Cyclic Production of Flexible Manufacturing Cells

Mazyar Ghadiri Nejad

Submitted to the
Institute of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Industrial Engineering

Eastern Mediterranean University
February 2018
Gazimağusa, North Cyprus

Approval of the Institute of Graduate Studies and Research

Assoc. Prof. Dr. Ali Hakan Ulusoy
Acting Director

I certify that this thesis satisfies the requirements as a thesis for the degree of Doctor of Philosophy in Industrial Engineering.

Assoc. Prof. Dr. Gökhan İzbirak
Chair, Department of Industrial Engineering

We certify that we have read this thesis and that in our opinion it is fully adequate in scope and quality as a thesis for the degree of Doctor of Philosophy in Industrial Engineering.

Prof. Dr. Béla Vizvári
Supervisor

Examining Committee

1. Prof. Dr. Serpil Erol
2. Prof. Dr. Murat Taner Testik
3. Prof. Dr. Béla Vizvári
4. Assoc. Prof. Dr. Gökhan İzbirak
5. Assoc. Prof. Dr. Adham Makkie

ABSTRACT

This thesis deals with two different flexible manufacturing cells. Both cells contain *m* identical computer numerical control (CNC) machines that are able to perform all the processes to produce a final product. The CNC machines are set up in a line layout. In the both cases, one input station and one output station exists at the beginning and at the end of the line, respectively. The items to be processed are kept in the input station, and the finished items are kept in the output station. In the second case, in addition to the input station, there is an individual input buffers attached to each machine. Using these buffers, each machine can be consecutively loaded twice in a cycle. In the both cells, a robot serves the machines and transports parts from the input station to a machine, loads the machine, and unloads the machine, after finishing its process, and puts the processed part in the output station.

In these cells, *m* different parts will be processed in every cycle. Each part is processed completely by one machine. If the system is at a specific state at the beginning of a cycle, it reaches the same state at the end of the cycle, and then repeats the same actions in the same order in the subsequent cycles. To show all of the possible cycles in such cells, a sequential part production matrix is presented considering a general case. The duration of a cycle is called cycle time. The objective function of both cell types is to find the order of robot operations that minimizes the cycle time which maximizes the long-run average throughput rate of each cell.

For the first case, a new mathematical model is presented to optimize the system. A reduced version of the new model is also provided. The reduced version is still an exact model of the minimization of the cycle time, however it does not determine the waiting times of the robot directly. These two models are more effective than the previous existed exact models in the literature. The solution of the reduced model requires significantly less CPU time comparing to the other models. A metaheuristic algorithm based on simulated annealing algorithm is proposed. In order to compute the minimum cycle time in each iteration of the algorithm, a linear programming model is needed to be solved which is the first case in the literature to the best of our knowledge. A new proof is provided for the lower bound of cycle time. This new proof facilitates the optimality analysis of several sequences of the robot movements.

For the second case, a mathematical model is presented to optimize the cyclic production. A two-machine cell is discussed in details. In addition to some lower bounds of the cycle time for different orders of robot movements, the optimal cycles and upper bounds for the cases with different activities are also investigated.

Keywords: Flexible manufacturing, CNC machines, Robotic cell, Cyclic scheduling, Metaheuristics.

ÖZ

Bu tez iki farklı esnek imalat hücresi ile ilgilidir. Her iki hücre de, nihai bir ürün üretmek için tüm işlemleri gerçekleştirebilen, aynı sayıda Bilgisayarlı Sayısal Dentim (CNC) makinesini içerir. CNC makineleri bir çizgi düzeninde ayarlanır. Her iki durumda da sıranın başında ve sonunda bir giriş istasyonu ve bir çıkış istasyonu bulunur. İşlenecek parçalar giriş istasyonunda tutulur ve bitmiş parçalar çıkış istasyonunda tutulur. İkinci durumda, giriş istasyonuna ek olarak, her bir makineye bağlı tek bir giriş tamponları bulunur. Bu tamponları kullanarak, her makine ardışık olarak bir döngüde iki kez yüklenebilir. Her iki hücrede de, robot makinelere hizmet eder ve parçaları giriş istasyonundan makineye nakleder, makineyi yükler ve işlemi tamamladıktan sonra makineyi boşaltır ve işlenmiş kısmı çıkış istasyonuna koyar.

Bu hücrelerde, her döngüde m sayıda farklı parça işlenecektir. Her parça bir makina tarafından tamamen işlenir. Sistem bir çevrimin başlangıcında belirli bir konumdaysa, çevrimin sonunda da aynı duruma geçer ve aynı aktiviteleri sonraki çevrimlerde de aynı sırada tekrarlar. Bu tür hücrelerdeki olası döngülerin tümünü de göstermek için, genel durum göz önüne aknarak ardışık parça üretim matrisi sunulmaktadır. Bir döngünün süresi döngü süresi olarak adlandırılır. Her iki hücre tipinin amaç fonksiyonu, her bir hücrenin uzun dönem ortalama çıktı oranını maksimize eden döngü süresini en aza indirecek robot işlemlerinin sırasını bulmaktır.

İlk durumda, sistemi optimize etmek için yeni bir matematiksel model sunulmuştur. Yeni modelin indirgenmiş bir versiyonu da gösterilmiştir. İndirgenmiş versiyon, döngü süresinin en aza indirgenmesinin hala kesin bir modeli olmasına rağmen robotun bekleme sürelerini doğrudan belirlememektedir. Bu iki model literatürde daha önce var olan kesin modellere göre daha etkilidir. İndirgenmiş modelin çözümü, diğer modellere kıyasla çok daha az CPU zamanı gerektirir. Benzetimli tavlama algoritmasına dayanan sezgi ötesi bir algoritma önerilmiştir. Algoritmanın her yinelenmesinde minimum döngü süresini hesaplamak için doğrusal bir programlama modelinin çözülmesi gerekmektedir. Döngü süresinin alt sınırı için yeni bir ispat gösterilmiştir. Bu yeni ispat, robot hareketlerinin birkaç sırasının optimallik analizini kolaylaştırımaktadır.

İkinci durumda, döngüsel üretimi optimize etmek için bir matematiksel model sunulmuştur. İki makine hücresi ayrıntılı olarak ele alınmıştır. Robot hareketlerinin farklı sıraları için döngü zamanının bazı alt sınırlarına ilaveten, farklı aktivitelere sahip durumlar için optimal çevrimler ve üst sınırlar da araştırılmıştır.

Anahtar Kelimeler: Esnek üretim, CNC makineleri, Robotik hücre, Çevrimsel çizelgeleme, Metaheuristik, Sezgi ötesi.

DEDICATION

To My Devoted Wife

and

My Family

ACKNOWLEDGMENT

I would like to thank Prof. Dr. Béla Vizvári for his passion for the science, and his continuous support of this study. Without his invaluable supervision, all my efforts could have been short-sighted.

