling subject to deadlines and resource dependent processing times. *European Journal of Operational Research*, 94:284–291, 1996.

[55] W. Jeng, J. Lin, and U. Wen. Algorithms for sequencing robot activities in a robot-centered parllel-processor workcell. *Computers & Operations Research*, 20:185–197, 1993.

[56] H. Kamoun, N.G. Hall, and C. Sriskandarajah. Scheduling in robotic cells: heuristics and cell design. *Operations Research*, 47(6):821–835, 1999.

[57] S. Karabati and P. Kouvelis. Flow-line scheduling problems with controllable processing times. *IIE Transactions*, 29:1–14, 1997.

[58] N. Karmarkar and R.M. Karp. The differencing method of set partitioning. Technical Report UCB/CSD 82/113, Computer Science Division, University of California, Berkeley, CA., 1982.

[59] A.V. Karzanov and E.M. Livshits. Minimal quality of operators for serving a homogeneous linear technological process, part 2. *Automation and Remote Control*, 39(3):445–450, 1978.

[60] V. Kats and E. Levner. Minimizing the number of robots to meet a given cyclic schedule. *Annals of Operations Research*, 69:209–226, 1997.

[61] V. Kats and E. Levner. A strongly polynomial algorithm for no-wait cyclic robotic flowshop scheduling. *Operations Research Letters*, 21:171–179, 1997.

[62] V. Kats and E. Levner. Cyclic scheduling of operations for a part type in an fms handled by a single robot: a parametric critical path approach. *International Journal of Flexible Manufacturing Systems*, 10:129–138, 1998.

[63] R.K. Kayan and M.S. Akturk. A new bounding mechanism for the cnc machine scheduling problems with controllable processing times. *European Journal of Operational Research*, 167:624–643, 2005.

[64] R.E. King, T.J. Hodgson, and F.E. Chafee. Robot task scheduling in a flexible manufacturing cell. *IIE Transactions*, 25(2):80–87, 1993.

[65] H. Kise, T. Shioyama, and T. Ibaraki. Automated two machine flowshop scheduling: a solvable case. *IIE Transactions*, 23:80–87, 1993.

[66] K. Kogan and E. Levner. A polynomial algorithm for scheduling small-scale manufacturing cells served by multiple robots. *Computers & Operations Research*, 25(1):53–62, 1997.

[67] M.M. Koksalan and A.B. Keha. Using genetic algorithms for single machine bicriteria scheduling problems. *European Journal of Operational Research*, 145(3):543–556, 2003.

[68] E. Koktener-Karasakal and M.M. Koksalan. A simulated annealing approach to bicriteria scheduling problems on a single machine. *Journal of Heuristics*, 6:311–327, 2000.

[69] C.Y. Lee, L. Lei, and M. Pinedo. Current trends in deterministic scheduling. *Annals of Operations Research*, 70:1–41, 1997.

[70] E. Levner, V. Kats, and V.E. Levit. An improved algorithm for cyclic flowshop scheduling in a robotic cell. *European Journal of Operational Research*, 97:500–508, 1997.

[71] E. Levner, K. Kogan, and Y. Levin. Scheduling a two-machine robotic cell: a solvable case. *Annals of Operations Research*, 57:217–232, 1995.

[72] E. Levner and M. Vlach. Single machine scheduling with fuzzy precedence constraints. Technical Report IS-RR-97-0031F, School of Information Science, Japan Institute of Science and Technology, Hokuriku, 1997.

[73] R.W. Lieberman and I.B. Turksen. Crane scheduling problems. *AIIE Transactions*, 13(4):304–311, 1981.

[74] S.D. Liman, S.S. Panwalkar, and S. Thongmee. A single machine scheduling problem with common due window and cotrollable processing times. *Annals of Operations Research*, 70:145–154, 1997.

[75] R. Logendran and C. Sriskandarajah. Sequencing of robot activities and parts in two-machine robotic cells. *International Journal of Production Research*, 34:3447–3463, 1996.

[76] S. Mohri, T. Masuda, and H. Ishii. Bi-criteria scheduling problem on three identical parallel machines. *International Journal of Production Research*, 60–61:529–536, 1999.

[77] S.K. Mukhopadhyay and S.K. Sahu. Priority-based tool allocation in a flexible manufacturing system. *International Journal of Production Research*, 34(7):1995–2018, 1996.

[78] A. Nagar, J. Haddock, and S. Heragu. Multiple criteria and bicriteria scheduling: A literature survey. *European Journal of Operational Research*, 81:88–104, 1995.

[79] E. Nowicki and S. Zdrzalka. A survey of results for sequencing problems with controllable processing times. *Discrete Applied Mathematics*, 26:271–287, 1990.

[80] E. Nowicki and S. Zdrzalka. A bicriterion approach to preemptive scheduling of parallel machines with controllable processing times. *Discrete Applied Mathematics*, 63:237–256, 1995.

[81] S.S. Panwalkar and R. Rajagopalan. Single machine sequencing with controllable processing times. *European Journal of Operational Research*, 59:298–301, 1992.

[82] C.H. Papdimitriou and P.C. Kanellakis. Flowshop scheduling with limited temporary storage. *Journal of the ACM*, 27(3):533–549, 1980.

[83] M. Pinedo. *Scheduling: Theory, Algorithms, and Systems*. Prentice Hall, Englewood Cliffs, New Jersey, 1995.

[84] A.J. Ruiz-Torres, E.E. Enscore, and R.R. Barton. Simulated annealing heuristics for the average flow time and the number of tardy jobs bicriteria identical parallel machine problem. *Computers & Industrial Engineering*, 33(1–2):257–260, 1997.

[85] S. Sethi, C. Sriskandarajah, and J. Sidney. Scheduling in dual gripper robotic cells for productivity gains. *IEEE Transactions on Robotics and Automation*, 17:324–341, 2001.

[86] S.P. Sethi, C. Sriskandarajah, G. Sorger, J. Blazewicz, and W. Kubiak. Sequencing of parts and robot moves in a robotic cell. *International Journal of Flexible Manufacturing Systems*, 4:331–358, 1992.

[87] D. Shabtay and M. Kaspi. Minimizing the total weighted flow time in a single machine with controllable processing times. *Computers & Operations Research*, 31:2279–2289, 2004.

[88] W. Song, Z.B. Zabinksky, and R.L. Storch. An algorithm for scheduling a chemical processing tank line. *Production Planning & Control*, 4(4):323–332, 1993.

[89] C. Sriskandarajah, I.G. Drobouchevitch, S. Sethi, and R. Chandrasekaran. Scheduling multiple parts in a robotic cell served by a dual gripper robot. *Operations Research*, 52(1):65–82, 2004.

[90] C. Sriskandarajah, N.G. Hall, and H. Kamoun. Scheduling large robotic cells without buffers. *Annals of Operations Research*, 76:287–321, 1998.

[91] H.I. Stern and G. Vitner. Scheduling parts in a combined production-transportation work cell. *Journal of the Operational Reseach Society*, 41:625–632, 1990.

[92] V. Suresh and D. Chaudhuri. Bicriteria scheduling problem for unrelated parallel machines. *Computers & Industrial Engineering*, 30(1):77–82, 1996.

[93] M.K. Tiwari and N.K. Vidyarthi. Solving machine loading problems in a flexible manufacturing system using a genetic algorithm based heuristic approach. *International Journal of Production Research*, 38(14):3357–3384, 2000.

[94] M.A. Trick. Scheduling multiple variable speed machines. *Operations Research*, 42:234–248, 1994.

[95] R.G. Vickson. Choosing the job sequence and processing times to minimize total processing plus flow cost on a single machine. *Operations Research*, 28:1155–1167, 1980.

[96] R.G. Vickson. Two single machine sequencing problems involving controllable job processing times. *AIIE Transactions*, 12(3):258–262, 1980.

[97] L.N. Van Wassenhove and K.R. Baker. A bicriterion approach to time/cost trade-offs in sequencing. *European Journal of Operational Research*, 11:48–54, 1982.

[98] J. Wu and S. Azarm. Metrics for quality assessment of a multi-objective design optimization solution set. *Journal of Mechanical Design*, 123(1):18–25, 2001.

[99] F. Zhang, G. Tang, and Z.-L. Chen. A 3/2 approximation algorithm for parallel machine scheduling with controllable processing times. *Operations Research Letters*, 29:41–47, 2001.