I am grateful to Assoc. Prof. Dr. Gökhan İzbirak, Chairman of the Industrial Engineering Department that helped me with various issues during my study.

I sincerely acknowledge Asst. Prof. Dr. Huseyin Güden for his advices, useful comments and his fresh innovative ideas and for all the mathematical and non-mathematical helps during my studying.

I obliged to appreciate Asst. Prof. Dr. Reza Vatankhah Barenji, for his constructive guidance and comments in publishing my articles. I believe that without his helps this project was impossible to do.

Finally, I would like to thank Prof. Gergely Kovács, all of my professors, colleagues, and friends who helped and supported me to finish this thesis.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

CNC	Computer Numerical Control
FMC	Flexible Manufacturing Cell
FMS	Flexible Manufacturing System
OBF	Objective Function
PPM	Proposed Mathematical Model
SA	Simulated Annealing
SAA	Simulated Annealing Algorithm
SPPM	Sequential Part Production Matrix
TRM	Time Reduction Method

Chapter 1

INTRODUCTION

1.1 Preface

Flexible manufacturing cells (FMCs) are used to produce standardized items at a high production speed and are used in reconfigurable manufacturing systems [1]. These cells are workplaces that contain a number of CNC machines. These CNC machines are usually linked together to produce some part types and are controlled by an automated control system [2]. The materials are handled by a robot between the machines. In parallel-machine cells, each machine is capable of performing all processes of producing a part [3, 4]. In general, the processing time of a part is different for different machines. Also, processing of different parts results in different processing times. In the latter case, it is supposed that each machine produces only one type of product. In group technology, each machine always processes the same type of parts, in which machines are assigned to part families.

It is assumed that distances between the input buffer and the first machine, between two successive machines and between the last machine and the output buffer are same. When an item is processed by any of the machines it becomes a finished item and goes to the output buffer. The robot moves through the line and performs the loading/unloading activities and transports the items. A system with *m* machines is illustrated in Figure 1.

It is assumed that the considered system repeats a cycle in its long run. If the system is at a specific state at the beginning of a cycle, it reaches the same state at the end of the cycle, and then repeats the same actions in the same order in the subsequent cycles. The duration of a cycle is called cycle time. It is assumed that each machine processes one part in each cycle. Decreasing the cycle time in such a system means increasing the production rate. The cycle time depends on the order of the actions. In such a system, determining the order of the actions to minimize the cycle time or to maximize the production rate is called an optimization problem.

Gultekin et al. [5] presented a mathematical model for the problem and expressed that the problem is NP-hard. In this thesis, we present a simulated annealing based metaheuristic (SA) algorithm, to solve the larger problems. When some metaheuristics are desired to be developed for solving the problem it is noticed that even the order of the activities are known it is not trivial to compute the minimum cycle time. For a given solution, i.e. order of the activities, in order to compute the minimum cycle time a linear programming model is needed to be solved. To the best of our knowledge, there is no such a study in the literature that to compute the objective function of a given solution a linear programming model is needed to be solved. The reason for such a need is explained in the following sections. Finally, we analyze the performances of the proposed metaheuristics using several numerical instances.

1.2 FMC different layouts

Regardless of having either a flow shop or parallel manufacturing system, there are two well-known layouts for the FMCs. The linear robotic cells are such that the input buffer (for the unprocessed parts), the machines of the cell and the output buffer (for the final product) are in a line [6] , for instance, from left to right (see Figure 1).

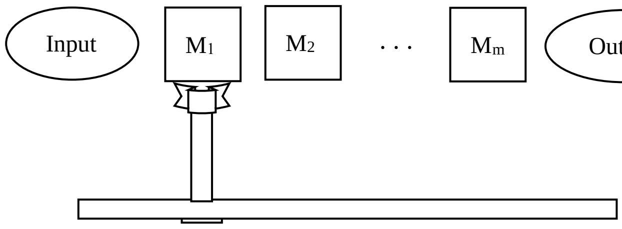

Figure 1. In-line m-machine robotic cell.

The circular robotic cells are such that the input buffer, machines and output buffer are arranged either clockwise or counter clockwise of a circle. It should be mentioned that there are two types of circular FMC. In the first type, there is only one buffer in the cell including the input and output buffers in the same place as you can see in Figure 2 [7, 8].

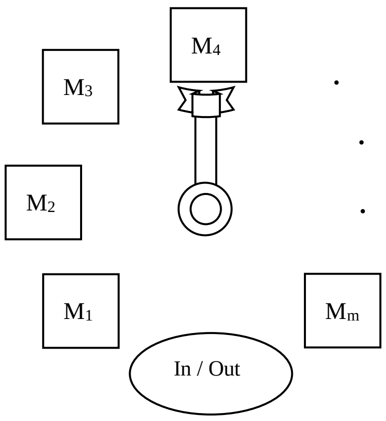

Figure 2. A one-buffer robot centered cell with three machines.

The second type is such that the input and output buffers are in different places [4, 9]. Obviously, all of the distance matrix, calculations and formulas will be the same as in-line FMC (see Figure 3) if the robot cannot move directly from input buffer to machine m by passing the output buffer or vice versa.

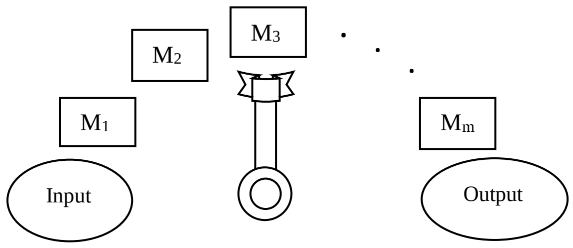

Figure 3. A circular robotic cell with different input and output buffers.

1.3 CNC machines with individual buffers

Although there are a considerable number of studies dealing with a robotic cell in the literature, there are very few studies that use a flexible cell with the machines that have individual buffers. From the practical point of view, using these buffers offers an attractive prospect to increase the production efficiency. At the same time, the increase in the combinatorial possibilities associated with the buffers severely complicates their theoretical analysis because each machine can be consecutively loaded twice using such buffers. In this study, we focus on the in-line layout (see Figure 4).

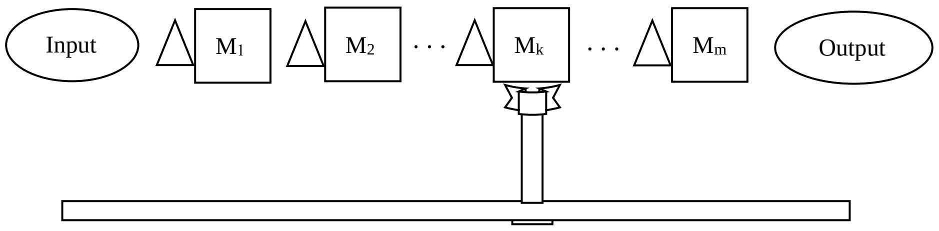

Figure 4. A robotic cell having machines with individual buffer.