[100] E. Ztizler, L. Thiele, M. Laumans, C. Fonseca, and V.G. Fonseca. Performance assessment of multi-objective optimizers: An analysis and review. *IEEE Transactions on Evolutionary Computation*, 7(2):117–132, 2003.

Appendix A

Pure cycles for 2-machine cells

Cycle	Activity Sequence	Cycle Time
C1	$A_{01}A_{02}A_{13}A_{23}$	$4\epsilon + 6\delta + 1/2(max\{0, P - 2\epsilon - 4\delta\})$
C2	$A_{01}A_{02}A_{23}A_{13}$	$4\epsilon + 6\delta + P/2$
C3	$A_{01}A_{13}A_{02}A_{23}$	$4\epsilon + 6\delta + P$
C4	$A_{01}A_{13}A_{23}A_{02}$	$4\epsilon + 6\delta + P/2$
C5	$A_{01}A_{23}A_{13}A_{02}$	$4\epsilon + 7\delta + 1/2(max\{0, P - 2\epsilon - 4\delta\})$
C6	$A_{01}A_{23}A_{02}A_{13}$	$4\epsilon + 7\delta + 1/2(max\{0, P - 4\epsilon - 8\delta\})$

Appendix B

Derivation of the 2-unit cycles

Let us present the procedure for deriving the activity sequences for the 2-unit robot move cycles. In a 2-unit cycle, each activity is made exactly twice. A 2-unit robot move cycle is in fact a combination of two 1-unit cycles. At some part of the 2-unit cycle it follows the activity sequence of one of the 1-unit cycles and in the remaining part it follows the activity sequence of the other 1-unit cycle. Then, in order to follow the activity sequences of two 1-unit robot move cycles, there must be a transition state from one of them to the other and later another transition from the latter one to the initial. This requires the two 1-unit robot move cycles to have at least one common state. In the context of this study, any state of the system can be defined as a triplet $(x_1x_2x_3)$, $x_i \in \{0, 1\}$ where $x_i = 0$ indicates that machine i is empty and $x_i = 1$ indicates that machine i is loaded. For example, (011) is a state in which machine 1 is empty and machines 2 and 3 are loaded. The following lists the activity sequences and the states of all 1-unit robot move cycles:

$S_1^3 : A_0A_1A_2A_3 : (000) \rightarrow (100) \rightarrow (010) \rightarrow (001)$,

$S_2^3 : A_0A_2A_1A_3 : (010) \rightarrow (110) \rightarrow (101) \rightarrow (011)$,

$S_3^3 : A_0A_1A_3A_2 : (001) \rightarrow (101) \rightarrow (011) \rightarrow (010)$,

$S_4^3 : A_0A_3A_1A_2 : (001) \rightarrow (101) \rightarrow (100) \rightarrow (010)$,
$S_5^3 : A_0A_2A_3A_1 : (010) \rightarrow (110) \rightarrow (101) \rightarrow (100)$,
$S_6^3 : A_0A_3A_2A_1 : (011) \rightarrow (111) \rightarrow (110) \rightarrow (101)$.

In the above list all cycles have a common state with each other except cycles S_1^3 and S_6^3. This means that we will have $C(6,2) - 1 = 14$, 2-unit robot move cycles. Let us give an example of constructing a 2-unit robot move cycle from two 1-unit robot move cycles. Let S_{ij}^3 be defined as the 2-unit cycle made up from 1-unit cycles Si^3 and Sj^3. Let us consider S_1^3 and S_2^3 for this example. The common state for these two cycles is (010). Thus, without loss of generality, we may start with the activities of S_1^3 and follow them until we reach state (010); $A_0A_1 \cdots$. At the common state we start following the activity sequence of S_2^3 until we reach to that common state again; $A_0A_1A_0A_2A_1A_3 \cdots$. Finally, we end up with the remaining activities of S_1^3; $A_0A_1A_0A_2A_1A_3A_2A_3$.

The robot activity sequences for each of the fourteen 2-unit robot move cycles over 3-machines are listed below.

$S_{12}^3 = A_0A_1A_0A_2A_1A_3A_2A_3$	$S_{26}^3 = A_0A_2A_1A_0A_3A_2A_1A_3$
$S_{13}^3 = A_0A_1A_2A_0A_1A_3A_2A_3$	$S_{34}^3 = A_0A_1A_3A_2A_0A_3A_1A_2$
$S_{14}^3 = A_0A_1A_2A_0A_3A_1A_2A_3$	$S_{35}^3 = A_0A_1A_3A_0A_2A_3A_1A_2$
$S_{15}^3 = A_0A_1A_0A_2A_3A_1A_2A_3$	$S_{36}^3 = A_0A_1A_0A_3A_2A_1A_3A_2$
$S_{23}^3 = A_0A_1A_3A_0A_2A_1A_3A_2$	$S_{45}^3 = A_0A_2A_3A_1A_2A_0A_3A_1$
$S_{24}^3 = A_0A_2A_1A_3A_2A_0A_3A_1$	$S_{46}^3 = A_0A_1A_0A_3A_2A_3A_1A_2$
$S_{25}^3 = A_0A_2A_1A_3A_0A_2A_3A_1$	$S_{56}^3 = A_0A_2A_1A_0A_3A_2A_3A_1$

Appendix C

Lower bounds for the 2-unit cycles

Let $\underline{T_{S^3_{ij(\Pi^*_k)}}}$ be the lower bound of the cycle time of the 2-unit cycle S^3_{ij} with any allocation matrix Π_k. Note that $\underline{T_{S^3_{ij(\Pi^*_k)}}} \leq min_{\Pi_k}\{T_{S^3_{ij(\Pi_k)}}\}$. We will show that $\underline{T_2} \leq \underline{T_{S^3_{ij(\Pi^*_k)}}}$. We will consider each 2-unit cycle, S^3_{ij}, one at a time and derive a lower bound, $\underline{T_{S^3_{ij(\Pi^*_k)}}}$, for each of them. For each of these cycles, we will consider one repetition of the cycle where we assume w.l.o.g. that the cycle starts with the state that the robot is in front of the input buffer just taking a part with i^{th} allocation type. We will find a lower bound for the total time of this particular repetition and show that this lower bound does not depend on i, which means that the lower bound for the total time of this repetition of the cycle is also a lower bound for the total time of all repetitions of the cycle. Now let us consider each 2-unit cycle one at a time.

$\mathbf{S^3_{12}}$: One can calculate the total time for one repetition of this cycle starting with loading a part with i^{th} allocation type as $16\epsilon + 20\delta + P_{i1} + w_{i2} + w_{(i+1)1} + w_{i3} + w_{(i+1)2} + P_{(i+1)3}$,

where $w_{i2} = (P_{i2} - 2\epsilon - 4\delta)^+$, $w_{(i+1)1} = (P_{(i+1)1} - 2\epsilon - 4\delta - w_{i2})^+$, $w_{i3} = (P_{i3} - 2\epsilon - 4\delta - w_{(i+1)1})^+$, and $w_{(i+1)2} = (P_{(i+1)2} - 2\epsilon - 4\delta - w_{i3})^+$. Since we have $P_{i2} - 2\epsilon - 4\delta \leq w_{i2}$, $P_{i3} - 2\epsilon - 4\delta - w_{(i+1)1} \leq w_{i3}$, $0 \leq w_{(i+1)2}$ and $0 \leq P_{(i+1)3}$, we have $16\epsilon + 20\delta + P_{i1} + P_{i2} - 2\epsilon - 4\delta + w_{(i+1)1} + P_{i3} - 2\epsilon - 4\delta - w_{(i+1)1} \leq 16\epsilon + 20\delta + P_{i1} + w_{i2} + w_{(i+1)1} + w_{i3} + w_{(i+1)2} + P_{(i+1)3}$. Since in a 2-unit cycle two parts are produced in one repetition, a lower bound to produce one part can be found as:

$\underline{T_{S^3_{12(\Pi^*_k)}}} = 1/2(P + 12\epsilon + 12\delta) > \underline{T_2}$.

S^3_{13} : Total time of one repetition of this cycle is $16\epsilon + 18\delta + P_{i1} + P_{i2} + P_{(i+1)1} + w_{i3} + w_{(i+1)2} + P_{(i+1)3}$, where $w_{i3} = (P_{i3} - 4\epsilon - 6\delta - P_{(i+1)1})^+$ and $w_{(i+1)2} = (P_{(i+1)2} - 2\epsilon - 4\delta - w_{i3})^+$.
Since $P_{(i+1)2} - 2\epsilon - 4\delta - w_{i3} \leq w_{(i+1)2}$, we have $16\epsilon + 18\delta + P_{(i+1)1} + w_{i3} + P_{(i+1)2} - 2\epsilon - 4\delta - w_{i3} + P_{(i+1)3} \leq 16\epsilon + 18\delta + P_{i1} + P_{i2} + P_{(i+1)1} + w_{i3} + w_{(i+1)2} + P_{(i+1)3}$. Thus, a lower bound for the time to produce one part can be found as:

$\underline{T_{S^3_{13(\Pi^*_k)}}} = 1/2(P + 14\epsilon + 14\delta) > \underline{T_2}$.