1.4 The Structure of the Thesis

This thesis is organized into six chapters as follows: The first chapter lays out the structure and content of the entire thesis. The second chapter contains the related literature about the studies of previous researchers and related works in this field.

Some available literature in the areas of using tool magazines in general and specifically in the robotic manufacturing cells with CNC machines are reviewed, and an appropriated area for using heuristic methods to solve such problems is prepared.

In the third chapter, a line layout FMC without individual buffers on each machine is considered. To determine the best optimal solution, sequential part production matrix is presented and the case of an FMC with two CNC machines is thoroughly discussed. The problem is defined and the mathematical formulations for minimizing cycle time and maximizing the minimum return time are presented. Additionally, the numerical results of the proposed models and an improved lower bound for the general case and a lower bound explained by an assignment problem are provided. Furthermore, optimal cycles that are computed by using the return time of a machine for some different structures are presented.

In the fourth chapter, a line layout FMC including individual buffers for each machine is considered. In this chapter, the problem is defined and a mathematical formulation to minimize the cycle time is presented. Additionally, the numerical results of the proposed model, an improved lower bound for the general case and a lower bound explained by an assignment problem are provided. Furthermore, optimal cycles that are computed by using the return time of a machine for some different structures are presented.

Fifth chapter contains three metaheuristic algorithms based on local search algorithm for solving large size problems defined in chapter three. In the metaheuristics, in order to compute the minimum cycle time of a given solution a linear programming model is needed to be solved which is the first case in the literature to the best of our knowledge. Several numerical examples are solved by the proposed algorithms and their performance and solutions are compared.

Finally, chapter six contains discussion and conclusion of the study and is provided some ideas about further researches.

Chapter 2

LITERATURE REVIEW

2.1 Preface

One of the first studies related to the sequencing of parts in a robotic cell was conducted by Sethi et al. [9]. The objective of this study was to maximize the long-run average throughput of the system. Assuming that one part is produced by each machine in a cycle, they developed the cycle-time formulas for robot-centered cells with two and three machines. Crama and Van de Klundert [6] presented a dynamic programming approach for finding a shortest cyclic schedule for the robot movements that can be infinitely repeated. Furthermore, they proved that the minimum long-run average cycle time can be achieved by a one-unit cycle for a three-machine cell [10]. Both of their studies were related to identical parts and inline robotic cells. Hall et al. [11] studied a variety of classical scheduling objectives and provided either a polynomial- or a pseudo polynomial-time algorithm. Brauner and Finke [12] discussed the dominant states of a cell with identical parts and developed an algebraic approach for an m-machine cell; they also proved that the one-unit cycle is suitable for the two- and three-machine cells and showed that it is not optimal for four and more machines. Abdekhodaee et al. [3] performed two operations per cycle on each machine with non-preemptable jobs for a flexible manufacturing cell with parallel machines, and they also considered equal processing times and equal setup times.

Akturk et al. [13] studied the scheduling of a two-machine cell with identical parts, which comprise a number of operations to be completed in these two machines. They found the optimal robot movement cycle and the assignment of operations to these two machines as the objective function with minimizing the cycle time. Gultekin et al. [14] considered an inline robotic cell with two or three CNC machines and presented lower and upper bounds for one- and two-unit robot movement cycles, respectively. They proved that their proposed cycle dominates all two-unit robot movement cycles and also presented the regions where the proposed cycle dominates all one-unit cycles. Gultekin et al. [15] proposed a new cycle for a two-machine cell and proved that it dominates all classical robot movement cycles considered in the literature. They proved that changing the layout from an inline robotic cell to a robot-centered cell reduces the cycle time for the m-machine cell and found the optimal number of machines that minimize the cycle time of the proposed cycle. In another study, Gultekin et al. [16] worked on a flexible manufacturing cell with two or three machines to minimize the manufacturing cost and the cycle time jointly. They considered a one-unit cycle and determined the efficient set of processing time vectors so that no other processing time vector provides both a smaller cycle time and a lower cost. They also compared these cycles with each other for determining the sufficient conditions in which each of the cycles dominates the rest. Gultekin et al. [5] also modeled the problem of determining the best pure cycle for an m-machine cell with multiple parts as a special traveling salesman problem for parallel machines. They proposed a mathematical model and a two-stage heuristic algorithm for solving such problems. Ghadiri Nejad *et al.* [17] suggest an MTZ based TSP model for scheduling problem of the flexible robotic cell with m machines and a robot. They provide a reduced version for their model by excluding waiting time variables and reported that, the reduced model is much efficient in comparison with the models reported in the literature. In another study, Ghadiri Nejad *et al.* developed a mathematical model for the scheduling problem of *m* machines and a robot FRC using “Network Flow” approach [18]. Akturk et al. [19] considered a machine-job assignment problem with controllable processing times and modeled it as a nonlinear mixed 0-1 profit maximization problem. They also reformulated the problem using a polynomial for a number of conic constraints. Yildiz et al. [20] differentiated two pure cycles and showed that these two cycles together dominate the rest of the pure cycles for a wide range of processing times. They established the worst case and showed that the objective function is the minimization of the cycle time. Uruk et al. [21] considered a two-machine flow shop scheduling problem with identical jobs and determined the assignment of flexible operations to the machines and processing time for each operation to minimize the cycle time.

2.2 CNC machines having tool magazine

There are some studies that considered a flexible manufacturing cell including CNC machines with tools for the CNC machines and also gripper for the robots to empower them to be more flexible. Some of the important studies are as follows.

Dawande *et al.* [4] focused on a survey related to previous studies on robotic cell scheduling problems. They also tried out to find a lower bound for a one-unit cycle time of an FMC with multiple robots containing single and dual grippers. Drobouchevitch et al. [8] tried out to find the optimal sequence of robot movements for maximizing the long-run average throughput rate of the cell. They worked on a special FMC containing one buffer as an input/output buffer in a robot-centered layout. They also considered a dual-gripper cell and a single-gripper cell with machine output buffers of one-unit capacity. As an important developing step in FMS discussions, Zeballos [22] presented a constraints programming methodology to deal with the scheduling of FMS consisting of search strategy and handled several features found in the industrial environment such as limitations on number of tool system, tool lifetime and tool magazine capacity of machines. Foumani and Jenab [23] studied one-unit cycles for an inline robotic cell and found the robot movement sequence that minimizes the cycle time. They also presented the regions of optimality when each part reenters the first machine twice and determined optimality conditions for different cycles when each part reenters both machines twice. After sensitivity analysis of both cases, they found the best and the worst cycle mathematically. Furthermore, these two researchers worked on m-unit pure cycles to find the robot movement sequence for minimizing the cycle time when the robot had the swapping ability [24]. They presented an improved pure cycle that always dominates pure cycles and introduced a lower bound. Jolai *et al* [25] studied a robotic cell scheduling problem with identical part types, when machines are flexible and able to swap. They determined all 1-unit cycle times and proposed a novel cycle for robot movements that dominates all robot move cycles in the literature.

2.3 Heuristic methods

De Giovani and Pezzella [26] proposed an improved genetic algorithm to solve the distributed and flexible job-shop scheduling problem and considered a flexible manufacturing unit including four separate FMC interconnected by a material handling system. Batur et al. [27] suggested the robot movement sequence as well as the processing times of the parts on each machine which jointly minimized the cycle time for a two-machine manufacturing cell which repeatedly produces a set of multiple part-types. They also constructed an efficient 2-stage heuristic algorithm and compared it to the most common heuristic approach in scheduling for longest processing time. Kim et al. [28] examined the cyclic scheduling problem for a dual-armed cluster tool that performs periodic cleaning processes. They identified sufficient conditions for which the conventional backward and swap sequences provide the minimum cycle time. They also proposed two heuristic scheduling strategies and compared them with the conventional scheduling methods and the lower bound of each schedule.

Chapter 3

THE FMC WITHOUT INDIVIDUAL BUFFER FOR EACH MACHINE

3.1 Preface

In this chapter, we consider *m* parallel and identical CNC machines placed on a line. Different parts with different processes can be performed by each machine and consequently the process time of each machine can be different. There are an input station in which the items to be processed are kept and an output station in which the finished items are kept. When an item is processed by any of the machines, it becomes a finished item and it must be taken to the output station. There is a robot that performs the loading/unloading activities and transports the items.

Some basic definitions, assumptions, parameters and sets are used in this thesis are as follows:

Definition 1. A loaded robot movement is when the robot takes a part from input buffer and moves to load a machine or takes a finished part from a machine, moves to output buffer to unload it.

Definition 2. An unloaded robot movement is moving the robot without a part.

The parameters and sets are used in this study are as follows; It is supposed that all data of the problems are integers.

ε: The time of the loading and unloading of machines and buffers. This time for all the machines, input and output buffers is the same and constant.

δ: The robot travel times which are the same between input buffer and the first machine, between every two consecutive machines and between the last machine and output buffer.

M_i: Machine i where i is the machine number, $i = 1, 2, ..., m$.

S: Set of all the possible different order for which m parts can be produced using m machines in each cycle.

p_i: The processing time of a part on machine i is p_i.

L_i: The loading of machine i, that include the robot movement from its position to the input buffer, taking a part, moving to machine i and loading machine i.

U_i: The unloading of the machine i by the robot. U_i includes taking a part from machine i, moving to the output buffer and putting the part in the output buffer.

L: The set of all the loading activities, $L = \{L_1, L_2, \ldots, L_m\}$.

U: The set of all the unloading activities, i.e. $U = \{U_1, U_2, \ldots, U_m\}$.

A: The set of all the activities which belong to the sets of L and U. Hence, $A = \{L_1, L_2, \ldots, L_m, U_1, U_2, \ldots, U_m\}$.

T: Cycle time, *i.e.* the time of loading, processing and unloading of all the parts of a cycle and returning to the initial state including waiting time of the robot.

w_i: Waiting time of the robot on machine *i* to unload the part that may occur if the robot has to wait on the machine until it finishes the process. Obviously the robot does not wait in the input or output buffer at all. If the time required after loading a machine until the robot returns to that machine to unload it is less than the processing time on the machine, then, waiting time is positive, otherwise it is zero.

Definition 3: When L_i is completed, the robot stays at machine *i*. Similarly, when U_i is completed, the robot stays at the output buffer.

Definition 4: When a machine processes a part, it is full and the machine cannot be loaded by a new part unless the robot unloads the machine first.

Definition 5: As there are *m* machines in the general case, in each cycle *m* parts are produced, since each part is completely produced by only one machine.

Definition 6: Every loading or unloading task is considered as an activity.

3.2 Sequential Part Production Matrix

To determine the best optimal solution, we must consider all of the different pure cycles of the system and find their cycle times. To find all of the possible pure cycles of a system we present sequential part production matrix (SPPM) as follows:

SPPM is a matrix that contains two columns. The first column is related to the loading activity and the second column is related to the unloading activity. Every row is related to one product and clearly, there are *m* rows as *m* products for an *m*-machine *FMC*. The sequence of the cycle can be distinguished by the numbers in the matrix. For example if *m = 2* and we want to write the matrix related to $L_1L_2U_1U_2$ cycle (which is illustrated in Figure 5) there will be a 2 by 2 matrix as there are two products in each cycle. Also because the first activity is loading machine 1, the matrix will be $\begin{bmatrix} 1 & - \\ - & - \end{bmatrix}$. In the next steps, machine 2 is loaded. Therefore, the matrix will change to $\begin{bmatrix} 1 & - \\ 2 & - \end{bmatrix}$ because as earlier mentioned the second row is related to the second part. To complete the cycle, machine 1 and machine 2 will be unloaded respectively. Thus the matrix will be completed as $\begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix}$.

In general, let $A_1, A_2, \ldots, A_{2m}$ be the sequence of activities in the cycle. Notice that $A_1 = L_1$ and the set formed from the elements of the sequence, i.e. $\{ A_1, A_2, \ldots, A_{2m} \}$ is the set A. The elements of the *SPPM* matrix are denoted by s_{ik}, where $i = 1,2, \ldots, m$ and $k = 1,2$. The value of the elements s_{i1} is l if $A_l = L_i$ and similarly, the value of s_{i2} is l if $A_l = U_i$.

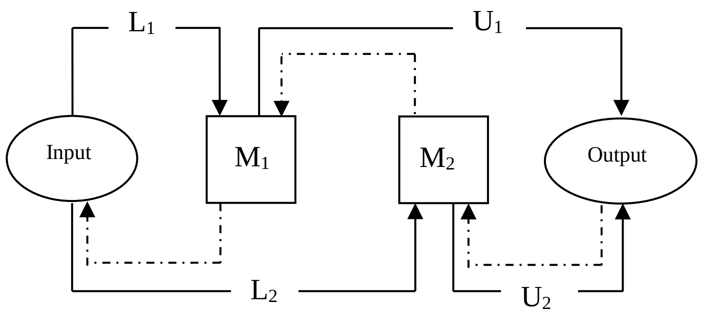

Figure 5. Robot movement sequence related to $L_1L_2U_1U_2$ cycle.