S^3_{14} : Total time of one repetition of this cycle is $16\epsilon + 20\delta + P_{i1} + P_{i2} + w_{(i+1)1} + w_{i3} + P_{(i+1)2} + P_{(i+1)3}$, where $w_{i3} = (P_{i3} - 2\epsilon - 6\delta)^+$ and $w_{(i+1)1} = (P_{(i+1)1} - 2\epsilon - 6\delta - w_{i3})^+$.
Since $P_{(i+1)1} - 2\epsilon - 6\delta - w_{i3} \leq w_{(i+1)1}$, we have $16\epsilon + 20\delta + P_{(i+1)1} - 2\epsilon - 6\delta - w_{i3} + w_{i3} + P_{(i+1)2} + P_{(i+1)3} \leq 16\epsilon + 20\delta + P_{i1} + P_{i2} + w_{(i+1)1} + w_{i3} + P_{(i+1)2} + P_{(i+1)3}$. Thus, a lower bound for the time to produce one part can be found as:

$\underline{T_{S^3_{14(\Pi^*_k)}}} = 1/2(P + 14\epsilon + 14\delta) > \underline{T_2}$.

S^3_{15} : Total time of one repetition of this cycle is $16\epsilon + 18\delta + P_{i1} + w_{i2} + w_{(i+1)1} + P_{i3} + P_{(i+1)2} + P_{(i+1)3}$, where $w_{i2} = (P_{i2} - 2\epsilon - 4\delta)^+$ and

$w_{(i+1)1} = (P_{(i+1)1} - 4\epsilon - 6\delta - w_{i2} - P_{i3})^+$.
Since $P_{i2} - 2\epsilon - 4\delta \leq w_{i2}$ and $0 \leq w_{(i+1)1}$, a lower bound for the total time to produce one part can be found as:
$\underline{T_{S^3_{15(\Pi^*_k)}}} = 1/2(P + 14\epsilon + 14\delta) > \underline{T_2}$.

$\mathbf{S^3_{23}}$: Total time of one repetition of this cycle is $16\epsilon + 22\delta + P_{i1} + w_{(i-1)3} + w_{i2} + w_{(i+1)1} + w_{i3} + w_{(i+1)2}$, where $w_{(i-1)3} = (P_{(i-1)3} - 4\epsilon - 6\delta - P_{i1})^+$,
$w_{i2} = (P_{i2} - 4\epsilon - 8\delta - w_{(i-1)3})^+$, $w_{(i+1)1} = (P_{(i+1)1} - 2\epsilon - 4\delta - w_{i2})^+$,
$w_{i3} = (P_{i3} - 2\epsilon - 4\delta - w_{(i+1)1})^+$, and $w_{(i+1)2} = (P_{(i+1)2} - 2\epsilon - 6\delta - w_{i3})^+$.
Since $P_{i2} - 4\epsilon - 8\delta - w_{(i-1)3} \leq w_{i2}$, $P_{i3} - 2\epsilon - 4\delta - w_{(i+1)1} \leq w_{i3}$ and $0 \leq w_{(i+1)2}$, a lower bound for the total time to produce one part can be found as:
$\underline{T_{S^3_{23(\Pi^*_k)}}} = 1/2(P + 10\epsilon + 10\delta) > \underline{T_2}$.

$\mathbf{S^3_{24}}$: Total time of one repetition of this cycle is $16\epsilon + 24\delta + w_{(i-1)2} + w_{i1} + w_{(i-1)3} + w_{i2} + w_{i3} + w_{(i+1)1}$, where $w_{(i-1)2} = (P_{(i-1)2} - 2\epsilon - 4\delta)^+$,
$w_{i1} = (P_{i1} - 2\epsilon - 4\delta - w_{(i-1)2})^+$,
$w_{(i-1)3} = (P_{(i-1)3} - 2\epsilon - 4\delta - w_{i1})^+$, $w_{i2} = (P_{i2} - 2\epsilon - 4\delta - w_{(i-1)3})^+$,
$w_{i3} = (P_{i3} - 2\epsilon - 6\delta)^+$, and $w_{(i+1)1} = (P_{(i+1)1} - 2\epsilon - 6\delta - w_{i3})^+$.
Since $P_{i1} - 2\epsilon - 4\delta - w_{(i-1)2} \leq w_{i1}$, $P_{i2} - 2\epsilon - 4\delta - w_{(i-1)3} \leq w_{i2}$, $P_{i3} - 2\epsilon - 6\delta \leq w_{i3}$ and $0 \leq w_{(i+1)1}$, a lower bound for the total time to produce one part can be found as:
$\underline{T_{S^3_{24(\Pi^*_k)}}} = 1/2(P + 10\epsilon + 10\delta) > \underline{T_2}$.

$\mathbf{S^3_{25}}$: Total time of one repetition of this cycle is $16\epsilon + 22\delta + w_{(i-1)2} + w_{i1} + w_{(i-1)3} + w_{i2} + P_{i3} + w_{(i+1)1}$, where $w_{(i-1)2} = (P_{(i-1)2} - 2\epsilon - 4\delta)^+$,
$w_{i1} = (P_{i1} - 2\epsilon - 4\delta - w_{(i-1)2})^+$,
$w_{(i-1)3} = (P_{(i-1)3} - 2\epsilon - 4\delta - w_{i1})^+$, $w_{i2} = (P_{i2} - 4\epsilon - 8\delta - w_{(i-1)3})^+$, and

$w_{(i+1)1} = (P_{(i+1)1} - 4\epsilon - 6\delta - w_{i2} - P_{i3})^+$.
Since, $P_{i1} - 2\epsilon - 4\delta - w_{(i-1)2} \leq w_{i1}$, $P_{i2} - 4\epsilon - 8\delta - w_{(i-1)3} \leq w_{i2}$ and $0 \leq w_{(i+1)1}$, a lower bound for the total time to produce one part can be found as:
$\underline{T_{S^3_{25(\Pi^*_k)}} = 1/2(P + 10\epsilon + 10\delta) > \underline{T_2}}$.

$\mathbf{S^3_{26}}$: Total time of one repetition of this cycle is $16\epsilon + 24\delta + w_{(i-1)2} + w_{i1} + w_{(i-1)3} + w_{i2} + w_{i3} + w_{(i+1)1}$, where $w_{(i-1)2} = (P_{(i-1)2} - 4\epsilon - 8\delta - w_{i3})^+$,
$w_{i1} = (P_{i1} - 2\epsilon - 4\delta - w_{(i-1)2})^+$,
$w_{(i-1)3} = (P_{(i-1)3} - 4\epsilon - 8\delta - w_{i1})^+$, $w_{i2} = (P_{i2} - 4\epsilon - 8\delta - w_{(i-1)3})^+$,
$w_{(i+1)1} = (P_{(i+1)1} - 4\epsilon - 8\delta - w_{(i-1)3} - w_{i2})^+$, and $w_{i3} = (P_{i3} - 2\epsilon - 4\delta - w_{(i+1)1})^+$.
Since $P_{i1} - 2\epsilon - 4\delta - w_{(i-1)2} \leq w_{i1}$, $P_{i2} - 4\epsilon - 8\delta - w_{(i-1)3} \leq w_{i2}$ and $P_{i3} - 2\epsilon - 4\delta - w_{(i+1)1} \leq w_{i3}$, a lower bound for the total time to produce one part can be found as:
$\underline{T_{S^3_{26(\Pi^*_k)}} = 1/2(P + 8\epsilon + 8\delta) = \underline{T_2}}$.