To write all of the pure cycles for an *m*-machine *FMC*, Gultekin et al. [15] proved that from *(2m)!* possible pure cycles, *(2m −1)!* different pure cycles exist in such a cell. To neglect repetitive permutation, like $L_1L_2U_1U_2$ and $U_1U_2L_1L_2$ that are the different representations of the same cycle, they assumed that all cycles will be started with activity L_1. In this way there are six different pure cycles in a 2-machine *FMC*. Let $S = \{1, 2, 3, 4, 5 \text{ and } 6\}$ be the set of all the possible strategies for processing two parts with two machines, the *SPPM* will be the following:

Table 1. SPPM for producing two parts in a two-machine cell

Strategy name	1	2	3	4	5	6
Activities Sequence	$L_1L_2U_1U_2$	$L_1L_2U_2U_1$	$L_1U_1L_2U_2$	$L_1U_1U_2L_2$	$L_1U_2L_2U_1$	$L_1U_2U_1L_2$
SPPM	$\begin{bmatrix}1 & 3\\2 & 4\end{bmatrix}$	$\begin{bmatrix}1 & 4\\2 & 3\end{bmatrix}$	$\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix}$	$\begin{bmatrix}1 & 2\\4 & 3\end{bmatrix}$	$\begin{bmatrix}1 & 4\\3 & 2\end{bmatrix}$	$\begin{bmatrix}1 & 3\\4 & 2\end{bmatrix}$

3.3 The case of m = 2

In this section, a robotic *FMC* with two machines is considered.

3.3.1 Cycle times

To determine the best optimal solution, minimal cycle time, we should consider all of the different pure cycles of the system and find their cycle times. There are six different pure cycles for an *FMC* with two machines which are shown in Table 2. For example cycle $L_1L_2U_1U_2$ shows that the robot starts from machine 1, goes to the input buffer, takes a part and loads machine 2. Then the robot goes to machine 1 and unloads it and puts the part in the output buffer. After that it goes to machine 2 and unloads it and puts the part in the output buffer and finally it goes to the input buffer, takes a part and loads machine 1.

Table 2. Sequences of the activities for producing two parts in a two-machine cell.

Case number	1	2	3	4	5	6
Activities Sequence	$L_1L_2U_1U_2$	$L_1L_2U_2U_1$	$L_1U_1L_2U_2$	$L_1U_1U_2L_2$	$L_1U_2L_2U_1$	$L_1U_2U_1L_2$

Theorem 1: The cycle times of all the robot movement cases for a *2*-machine cell with non-identical parts are as follows:

$T_1 = 8\delta + 6\varepsilon + max\{4\delta + 2\varepsilon, p_1, p_2\}$

$T_2 = 6\delta + 4\varepsilon + max\{6\delta + 4\varepsilon + p_2, p_1\}$

$T_3 = 12\delta + 8\varepsilon + p_1 + p_2$

$T_4 = 6\delta + 4\varepsilon + max\{6\delta + 4\varepsilon + p_1, p_2\}$

$T_5 = 6\delta + 4\varepsilon + max\{8\delta + 4\varepsilon, p_1, p_2\}$

$T_6 = 10\delta + 6\varepsilon + max\{4\delta + 2\varepsilon, p_1, p_2\}$

Proof: The cycle times calculations for all of the cases are quite similar. Therefore, we limit ourselves to illustrate the method through providing the proof for only one case say T_1. In order to provide a better understanding for the calculations, the duration of each step is expressed at the end of it. Hence, the cycle time can be calculated by getting the summation of these durations. The steps for cycle time calculation for case 1 (T_1), starting from machine 1, when it is loaded (L_1) is as follows:

1- The robot, moves from machine 1, to the input buffer, picks up a part, moves to machine 2, and loads it. ($\delta + \varepsilon + 2\delta + \varepsilon$)

2- The robot, moves to machine 1, waits on machine 1 if it is necessary until its process is finished, picks the part up, moves to the output buffer and unloads it. ($\delta + w_1 + \varepsilon + 2\delta + \varepsilon$)

3- The robot returns to machine 2, waits on machine 2, if it is necessary until the process is finished, picks up the part, moves to the output buffer and unloads it. ($\delta + w_2 + \varepsilon + \delta + \varepsilon$)

4- The robot returns to the input buffer. (3δ)

5- The robot picks up a part from the input buffer, moves to machine 1 and loads it. $(\varepsilon + \delta + \varepsilon)$

The cycle time using this case is:

$T_1 = (\delta + \varepsilon + 2\delta + \varepsilon) + (\delta + w_1 + \varepsilon + 2\delta + \varepsilon) + (\delta + w_2 + \varepsilon + \delta + \varepsilon) + (3\delta) + (\varepsilon + \delta + \varepsilon) = 12\delta + 8\varepsilon + w_1 + w_2;$ where $w_1 = max\{0, p_1 - (4\delta + 2\varepsilon)\}$ and $w_2 = max\{0, p_2 - (4\delta + 2\varepsilon + w_1)\}$.

As it can be seen, the cycle time is dependent on w_1 and w_2, therefore it is necessary to discuss the different cases of them. For T_1, the unknown waiting times are w_1 and w_2 and their values may vary depending on the values of p_1, p_2 and $4\delta + 2\varepsilon$.

If $p_1 \leq 4\delta + 2\varepsilon$ and $p_2 \leq 4\delta + 2\varepsilon$ then $w_1 = w_2 = 0$, therefore, $T_1 = 12\delta + 8\varepsilon$.

If $p_1 > 4\delta + 2\varepsilon$ and $p_2 \leq p_1$ then $w_1 = p_1 - (4\delta + 2\varepsilon)$ and $w_2 = 0$, therefore, $T_1 = 8\delta + 6\varepsilon + p_1$.

If $p_2 > 4\delta + 2\varepsilon \geq p_1$ then $w_1 = 0$ and $w_2 = p_2 - (4\delta + 2\varepsilon)$, therefore, $T_1 = 8\delta + 6\varepsilon + p_2$.

If $p_2 > p_1 \geq 4\delta + 2\varepsilon$ then $w_1 = p_1 - (4\delta + 2\varepsilon)$ and $w_2 = p_2 - p_1$, therefore, $T_1 = 8\delta + 6\varepsilon + p_2$.