$\mathbf{S^3_{34}}$: Total time of one repetition of this cycle is $16\epsilon + 22\delta + P_{i1} + w_{(i-1)3} + w_{i2} + w_{i3} + w_{(i+1)1} + P_{(i+1)2}$, where $w_{(i-1)3} = (P_{(i-1)3} - 4\epsilon - 6\delta - P_{i1})^+$,
$w_{i2} = (P_{i2} - 2\epsilon - 4\delta - w_{(i-1)3})^+$,
$w_{i3} = (P_{i3} - 2\epsilon - 6\delta)^+$, and $w_{(i+1)1} = (P_{(i+1)1} - 2\epsilon - 6\delta - w_{i3})^+$.
Since $P_{i2} - 2\epsilon - 4\delta - w_{(i-1)3} \leq w_{i2}$, $P_{i3} - 2\epsilon - 6\delta \leq w_{i3}$ and $0 \leq w_{(i+1)1}$, a lower bound for the total time to produce one part can be found as:
$\underline{T_{S^3_{34(\Pi^*_k)}} = 1/2(P + 12\epsilon + 12\delta) > \underline{T_2}}$.

$\mathbf{S^3_{35}}$: Total time of one repetition of this cycle is $16\epsilon + 20\delta + P_{i1} + w_{(i-1)3} + w_{i2} + P_{i3} + w_{(i+1)1} + P_{(i+1)2}$, where $w_{(i-1)3} = (P_{(i-1)3} - 4\epsilon - 6\delta - P_{i1})^+$,
$w_{i2} = (P_{i2} - 4\epsilon - 8\delta - w_{(i-1)3})^+$, and $w_{(i+1)1} = (P_{(i+1)1} - 4\epsilon - 6\delta - w_{i2} - P_{i3})^+$.
Since $P_{i2} - 4\epsilon - 8\delta - w_{(i-1)3} \leq w_{i2}$ and $0 \leq w_{(i+1)1}$, a lower bound for the total

time to produce one part can be found as:
$\underline{T_{S^3_{35(\Pi^*_k)}} = 1/2(P + 12\epsilon + 12\delta) > \underline{T_2}}$.

$\mathbf{S^3_{36}}$: Total time of one repetition of this cycle is $16\epsilon + 22\delta + P_{i1} + w_{(i+1)1} + w_{(i-1)3} + w_{i2} + w_{i3} + w_{(i+1)2}$, where $w_{(i-1)3} = (P_{(i-1)3} - 6\epsilon - 10\delta - P_{i1})^+$, $w_{i2} = (P_{i2} - 4\epsilon - 8\delta - w_{(i-1)3})^+$,
$w_{(i+1)1} = (P_{(i+1)1} - 4\epsilon - 8\delta - w_{(i-1)3} - w_{i2})^+$, $w_{i3} = (P_{i3} - 2\epsilon - 4\delta - w_{(i+1)1})^+$, and
$w_{(i+1)2} = (P_{(i+1)2} - 2\epsilon - 4\delta - w_{i3})^+$.
Since $P_{i2} - 4\epsilon - 8\delta - w_{(i-1)3} \leq w_{i2}$, $P_{i3} - 2\epsilon - 4\delta - w_{(i+1)1} \leq w_{i3}$ and $0 \leq w_{(i+1)2}$, a lower bound for the total time to produce one part can be found as:
$\underline{T_{S^3_{36(\Pi^*_k)}} = 1/2(P + 10\epsilon + 10\delta) > \underline{T_2}}$.

$\mathbf{S^3_{45}}$: Total time of one repetition of this cycle is $16\epsilon + 22\delta + w_{(i-1)2} + P_{(i-1)3} + w_{i1} + P_{i2} + w_{i3} + w_{(i+1)1}$, where $w_{(i-1)2} = (P_{(i-1)2} - 2\epsilon - 4\delta)^+$, $w_{i1} = (P_{i1} - 4\epsilon - 6\delta - w_{(i-1)2} - P_{(i-1)3})^+$,
$w_{i3} = (P_{i3} - 2\epsilon - 6\delta)^+$, and $w_{(i+1)1} = (P_{(i+1)1} - 2\epsilon - 6\delta - w_{i3})^+$.
Since $P_{i1} - 4\epsilon - 6\delta - w_{(i-1)2} - P_{(i-1)3} \leq w_{i1}$, $P_{i3} - 2\epsilon - 6\delta \leq w_{i3} =$ and $0 \leq w_{(i+1)1}$, a lower bound for the total time to produce one part can be found as:
$\underline{T_{S^3_{45(\Pi^*_k)}} = 1/2(P + 10\epsilon + 10\delta) > \underline{T_2}}$.

$\mathbf{S^3_{46}}$: Total time of one repetition of this cycle is $16\epsilon + 20\delta + P_{i1} + w_{(i-1)3} + w_{i2} + P_{i3} + w_{(i+1)1} + w_{(i+1)2}$, where $w_{(i-1)3} = (P_{(i-1)3} - 6\epsilon - 10\delta - P_{i1})^+$, $w_{i2} = (P_{i2} - 4\epsilon - 8\delta - w_{(i-1)3})^+$, and
$w_{(i+1)1} = (P_{(i+1)1} - 6\epsilon - 10\delta - w_{(i-1)3} - w_{i2} - P_{i3})^+$.
Since $P_{i2} - 4\epsilon - 8\delta - w_{(i-1)3} \leq w_{i2}$ and $0 \leq w_{(i+1)1}$, a lower bound for the total time to produce one part can be found as:

$\underline{T_{S^3_{46(\Pi^*_k)}}} = 1/2(P + 12\epsilon + 12\delta) > \underline{T_2}$.

$\mathbf{S^3_{56}}$: Total time of one repetition of this cycle is $16\epsilon + 22\delta + w_{(i-1)2} + w_{i1} + w_{(i-1)3} + w_{i2} + P_{i3} + w_{(i+1)1}$, where $w_{(i-1)2} = (P_{(i-1)2} - 2\epsilon - 4\delta)^+$, $w_{i1} = (P_{i1} - 2\epsilon - 4\delta - w_{(i-1)2})^+$,
$w_{(i-1)3} = (P_{(i-1)3} - 4\epsilon - 8\delta - w_{i1})^+$, $w_{i2} = (P_{i2} - 4\epsilon - 8\delta - w_{(i-1)3})^+$, and
$w_{(i+1)1} = (P_{(i+1)1} - 6\epsilon - 10\delta - w_{(i-1)3} - w_{i2} - P_{i3})^+$.
Since $P_{i1} - 2\epsilon - 4\delta - w_{(i-1)2} \leq w_{i1}$, $P_{i2} - 4\epsilon - 8\delta - w_{(i-1)3} \leq w_{i2}$ and $0 \leq w_{(i+1)1}$, a lower bound for the total time to produce one part can be found as:
$\underline{T_{S^3_{56(\Pi^*_k)}}} = 1/2(P + 10\epsilon + 10\delta) > \underline{T_2}$.

Thus, for any 2-unit cycle, S^3_{ij}, we showed that $\underline{T_2} \leq \underline{T_{S^3_{ij(\Pi^*_k)}}}$. Consequently, we can use $\underline{T_2}$ as a global lower bound for the 2-unit robot move cycles. □

Appendix D

Cycle time calculations

We will find the cycle times of S_1^2, S_2^2 and $S_{12}S_{21}$ for a given k-allocation type. Let us start with S_1^2. Sethi et al. [86] proved the cycle time of S_1^2 to be:

$$T_{S_1^2} = 6\epsilon + 6\delta + a + b,$$

where a and b are the processing times of the part on the first and second machines respectively. With our notation this cycle time is the following

$$T_{S_1^2} = 6\epsilon + 6\delta + P + P^{M1} + P^{M2}, \tag{D.1}$$

and it does not depend on the allocation type. So whatever the allocation is, this cycle time is the same.