The calculations of T_1 can be summarized as $8\delta + 6\varepsilon + max\{4\delta + 2\varepsilon, p_1, p_2\}$. By using this method, all pure cycles' times can be proved as mentioned. □

3.3.2 The optimal cycle time as a function of the processing time

To find a lower bound for the cycle time, all of the cycle times which are conditional should be separated. For example T_1 can be separated to $T_{11} = 12\delta + 8\varepsilon$, if $p_1 \leq$

$4\delta + 2\varepsilon$ and $p_2 \leq 4\delta + 2\varepsilon$, $T_{12} = 8\delta + 6\varepsilon + p_1$, if $p_1 > 4\delta + 2\varepsilon$ and $p_1 > p_2$ and $T_{13} = 8\delta + 6\varepsilon + p_2$, if $p_2 > 4\delta + 2\varepsilon$ and $p_2 > p_1$. By using these separated formulas, there will be 14 different cases as follows that should be compared together:

$T_{11} = 12\delta + 8\varepsilon$, if $p_1 \leq 4\delta + 2\varepsilon$ and $p_2 \leq 4\delta + 2\varepsilon$

$T_{12} = 8\delta + 6\varepsilon + p_1$, if $p_1 > 4\delta + 2\varepsilon$ and $p_1 \geq p_2$

$T_{13} = 8\delta + 6\varepsilon + p_2$, if $p_2 > 4\delta + 2\varepsilon$ and $p_2 > p_1$

$T_{21} = 12\delta + 8\varepsilon + p_2$, if $p_1 \leq 6\delta + 4\varepsilon + p_2$

$T_{22} = 6\delta + 4\varepsilon + p_1$, if $p_1 > 6\delta + 4\varepsilon + p_2$

$T_3 = 12\delta + 8\varepsilon + p_1 + p_2$

$T_{41} = 12\delta + 8\varepsilon + p_1$, if $p_2 \leq 6\delta + 4\varepsilon + p_1$

$T_{42} = 6\delta + 4\varepsilon + p_2$, if $p_2 > 6\delta + 4\varepsilon + p_1$

$T_{51} = 14\delta + 8\varepsilon$, if $p_1 \leq 8\delta + 4\varepsilon$ and $p_2 \leq 8\delta + 4\varepsilon$

$T_{52} = 6\delta + 4\varepsilon + p_1$, if $p_1 > 8\delta + 4\varepsilon$ and $p_1 \geq p_2$

$T_{53} = 6\delta + 4\varepsilon + p_2$, if $p_2 > 8\delta + 4\varepsilon$ and $p_2 \geq p_1$

$T_{61} = 14\delta + 8\varepsilon$, if $p_1 \leq 8\delta + 4\varepsilon$ and $p_2 \leq 8\delta + 4\varepsilon$

$T_{62} = 10\delta + 6\varepsilon + p_1$, if $p_1 > 4\delta + 2\varepsilon$ and $p_1 > p_2$

$T_{63} = 10\delta + 6\varepsilon + p_2$, if $p_2 > 4\delta + 2\varepsilon$ and $p_2 > p_1$

To find the lower bound in terms of the processing times, we consider the intervals that p_1 and p_2 belong to. According to the formulae, the intervals are as follows:

1) $p_1, p_2 \in [0, 4\delta + 2\varepsilon]$ that includes $T_{11}, T_{21}, T_3, T_{41}, T_{51}$ and T_{61}.
2) $p_1 \in (4\delta + 2\varepsilon, 8\delta + 4\varepsilon]$ and $p_1 \geq p_2$ that includes $T_{12}, T_{21}, T_{22}, T_3, T_{41}, T_{51}$ and T_{62}.

3) $p_2 \in (4\delta + 2\varepsilon, 8\delta + 4\varepsilon]$ and $p_2 > p_1$ that includes $T_{13}, T_{21}, T_3, T_{41}, T_{42}, T_{51}$ and T_{63}.

4) $p_1 \in (8\delta + 4\varepsilon, \infty)$ and $p_1 \geq p_2$ that includes $T_{12}, T_{21}, T_{22}, T_3, T_{41}, T_{52}$ and T_{62}.

5) $p_2 \in (8\delta + 4\varepsilon, \infty)$ and $p_2 > p_1$ that includes $T_{13}, T_{21}, T_3, T_{41}, T_{42}, T_{53}$ and T_{63}.

In the first interval, when both p_1 and p_2 are less than or equal to $4\delta + 2\varepsilon$, T_{11} is the least cycle time. Because T_{21} and T_{41} have a p_2 and a p_1 more than T_{11}, respectively, T_3 has $p_1 + p_2$ more than T_{11} and also T_{51} and T_{61} have 2δ more than T_{11}. By using such comparisons for intervals of states 2 to 5, the following lemma is true.

Lemma 1: If T_{min} denotes the minimal cycle time, then

$$T_{min} = \begin{cases} T_{11} = 12\delta + 8\varepsilon & \text{if} \quad p_1 \leq 4\delta + 2\varepsilon \text{ and } p_2 \leq 4\delta + 2\varepsilon \\ T_{12} = 8\delta + 6\varepsilon + p_1 & \text{if } p_1 \geq p_2 \text{ and } 4\delta + 2\varepsilon < p_1 \leq 6\delta + 2\varepsilon \\ T_{51} = 14\delta + 8\varepsilon & \text{if } p_1 \geq p_2 \text{ and } 6\delta + 2\varepsilon < p_1 \leq 8\delta + 2\varepsilon \\ T_{13} = 8\delta + 6\varepsilon + p_2 & \text{if } p_1 < p_2 \text{ and } 4\delta + 2\varepsilon < p_2 \leq 6\delta + 2\varepsilon \\ T_{51} = 14\delta + 8\varepsilon & \text{if } p_1 < p_2 \text{ and } 6\delta + 2\varepsilon < p_1 \leq 8\delta + 2\varepsilon \\ T_{52} = 6\delta + 4\varepsilon + p_1 & \text{if} \quad p_1 \geq p_2 \text{ and } 8\delta + 4\varepsilon < p_1 \\ T_{53} = 6\delta + 4\varepsilon + p_2 & \text{if} \quad p_2 > p_1 \text{ and } 8\delta + 4\varepsilon < p_2 \end{cases}$$

□

3.3.3 An upper bound for cycle time

Considering the cycle time formula for different sequences of the robot movements and comparing them with each other, all five intervals which are mentioned in section 3.3.2 must be considered. In the first interval in which p_1 and p_2 are less than $4\delta + 2\varepsilon$, T_3 is greater than T_{11}, T_{21} and T_4. On the other hand, comparisons between T_3, $T_{51} = T_{61}$ show that if $p_1 + p_2 \leq 2\delta$ then $T_{51} = T_{61}$ is the upper bound for the cycle time, otherwise T_3 will be the upper bound. Comparing all the different cycle times in the second and third intervals, it is easy to see that T_3 is always the greatest cycle time among the others. The upper bound of the cycle time for a 2-machine cell with non-identical parts has been summarized as follows:

$$T_{max} = \begin{cases} 14\delta + 8\varepsilon & \text{if } p_1 + p_2 \leq 2\delta \\ 12\delta + 8\varepsilon + p_1 + p_2 & \text{otherwise} \end{cases}$$

3.4 Definitions, parameters and sets

The rest of the parameters and sets are used to present the proposed mathematical models are as follows:

d_{jk}: The time to complete activity k just after completing activity j.

To calculate d_{jk}, the position of the robot after performing each activity must be taken into consideration. After a loading activity, the position of the robot is at the machine which was loaded, while after an unloading activity, this position is at the output buffer. Therefore, if activity k is an unloading activity (U_k), the time required for the robot to pick a part up and unload it into the output buffer includes w_k because perhaps when the robot reaches machine k to pick the k^{th} part up, the process of that part has not finished yet. Specially, if $j = L_i$ and $k = U_i$ in the previous step, the robot loads part i on machine i and immediately it must unload the same part, therefore the robot's waiting time on machine i will be equal to the processing time p_i. For example, if we want to calculate the time related to $d_{L_1U_m}$, after loading machine 1, the robot is at machine 1. Therefore, it must go from machine 1 to machine m by using $(m-1)\delta$ time units to unload it. In this case if the machine m has finished the process of the part, there will be no waiting time, otherwise, the robot must wait on machine m that is equal to w_m time units. Then the robot must get a part (ε), move to the output buffer (δ) and unload the part m. Thus, the total time for $d_{L_1U_m}$ is equal to $m\delta + 2\varepsilon + w_m$. On the other hand, if activity k is related to a loading activity, regardless of what activity j is (loading or unloading), there will be no waiting time and the time distance will be calculated easily. In Table 3, the time distance matrix for performing every two consecutive activities in an *m*-machine flexible manufacturing cell by the robot can be seen.

Table 3. Time distance Matrix for *m*-machine FMC.

j \ k	L1	L2	…	Lm-1	Lm	U1	U2	…	Um-1	Um
L1	-	$3\delta+2\varepsilon$	…	$m\delta+2\varepsilon$	$(m+1)\delta+2\varepsilon$	$m\delta+2\varepsilon+p_1$	$m\delta+2\varepsilon$ $+w_2$	…	$m\delta+2\varepsilon$ $+w_{m-1}$	$m\delta+2\varepsilon$ $+w_m$
L2	$3\delta+2\varepsilon$	-	…	$(m+1)\delta+2\varepsilon$	$(m+2)\delta+2\varepsilon$	$(m+1)\delta$ $+2\varepsilon+w_1$	$(m-1)\delta$ $+2\varepsilon+p_2$	…	$(m-1)\delta$ $+2\varepsilon+w_{m-1}$	$(m-1)\delta$ $+2\varepsilon+w_m$
.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.
Lm-1	$m\delta+2\varepsilon$	$(m+1)\delta+2\varepsilon$	…	-	$(2m-1)\delta+2\varepsilon$	$(2m-2)\delta$ $+2\varepsilon+w_1$	$(2m-4)\delta$ $+2\varepsilon+w_2$	…	$2\delta+2\varepsilon+p_{m-1}$	$2\delta+2\varepsilon$ $+w_m$
Lm	$(m+1)\delta+2\varepsilon$	$(m+2)\delta+2\varepsilon$	…	$(2m-1)\delta+2\varepsilon$	-	$(2m-1)\delta$ $+2\varepsilon+w_1$	$(2m-3)\delta$ $+2\varepsilon+w_2$	…	$3\delta+2\varepsilon$ $+w_{m-1}$	$\delta+2\varepsilon+p_m$
U1	$(m+2)\delta+2\varepsilon$	$(m+3)\delta+2\varepsilon$	…	$(2m)\delta+2\varepsilon$	$(2m+1)\delta+2\varepsilon$	-	$(2m-2)\delta$ $+2\varepsilon+w_2$	…	$4\delta+2\varepsilon$ $+w_{m-1}$	$2\delta+2\varepsilon$ $+w_m$
U2	$(m+2)\delta+2\varepsilon$	$(m+3)\delta+2\varepsilon$	…	$(2m)\delta+2\varepsilon$	$(2m+1)\delta+2\varepsilon$	$(2m)\delta$ $+2\varepsilon+w_1$	-	…	$4\delta+2\varepsilon$ $+w_{m-1}$	$2\delta+2\varepsilon$ $+w_m$
.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.
Um-1	$(m+2)\delta+2\varepsilon$	$(m+3)\delta+2\varepsilon$	…	$(2m)\delta+2\varepsilon$	$(2m+1)\delta+2\varepsilon$	$(2m)\delta$ $+2\varepsilon+w_1$	$(2m-2)\delta$ $+2\varepsilon+w_2$	…	-	$2\delta+2\varepsilon$ $+w_m$
Um	$(m+2)\delta+2\varepsilon$	$(m+3)\delta+2\varepsilon$	…	$(2m)\delta+2\varepsilon$	$(2m+1)\delta+2\varepsilon$	$(2m)\delta$ $+2\varepsilon+w_1$	$(2m-2)\delta$ $+2\varepsilon+w_2$	…	$4\delta+2\varepsilon$ $+w_{m-1}$	-

Also in this study the following decision variables are used:

t_{Li}: The completion time of loading the part on machine *i*.

t_{Ui}: The completion time of unloading machine *i* and putting the part into the output buffer.

x_{jk}: A binary decision variable that will be 1 if activity *j* is followed by activity *k* where $j, k \in A$ and 0, otherwise.

z_k: A binary variable that is 1, when *L_k* precedes *U_k* and 0, otherwise. If at the beginning of the cycle machine k processes a part then *U_k* precedes *L_k*, otherwise, not.

3.5 Mathematical models

In this section, three different mathematical models are reported. The first model is called the complete model, which is an alternative to the model of Gultekin et al. [5]. The second model is the reduced model. In this model, all the waiting time variables and constraints that are related to the calculation of the waiting times are eliminated. The last mathematical model maximizes the minimum robot return time to each machine. This value is very important because the cycle time and processing times are connected to each other and the feasibility of the cycle time can be verified by this model.