Now consider S_2^2. Sethi et al. [86] also provided that:

$$T_{S_2^2} = 6\epsilon + 8\delta + max\{0, a - (2\epsilon + 4\delta), b - (2\epsilon + 4\delta)\},$$

where a and b are defined as above. With our definitions the above equation is the cycle time of S_2^2 with one-allocation type where, $a = P^{M1} + P_{11}$ and $b = P^{M2} + P_{12}$. Then the cycle time of a one-allocation for cycle S_2^2 with our notation is the following:

$$T_{S_2^2(\Pi_1)} = 6\epsilon + 8\delta + max\{0, P^{M1} + P_{11} - (2\epsilon + 4\delta), P^{M2} + P_{12} - (2\epsilon + 4\delta)\}. \tag{D.2}$$

Now consider a two-allocation type for the cycle S_2^2: for a part with the first allocation type, the processing time on the first machine is $P^{M1} + P_{11}$ and on the second machine is $P^{M2} + P_{12}$. For a part with the second allocation type, the processing times on the first and the second machines are $P^{M1} + P_{21}$ and $P^{M2} + P_{22}$ respectively. Then, the long run average cycle time to produce one part with cycle S_2^2 with a specific allocation matrix Π_2 is:

$$\begin{aligned} T_{S_2^2(\Pi_2)} = \; & 6\epsilon + 8\delta \\ & +1/2(max\{0, P^{M1} + P_{11} - (2\epsilon + 4\delta), P^{M2} + P_{22} - (2\epsilon + 4\delta)\}) \\ & +1/2(max\{0, P^{M1} + P_{21} - (2\epsilon + 4\delta), P^{M2} + P_{12} - (2\epsilon + 4\delta)\}). \end{aligned} \tag{D.3}$$

We can easily generalize this for a k-allocation type for the cycle S_2^2 as follows:

$$\begin{aligned} T_{S_2^2(\Pi_k)} = \; & 6\epsilon + 8\delta \\ & +1/k(max\{0, P^{M1} + P_{11} - (2\epsilon + 4\delta), P^{M2} + P_{k2} - (2\epsilon + 4\delta)\}) \\ & +1/k(max\{0, P^{M1} + P_{21} - (2\epsilon + 4\delta), P^{M2} + P_{12} - (2\epsilon + 4\delta)\}) \\ & +1/k(max\{0, P^{M1} + P_{31} - (2\epsilon + 4\delta), P^{M2} + P_{22} - (2\epsilon + 4\delta)\}) \\ & +\ldots \\ & +1/k(max\{0, P^{M1} + P_{k1} - (2\epsilon + 4\delta), P^{M2} + P_{(k-1)2} - (2\epsilon + 4\delta)\}). \end{aligned} \tag{D.4}$$

Now let us consider cycle $S_{12}S_{21}$. The long run average cycle time for one-allocation type for the cycle $S_{12}S_{21}$ can be derived as follows: The activity sequence of this robot move cycle is $A_0A_1A_0A_2A_1A_2$. Initially both machines are empty and the robot is in front of the input buffer just starting to take a part. The robot takes a part from the input buffer transports it to the first

machine and loads it, $(\epsilon + \delta + \epsilon)$; waits in front of the machine to finish the processing of the part, $(P^{M1}+P_{11})$; unloads the part transports it to the second machine and loads it, $(\epsilon+\delta+\epsilon)$; returns back to the input buffer, takes another part, transports it to the first machine and loads it, $(2\delta + \epsilon + \delta + \epsilon)$; travels to the second machine, (δ); waits if necessary, (w_{12}); unloads the machine, transports the part to the output buffer and drops it $(\epsilon + \delta + \epsilon)$; travels to the first machine, (2δ); waits if necessary, (w_{11}); unloads the machine, transports the part to the second machine and loads it, $(\epsilon + \delta + \epsilon)$; waits for the machine to finish the processing, $(P^{M2} + P_{12})$; unloads the machine, transports it to the output buffer, drops the part and returns back to the input buffer so that the initial and the final states are the same, $(\epsilon + \delta + \epsilon + 3\delta\ )$. As a result, total time to produce two parts is:

$$T_{S_{12}S_{21}(\Pi_1)} = 12\epsilon + 14\delta + P + P^{M1} + P^{M2} + w_{11} + w_{12},$$

where $w_{11} = max\{0, P^{M1} + P_{11} - (2\epsilon + 4\delta + w_{12})\}$ and $w_{12} = max\{0, P^{M2} + P_{12} - (2\epsilon + 4\delta)\}$.
In other words, $w_{11}+w_{12} = max\{0, P^{M1}+P_{11}-(2\epsilon+4\delta), P^{M2}+P_{12}-(2\epsilon+4\delta)\}$.

Hence the cycle time for one-allocation for the cycle $S_{12}S_{21}$ is:

$$\begin{aligned} T_{S_{12}S_{21}(\Pi_1)} &= \tfrac{1}{2}\left(2\epsilon + 14\delta + P + P^{M1} + P^{M2}\right) \\ &\quad + \tfrac{1}{2}\left(max\{0, P^{M1} + P_{11} - (2\epsilon + 4\delta), P^{M2} + P_{12} - (2\epsilon + 4\delta)\}\right). \end{aligned} \tag{D.5}$$

Now consider the case when we have two-allocation types:

$$T_{S_{12}S_{21}(\Pi_2)} = \frac{12\epsilon + 14\delta + P^{M1} + P^{M2} + P_{11} + P_{22} + w_{12} + w_{21}}{2},$$

where $w_{12} = max\{0, P^{M1} + P_{12} - (2\epsilon + 4\delta + w_{21})\}$ and $w_{21} = max\{0, P^{M2} +$

$P_{21} - (2\epsilon + 4\delta)\}$. In other words,

$$\begin{aligned} T_{S_{12}S_{21}(\Pi_2)} = \quad & 1/2(12\epsilon + 14\delta + P^{M1} + P^{M2} + P_{11} + P_{22}) \\ & +1/2(max\{0, P^{M1} + P_{21} - (2\epsilon + 4\delta), P^{M2} + P_{12} - (2\epsilon + 4\delta)\}). \end{aligned} \tag{D.6}$$

Appendix E

Proof of Theorem 6.4

The following definition will play a crucial role in the forthcoming context.

Definition E.1 State 0 *is defined to be the state in which both machines are empty and the robot is in front of the input buffer.*

Consider any n-unit robot move cycle during which State 0 is encountered. Clearly, the allocations immediately preceding and those immediately following this state can be treated as completely independent from each other. As far as the cycle time computations are involved, the contribution of the portion of the cycle between two State 0's is simply additive. This observation deserves further attention:

Fact E.1 *The average cycle time of an n-unit robot move cycle is simply the average of the cycle time corresponding to a sub-cycle between two State 0's and the cycle time of the remaining cycle after the sub-cycle is extracted.*

After inspecting the cycles we have introduced thus far, it is easy to state that:

Fact E.2 *S_1 cycle starts and ends with State 0. S_{12} sequence starts with State 0 and S_{21} sequence ends with State 0.*

With these observations, we are now ready to proceed with the proof of Theorem 6.4.

Proof. Hall et al. [43] showed that any n-unit robot move cycle can be represented by the four robot move sequences: S_1^2, S_2^2, S_{12} and S_{21}. However, not all of these sequences can follow any other since otherwise the basic feasibility assumptions of Crama et al. [21] stating that the robot cannot load an already loaded machine and cannot unload an already empty machine may be violated. Figure 6.1 depicts the feasible transitions from one sequence to another. Now let us consider any n-unit robot move cycle with the optimal allocation type. In its generality, we assume this cycle to contain at least one of these four robot move sequences. The allocation of the operations does not affect the loading/unloading and transportation times but only affects the processing times which in turn affect the waiting times of the robot in front of the machines. Let us first analyze one of the S_1^2 cycles within this n-unit robot move cycle. Using Facts E.1 and E.2 above we conclude that the allocation for this cycle does not affect the remaining part of the cycle. Also, as mentioned previously, there is no allocation problem for S_1^2 cycle, and the waiting time for S_1^2 is $P + P^{M1} + P^{M2}$, independent of the allocation. When we remove the activity sequence $A_0A_1A_2$, corresponding to S_1^2 from the activity sequence of the n-unit robot move cycle, we get a new feasible $(n-1)$-unit robot move cycle for which the optimal allocation for the remaining parts does not change.

Assume that there are a total of z S_1^2 cycles within the n-unit robot move cycle where $z = 0, 1, \ldots, n$. Let T_{n-z} be the cycle time of the new $(n-z)$-unit robot move cycle, say C_{n-z}, which is attained by removing the activity sequences of all S_1^2 cycles. Hence, the cycle time of our n-unit robot move cycle

is:

$$T_n = \frac{z * T_{S_1^2} + (n-z) * T_{n-z}}{n} .$$

This equation states that $T_n \geq min\{T_{S_1^2}, T_{n-z}\}$.

Now let us consider C_{n-z}. From Figure 6.1 we see that between two S_{12} sequences one S_{21} sequence must be performed. Between one S_{12} and one S_{21} any number of S_2^2 sequences can be performed. Thus we can partition C_{n-z} into sub-cycles each of which starts with S_{12} performs a number of S_2^2's and ends up with S_{21}. Furthermore, as stated as Fact E.1 above, the allocation for such cycles does not affect the allocation for the remaining part of the cycle and is not affected by the allocation of the remaining part of the cycle. The cycle time of the whole $(n-z)$-unit robot move cycle is a convex combination of the cycle times of these sub-cycles. Let T_i be the cycle time of the ith sub-cycle with the optimal allocation type then, clearly, $T_{n-z} \geq min_i\{T_i\}$. Henceforth, $T_n \geq min\{T_{S_1^2}, min_i\{T_i\}\}$.

Now let us consider any such cycle which starts with S_{12}, performs a total of l S_2^2's $l = 0, 1, \ldots$ and ends up with S_{21}. Note that this is an $(l+2)$-unit cycle. Let us represent this cycle as $S_{12}(l - S_2^2)S_{21}$. Since in every cycle each machine is loaded and unloaded an equal number of times, the average loading/unloading time to produce one part for all cycles are equal to each other, namely, 6ϵ. Repeating cycle S_2^2 l times requires a total of $8l\delta$. Repeating S_{12} one time, which has an activity sequence of $A_0A_1A_0$, requires 5δ and the following activity of this sequence can only be A_2 which requires an additional δ to travel between final state of S_{12} (first machine) to the initial state of the next sequence (second machine). In a similar way repeating S_{21} $(A_2A_1A_2)$ one time requires 5δ and an additional 3δ to travel between the last state of S_{21} (output buffer) to initial state of the following sequence which must start with activity A_0 (input buffer). Then the average travel time to produce one part

with the cycle $S_{12}(l - S_2^2)S_{21}$ is:

$$\frac{6\delta + 8l\delta + 8\delta}{l+2} = \frac{8l+14}{l+2}\delta.$$

The last component in our cycle time representations is the total waiting time of the robot in front of the machines. We can find the waiting time as follows: Assume for the cycle $S_{12}(l - S_2^2)S_{21}$ using k allocation types is optimal and consider the specific allocation matrix Π_k. The first part will be produced according to the cycle S_{12} which implies a full waiting time in front of the first machine: $P^{M1} + P_{11}$, and will be processed according to S_2^2 on the second machine which means a partial waiting time in front of the second machine: $w_{12} = max\{0, P^{M2} + P_{12}) - (2\epsilon + 4\delta)\}$. The next l parts will be processed according to S_2^2 cycle on both machines. The partial waiting time for the second part on the first machine is: $w_{21} = max\{0, P^{M1} + P_{21} - (2\epsilon + 4\delta + w_{12})\}$. Then we have:

$$w_{12} + w_{21} = max\{0, P^{M1} + P_{12} - (2\epsilon + 4\delta), P^{M2} + P_{21} - (2\epsilon + 4\delta)\}.$$

Proceeding in the same way for all parts, total waiting time to produce $l + 2$ parts is:

$$\begin{aligned}
&P^{M1} + P^{M2} + P_{11} + P_{(l+2)2} \\
&+max\{0, P^{M1} + P_{21} - (2\epsilon + 4\delta), P^{M2} + P_{12} - (2\epsilon + 4\delta)\} \\
&+max\{0, P^{M1} + P_{31} - (2\epsilon + 4\delta), P^{M2} + P_{22} - (2\epsilon + 4\delta)\} \\
&+max\{0, P^{M1} + P_{41} - (2\epsilon + 4\delta), P^{M2} + P_{32} - (2\epsilon + 4\delta)\} \\
&+ \ldots \\
&+max\{0, P^{M1} + P_{(l+1)1} - (2\epsilon + 4\delta), P^{M2} + P_{l2} - (2\epsilon + 4\delta)\} \\
&+max\{0, P^{M1} + P_{(l+2)1} - (2\epsilon + 4\delta), P^{M2} + P_{(l+1)2} - (2\epsilon + 4\delta)\}.
\end{aligned}$$

In the above equation let us insert P_{11} and $P_{(l+2)2}$ into the first and last max terms respectively. By this change all other max terms remain the same but

these two become:
$max\{P_{11}, P^{M1} + P_{21} + P_{11} - (2\epsilon + 4\delta), P^{M2} + P - (2\epsilon + 4\delta)\}$ and
$max\{P_{(l+2)2}, P^{M1} + P - (2\epsilon + 4\delta), P^{M2} + P_{(l+1)2} + P_{(l+2)2} - (2\epsilon + 4\delta)\}$.
Under the optimal allocation we must have $P^*_{11} = 0$ and $P_{(l+2)2} = 0$. This means for the first part allocating all operations that are in set O to the second machine and for the last part allocating them to the first machine. Furthermore, since we assumed that $P^{M1} \geq P^{M2}$, letting $P_{(l+1)2} = P$ does not change the cycle time. As a result of these, the last max term of the above equation becomes:

$$max\{0, P^{M1} + P - (2\epsilon + 4\delta), P^{M2} + P - (2\epsilon + 4\delta)\} = max\{0, P^{M1} + P - (2\epsilon + 4\delta)\}.$$

After including the loading/unloading and travel times, the cycle time of $S_{12}(l - S_2^2)S_{21}$ under optimal allocation matrix Π^*_k becomes:

$$6\epsilon + \tfrac{(14+8l)}{(l+2)}\delta + \tfrac{1}{(l+2)}(P^{M1} + P^{M2}) \qquad (E.1)$$

$$+1/(l+2)(max\{0, P^{M1} + P^*_{21} - (2\epsilon + 4\delta), P^{M2} + P - (2\epsilon + 4\delta)\}) \qquad (E.2)$$

$$+1/(l+2)(max\{0, P^{M1} + P^*_{31} - (2\epsilon + 4\delta), P^{M2} + P^*_{22} - (2\epsilon + 4\delta)\}) \qquad (E.3)$$

$$+1/(l+2)(max\{0, P^{M1} + P^*_{41} - (2\epsilon + 4\delta), P^{M2} + P^*_{32} - (2\epsilon + 4\delta)\}) \qquad (E.4)$$

$$+\ldots$$

$$+1/(l+2)(max\{0, P^{M1} - (2\epsilon + 4\delta), P^{M2} + P^*_{l2})) - (2\epsilon + 4\delta)\}) \qquad (E.(l+1))$$

$$+1/(l+2)(max\{0, P^{M1} + P - (2\epsilon + 4\delta)\}) \qquad (E.(l+2)).$$

Let $W(S_{12}S_{21})$ represent the waiting time of two-allocation for cycle $S_{12}S_{21}$ with optimal allocation Π^*, which in equation (6.3) was found to be: $W(S_{12}S_{21}) = \frac{1}{2}(max\{0, P + P^{M1} - (2\epsilon + 4\delta)\})$. Then, the $(E.(l+2))^{nd}$ component in the above representation is $\frac{2}{(l+2)}W(S_{12}S_{21})$. Consider lines $(E.2)$ through $(E.(l+1))$ above. They share the same pattern as the waiting time

component corresponding to an l-allocation type for the cycle S_2^2 in (D.4). Let $W(S_2^2)$ be this waiting time component in (D.4) when we plug in the respective allocations from $(E.2)$ through $(E.(l+1))$. The summation of the components $(E.2)$ through $(E.(l+1))$ is then $\frac{l}{l+2}W(S_2)^2$. In conclusion,

$$T_{S_{12}(l-S_2^2)S_{21}} = \frac{2(6\epsilon + 7\delta + \frac{P^{M1}+P^{M2}}{2} + W(S_{12}S_{21})) + l(6\epsilon + 8\delta + W(S_2^2))}{l+2}.$$

In particular, the cycle time of $S_{12}(l - S_2^2)S_{21}$ is a convex combination of the cycle times of two-allocation for $S_{12}S_{21}$ and l-allocation for S_2^2. Clearly,

$$T_{S_{12}(l-S_2^2)S_{21}} \geq min\{T_{S_2^2(\Pi^*)}, T_{S_{12}S_{21}(\Pi^*)}\}.$$

Finally, we have managed to show that the cycle time of the n-unit cycle we started out with, $T_n \geq min\{T_{S_1^2}, T_{S_2^2(\Pi^*)}, T_{S_{12}S_{21}(\Pi^*)}\}$. □

Appendix F

1-unit cycles for 3-machine cells

Here we will present the robot activity sequences and the cycle times of the six feasible 1-unit cycles for a 3-machine robotic cell.

$S_1^3 : A_0A_1A_2A_3 : 8\epsilon + 8\delta + P_1 + P_2 + P_3,$

$S_2^3 : A_0A_2A_1A_3 : \max\{8\epsilon + 12\delta, P_1 + 6\epsilon + 8\delta, P_2 + 4\epsilon + 4\delta, P_3 + 6\epsilon + 8\delta, (P_1 + P_2 + P_3)/2 + 4\epsilon + 4\delta\},$

$S_3^3 : A_0A_1A_3A_2 : \max\{P_1 + 8\epsilon + 10\delta, P_1 + P_2 + 6\epsilon + 6\delta, P_3 + 4\epsilon + 4\delta\},$

$S_4^3 : A_0A_3A_1A_2 : \max\{P_1 + P_2 + 6\epsilon + 6\delta, P_2 + 8\epsilon + 12\delta, P_2 + P_3 + 6\epsilon + 6\delta\},$

$S_5^3 : A_0A_2A_3A_1 : \max\{P_1 + 4\epsilon + 4\delta, P_2 + P_3 + 6\epsilon + 6\delta, P_3 + 8\epsilon + 10\delta\},$

$S_6^3 : A_0A_3A_2A_1 : \max\{8\epsilon + 12\delta, P_1 + 4\epsilon + 4\delta, P_2 + 4\epsilon + 4\delta, P_3 + 4\epsilon + 4\delta\}.$

Appendix G

Computational results

					R_1						
Factors			Rep	N	$= 0$	> 0	Avg	Case Avg	< 0	Avg	Case Avg
B	C	D					($\times 10^{-6}$)	($\times 10^{-6}$)		($\times 10^{-6}$)	($\times 10^{-6}$)
			1	20	0	3	0,214		17	-6,378	
			2	20	0	3	0,159		17	-6,542	
0	0	0	3	20	0	3	0,310	0,260	17	-11,708	-6,629
			4	20	0	2	0,404		18	-4,175	
			5	20	0	0	-		20	-4,810	
			1	20	0	2	0,079		18	-3,323	
			2	20	0	7	0,400		13	-0,592	
1	0	0	3	20	0	7	0,368	0,338	13	-4,542	-3,476
			4	20	0	0	-		20	-6,659	
			5	20	0	2	0,275		18	-1,404	
			1	20	1	3	1,969		16	-7,752	
			2	20	0	3	0,317		17	-25,227	
0	1	0	3	20	0	6	0,304	0,608	14	-9,843	-12,507
			4	20	0	4	0,568		16	-12,986	
			5	20	0	4	0,301		16	-5,598	
			1	20	0	2	0,483		18	-6,115	
			2	20	0	1	0,651		19	-5,787	
1	1	0	3	20	0	3	0,396	0,589	17	-4,571	-7,934
			4	20	0	1	0,886		19	-9,642	
			5	20	0	4	0,698		16	-14,073	
			1	20	0	0	-		20	-5,875	
			2	20	1	0	-		19	-10,855	
0	0	1	3	20	0	1	0,027	0,092	19	-12,914	-9,115
			4	20	0	2	0,079		18	-5,938	
			5	20	0	1	0,182		19	-9,994	
			1	20	0	1	0,077		19	-1,657	
			2	20	1	4	0,173		15	-0,214	
1	0	1	3	20	0	1	0,068	0,156	19	-1,622	-2,745
			4	20	0	1	0,127		19	-4,172	
			5	20	0	1	0,289		19	-5,529	
			1	20	0	1	6,721		19	-897,315	
			2	20	0	0	-		20	-8,403	
0	1	1	3	20	1	2	0,229	1,127	17	-13,857	-195,518
			4	20	0	5	0,368		15	-3,099	
			5	20	0	0	-		20	-14,652	
			1	20	1	3	0,032		16	-2,890	
			2	20	0	0	-		20	-9,345	
1	1	1	3	20	0	3	0,098	0,099	17	-3,364	-7,161
			4	20	1	2	0,186		17	-15,743	
			5	20	0	2	0,115		18	-4,013	

Table G.1: Comparison of EFFRONT with DICOPT for 20 operations

					R_2						
Factors			Rep	N	$= 0$	> 0	Avg	Case Avg	< 0	Avg	Case Avg
B	C	D					($\times 10^{-6}$)	($\times 10^{-6}$)		($\times 10^{-6}$)	($\times 10^{-6}$)
			1	20	1	9	0,803		10	-3,060	
			2	20	0	10	1,024		10	-7,806	
0	0	0	3	20	1	12	1,128	1,012	7	-11,580	-5,665
			4	20	0	12	0,611		8	-4,601	
			5	20	0	12	1,447		8	-2,134	
			1	20	0	9	0,227		11	-4,528	
			2	20	0	9	0,741		11	-0,597	
1	0	0	3	20	0	9	0,599	0,484	11	-5,168	-3,448
			4	20	0	7	0,454		13	-6,337	
			5	20	0	8	0,380		12	-0,365	
			1	20	0	16	1,472		4	-23,060	
			2	20	0	15	1,281		5	-65,739	
0	1	0	3	20	0	16	1,316	1,287	4	-32,734	-35,440
			4	20	0	15	1,265		5	-34,156	
			5	20	0	16	1,102		4	-14,261	
			1	20	0	6	0,560		14	-7,652	
			2	20	0	9	0,819		11	-8,100	
1	1	0	3	20	0	12	0,784	0,807	8	-1,278	-8,959
			4	20	0	8	0,951		12	-9,068	
			5	20	0	5	0,905		15	-14,818	
			1	20	0	10	1,103		10	-6,296	
			2	20	0	7	0,779		13	-12,234	
0	0	1	3	20	0	11	0,531	0,847	9	-25,476	-10,175
			4	20	0	9	0,971		11	-3,442	
			5	20	0	9	0,878		11	-5,482	
			1	20	0	5	0,290		15	-0,771	
			2	20	0	5	0,144		15	-0,204	
1	0	1	3	20	0	7	0,089	0,170	13	-1,138	-1,526
			4	20	0	9	0,139		11	-0,923	
			5	20	0	9	0,214		11	-5,419	
			1	20	0	1	8,337		19	-892,034	
			2	20	0	9	0,904		11	-7,725	
0	1	1	3	20	0	10	0,577	0,854	10	-20,499	-306,385
			4	20	0	12	0,512		8	-4,048	
			5	20	0	11	0,756		9	-21,440	
			1	20	0	13	0,293		7	-4,977	
			2	20	0	4	0,199		16	-6,256	
1	1	1	3	20	0	12	0,347	0,280	8	-3,764	-7,962
			4	20	0	5	0,107		15	-15,275	
			5	20	0	14	0,296		6	-3,317	

Table G.2: Comparison of EFFRONT with BARON for 20 operations

					R_3					
Factors			Rep	N	$= 0$	> 0	Avg	Case Avg	< 0	Case Avg
B	C	D					($\times 10^{-6}$)	($\times 10^{-6}$)		($\times 10^{-6}$)
			1	20	2	18	4,690		0	-
			2	20	3	17	2,524		0	-
0	0	0	3	20	2	18	7,255	4,436	0	-
			4	20	1	19	2,361		0	-
			5	20	2	18	5,360		0	-
			1	20	2	18	0,660		0	-
			2	20	4	16	0,312		0	-
1	0	0	3	20	3	17	0,295	1,140	0	-
			4	20	2	18	2,999		0	-
			5	20	2	18	1,299		0	-
			1	20	1	19	2,602		0	-
			2	20	2	18	6,580		0	-
0	1	0	3	20	4	16	1,630	3,371	0	-
			4	20	3	17	3,159		0	-
			5	20	2	18	2,719		0	-
			1	20	7	13	0,411		0	-
			2	20	2	18	1,532		0	-
1	1	0	3	20	1	19	3,985	2,557	0	-
			4	20	2	18	4,506		0	-
			5	20	12	8	0,579		0	-
			1	20	1	19	3,452		0	-
			2	20	1	19	2,772		0	-
0	0	1	3	20	1	19	1,152	3,818	0	-
			4	20	3	17	4,565		0	-
			5	20	1	19	7,227		0	-
			1	20	1	19	1,120		0	-
			2	20	10	10	0,018		0	-
1	0	1	3	20	1	19	0,873	1,828	0	-
			4	20	1	19	3,697		0	-
			5	20	2	18	2,615		0	-
			1	20	2	18	5,667		0	-
			2	20	2	18	5,068		0	-
0	1	1	3	20	3	17	2,111	3,999	0	-
			4	20	3	17	1,082		0	-
			5	20	1	19	5,705		0	-
			1	20	2	18	0,839		0	-
			2	20	1	19	4,610		0	-
1	1	1	3	20	2	18	1,720	2,512	0	-
			4	20	2	18	2,149		0	-
			5	20	2	18	3,125		0	-

Table G.3: Comparison of DICOPT with BARON for 20 operations

					R_1						
Factors			Rep	N	$= 0$	> 0	Avg	Case Avg	< 0	Avg	Case Avg
B	C	D					($\times 10^{-6}$)	($\times 10^{-6}$)		($\times 10^{-6}$)	($\times 10^{-6}$)
			1	20	1	2	0,027		17	-2,960	
			2	20	0	3	0,088		17	-1,210	
0	0	0	3	20	0	1	0,220	0,088	19	-4,744	-2,603
			4	20	0	3	0,081		17	-0,421	
			5	20	0	1	0,101		19	-3,343	
			1	20	0	5	0,087		15	-0,269	
			2	20	0	7	0,131		13	-0,152	
1	0	0	3	20	0	7	0,121	0,125	13	-0,122	-0,174
			4	20	0	6	0,217		14	-0,139	
			5	20	0	8	0,079		12	-0,175	
			1	20	1	2	0,049		17	-0,464	
			2	20	0	1	0,178		19	-10,434	
0	1	0	3	20	0	4	0,178	0,131	16	-1,991	-3,656
			4	20	0	2	0,100		18	-2,284	
			5	20	0	1	0,120		19	-2,435	
			1	20	0	2	0,247		18	-1,756	
			2	20	1	1	0,022		18	-5,046	
1	1	0	3	20	0	2	0,198	0,162	18	-0,560	-1,864
			4	20	0	6	0,120		14	-0,570	
			5	20	0	12	0,174		8	-0,146	
			1	20	0	0	-		20	-13,079	
			2	20	0	1	0,057		19	-4,875	
0	0	1	3	20	0	11	0,065	0,054	9	-0,053	-6,334
			4	20	0	4	0,045		16	-0,115	
			5	20	0	3	0,024		17	-9,206	
			1	20	0	4	0,032		16	-0,392	
			2	20	0	7	0,070		13	-0,522	
1	0	1	3	20	1	6	0,042	0,069	13	-0,119	-0,264
			4	20	0	3	0,041		17	-0,143	
			5	20	0	13	0,098		7	-0,053	
			1	20	0	0	-		20	-2,628	
			2	20	0	4	0,070		16	-0,156	
0	1	1	3	20	0	1	0,052	0,067	19	-5,439	-3,078
			4	20	0	1	0,032		19	-4,075	
			5	20	0	2	0,088		18	-2,632	
			1	20	0	10	0,096		10	-0,107	
			2	20	0	2	0,048		18	-0,189	
1	1	1	3	20	0	3	0,073	0,077	17	-0,850	-0,453
			4	20	1	1	0,003		18	-0,492	
			5	20	0	2	0,058		18	-0,496	

Table G.4: Comparison of EFFRONT with DICOPT for 50 operations

Factors			Rep	N	R_1						
					$= 0$	> 0	Avg ($\times 10^{-6}$)	Case Avg ($\times 10^{-6}$)	< 0	Avg ($\times 10^{-6}$)	Case Avg ($\times 10^{-6}$)
B	C	D									
			1	20	0	6	0,072		14	-761,084	
			2	20	0	1	0,094		19	-9,120	
0	0	0	3	20	0	5	0,093	0,075	15	-0,176	-138,228
			4	20	0	3	0,080		17	-5,058	
			5	20	0	6	0,058		14	-0,208	
			1	20	0	3	0,089		17	-0,566	
			2	20	1	4	0,035		15	-0,399	
1	0	0	3	20	0	10	0,075	0,078	10	-0,192	-0,329
			4	20	0	5	0,078		15	-0,188	
			5	20	0	14	0,090		6	-0,068	
			1	20	0	7	0,068		13	-0,189	
			2	20	0	3	0,093		17	-0,923	
0	1	0	3	20	0	5	0,115	0,091	15	-3,022	-2,171
			4	20	0	5	0,088		15	-0,193	
			5	20	0	2	0,119		18	-5,720	
			1	20	0	5	0,022		15	-0,173	
			2	20	0	11	0,121		9	-0,114	
1	1	0	3	20	0	6	0,100	0,101	14	-0,141	-0,161
			4	20	0	6	0,092		14	-0,179	
			5	20	0	10	0,124		10	-0,186	
			1	20	0	2	0,012		18	-156,523	
			2	20	0	1	0,039		19	-0,357	
0	0	1	3	20	0	2	0,026	0,023	18	-2,159	-31,942
			4	20	1	1	0,037		18	-0,874	
			5	20	0	1	0,009		19	-3,153	
			1	20	0	2	0,060		18	-0,500	
			2	20	1	10	0,058		9	-0,033	
1	0	1	3	20	0	12	0,055	0,051	8	-0,054	-0,420
			4	20	0	11	0,041		9	-0,057	
			5	20	0	2	0,033		18	-0,879	
			1	20	0	9	0,065		11	-0,238	
			2	20	0	0	-		20	-26,136	
0	1	1	3	20	0	2	0,040	0,060	18	-23,952	-12,719
			4	20	0	6	0,056		14	-0,089	
			5	20	0	1	0,080		19	-4,484	
			1	20	0	9	0,059		11	-0,074	
			2	20	0	12	0,058		8	-0,041	
1	1	1	3	20	2	4	0,047	0,062	14	-0,178	-0,527
			4	20	0	3	0,055		17	-1,559	
			5	20	0	7	0,085		13	-0,234	

Table G.5: Comparison of EFFRONT with DICOPT for 80 operations

Appendix H

ANOVA

Factor			Sum of Squares	df	Mean Square	F	Sig.
B	E-D	Between Groups	421148,621	1	421148,621	7,86	0,005
		Within Groups	42756562,86	798	53579,653		
		Total	43177711,49	799			
	E-B	Between Groups	388350,586	1	388350,586	7,246	0,007
		Within Groups	42767898,24	798	53593,857		
		Total	43156248,83	799			
	D-B	Between Groups	664,829	1	664,829	23,348	0
		Within Groups	22723,051	798	28,475		
		Total	23387,88	799			
C	E-D	Between Groups	409857,987	1	409857,987	7,647	0,006
		Within Groups	42767853,5	798	53593,801		
		Total	43177711,49	799			
	E-B	Between Groups	406853,733	1	406853,733	7,595	0,006
		Within Groups	42749395,09	798	53570,671		
		Total	43156248,83	799			
	D-B	Between Groups	5,533	1	5,533	0,189	0,664
		Within Groups	23382,347	798	29,301		
		Total	23387,88	799			
D	E-D	Between Groups	360605,577	1	360605,577	6,721	0,01
		Within Groups	42817105,91	798	53655,521		
		Total	43177711,49	799			
	E-B	Between Groups	356145,59	1	356145,59	6,64	0,01
		Within Groups	42800103,24	798	53634,215		
		Total	43156248,83	799			
	D-B	Between Groups	13,889	1	13,889	0,474	0,491
		Within Groups	23373,991	798	29,291		
		Total	23387,88	799			

Table H.1: ANOVA tables for $p = 20$ operations

p	Factor		Sum of Squares	df	Mean Square	F	Sig.
50	B	Between Groups	1663,62	1	1663,62	68,81	0
		Within Groups	19293,329	798	24,177		
		Total	20956,949	799			
	C	Between Groups	0,264	1	0,264	0,01	0,92
		Within Groups	20956,684	798	26,262		
		Total	20956,949	799			
	D	Between Groups	26,185	1	26,185	0,998	0,318
		Within Groups	20930,764	798	26,229		
		Total	20956,949	799			
80	B	Between Groups	280756,559	1	280756,559	1,854	0,174
		Within Groups	120873263	798	151470,254		
		Total	121154019,6	799			
	C	Between Groups	200116,921	1	200116,921	1,32	0,251
		Within Groups	120953902,7	798	151571,307		
		Total	121154019,6	799			
	D	Between Groups	62555,897	1	62555,897	0,412	0,521
		Within Groups	121091463,7	798	151743,689		
		Total	121154019,6	799			

Table H.2: ANOVA tables for $p = 50$ and $p = 80$ operations

VITA

Hakan Gültekin was born on May 5, 1978 in Nevşehir, Türkiye. He received his high school education at Bursa Ulubatlı Hasan Anadolu Lisesi, Türkiye. He has received his B.S. and M.S. degrees from the Department of Industrial Engineering, Bilkent University in 2000 and 2002, respectively. Since September 2000, he has been working with Prof. Selim Aktürk for his graduate study at the same department. He is married with Feyza Gültekin and father of Ayşe Melek Gültekin and Abdulkadir Gültekin.