3.5.1 Complete mathematical model

The complete mathematical model that explicitly contains the waiting time variables is described as follows:

Minimize T (1)

$$t_{L_k} \geq t_{L_j} + d_{L_jL_k}x_{L_jL_k} - \left(1 - x_{L_jL_k}\right)M, \ \ \forall\, j,k = 1,\ldots,m, j \neq k, k \neq 1 \tag{2}$$

$$t_{L_k} \geq t_{U_j} + d_{U_jL_k}x_{U_jL_k} - \left(1 - x_{U_jL_k}\right)M, \ \ \forall\, j,k = 1,\ldots,m, k \neq 1 \tag{3}$$

$$t_{U_k} \geq t_{L_k} + \ d_{L_kU_k} - (1 - z_k)M, \ \ \forall\, k = 1,\ldots,m \tag{4}$$

$$t_{U_k} \geq (t_{L_k} - T) + d_{L_kU_k} - Mz_k, \ \ \forall\, k = 1,\ldots,m \tag{5}$$

$$t_{U_k} \leq t_{L_k} + Mz_k - 1, \ \ \forall\, k = 1,\ldots,m \tag{6}$$

$$t_{U_k} \geq t_l + d_{lU_k}x_{lU_k} + w_k - \left(1 - x_{lU_k}\right)M, \ \ \forall\, l \in \{L_j, U_j\}, \ j,k = 1,\ldots,m, j \neq k \tag{7}$$

$$t_{U_k} \leq t_l + d_{lU_k}x_{lU_k} + w_k + \left(1 - x_{lU_k}\right)M, \ \ \forall\, l \in \{L_j, U_j\}, \ j,k = 1,\ldots,m, j \neq k \tag{8}$$

$$t_{L_1} = 0 \tag{9}$$

$T \geq t_l + d_{lL_1} x_{lL_1}, \ \forall \, l \, \epsilon \{L_k, U_k\}, k = 1, \ldots, m, k \neq L_1$ (10)

$\sum_{q \epsilon A} x_{lq} = 1, \ \forall \, l \epsilon \, A \mid l \neq q$ (11)

$\sum_{l \epsilon A} x_{lq} = 1, \ \forall \, q \epsilon \, A \mid l \neq q$ (12)

$x_{lq} \in \{0,1\}, \forall l, q \in A; z_k \in \{0,1\}, w_k \geq 0, \forall k; \ t_j \geq 0, \ \forall \, j \in A; \ y_{lU_k} \geq 0, \forall l, k$ (13)

The objective function minimizes the cycle time which is shown by formula (1). Constraint (2) considers all cases that the robot consecutively loads on two different machines, where M is a large number which is at least as large as the cycle time. Constraint (3) is related to the cases in which an unloading activity is followed by a loading activity. Constraints (4)–(6) are related to the loading activities that are followed by an unloading activity. If the loading and unloading activities are related to the same machine, the mathematical constraint will be shown by formulae (4) and (5). Formula (5) describes the case in which the loading of the machine was performed in the previous cycle. Constraint (4) claims that if $\mathrm{z_k} = 1$, then $\mathrm{t_{U_k}} \geq \mathrm{t_{L_k}}$. The contrary must be claimed as well, i.e., if $\mathrm{z_k} = 0$, then $\mathrm{t_{L_k}} \geq \mathrm{t_{U_k}}$, which is shown by constraint (6). Formulae (7) and (8) determine the waiting times if an activity is followed by an unloading activity on a different machine. In these cases, the robot may have waiting times (wk), where $\mathrm{d_{lU_k}}$ is the element of Table 3 without $\mathrm{w_k}$. To prevent considering similar cycles such as $\mathrm{L_1L_2U_1U_2}$ and $\mathrm{U_2L_1L_2U_1}$, it is assumed that the cycle starts when machine 1 is loaded (L_1), which is shown by constraint (9).

To calculate the cycle time, the time when the robot moves after the last activity of the cycle to the input station and carries a part and loads machine 1, should be considered. Constraint (10) applies these calculations. Constraints (11) and (12) are the classical assignment constraints that are related to the sequence of the activities in a cycle. It is clear that when the robot performs an activity, in the next step only one of the other activities from the set A can be performed. Finally, Constraint (13) defines the decision variables.

3.5.2 The reduced mathematical model

To decrease the CPU time, the constraints and variables that are related to the waiting times can be excluded from the complete model. Therefore, instead of formulae (7) and (8), formula (14) can be added to the model. The new mathematical model that is called the *reduced model* is derived from formulae (1) to (6) and (9) to (14) collectively.

$$t_{U_k} \geq t_l + d_{lU_k}x_{lU_k} - \left(1 - x_{lU_k}\right)M,\ \forall\, l \in \left\{L_j, U_j\right\}, j, k = 1, \dots, m, j \neq k. \qquad (14)$$

3.5.3 Maximization of the minimum return time

The return time for machine k in a cycle is the time between loading the machine (L_k) to the moment when the robot returns to the same machine to unload it. The waiting time on machine k is zero if the return time is longer than the processing time.

The return time is $t_{U_k} - t_{L_k} - (m + 1 - k)\delta - 2\varepsilon$ if U_k is after L_k, i.e., $z_k = 1$. If U_k is before L_k, then the part that is loaded in this cycle will be unloaded in the next cycle, i.e., the unloaded activity will be completed at $t_{U_k} + T$. Thus, in this case the return time is $t_{U_k} + T - t_{L_k} - (m + 1 - k)\delta - 2\varepsilon$ in this case. Hence, the general formula for the return time is $t_{U_k} - t_{L_k} - (m + 1 - k)\delta - 2\varepsilon + T(1 - z_k)$. For a given cycle and the cycle time, the minimum return time is the maximum processing time such that the cycle can produce the cycle time. Thus, if the cycle time is fixed then the maximization of the minimum return time results in a cycle such that the cycle time still has a fixed value, but for any larger processing time the cycle time is also larger. This problem can be formulated as follows. The constraints from (2) to (13) are unchanged. The objective function is the maximization of the new variable, say h, such that:

$$t_{U_k} - t_{L_k} - (m + 1 - k)\delta - 2\varepsilon + T(1 - z_k) \geq h, \forall k \tag{15}$$

$$\text{Max}\, h \tag{16}$$

The problem must contain a fixed cycle time, i.e.,

$$\text{T = requested value} \tag{17}$$

To maximize the minimum return time, formulae (2)–(13) with formulae (15)–(17) must be considered where formula (16) is the objective function of that model.

3.6 Numerical results

3.6.1 The minimization of the cycle time

To compare the proposed models with the model of Gultekin et al. [5], an FMC for producing identical products with fixed process time, and fixed ε and δ, is considered. All results for solving the models were implemented for two to six machines using the CPLEX 12.6 software, and executed on an Intel(R) Pentium(R) Dual CPUE2180 @ 2.00 GHz CPU and 2.00 GB of RAM. Table 4 shows the objective functions and the CPU times for the models for the first scenario, considering ε and δ 1 and 2, respectively.

Table 4. CPU times of solving the proposed models and the Gultekin et al. model.

	3-machine cell				4-machine cell				5-machine cell			
P	Opt. cycle time	CPU time (seconds)			Opt. cycle time	CPU time (seconds)			Opt. cycle time	CPU time (seconds)		
		Gultekin et al. model	Proposed model	Reduced model		Gultekin et al. model	Proposed model	Reduced model		Gultekin et al. model	Proposed model	Reduced model
0	60	0.29	0.15	0.12	96	1.71	1.20	0.82	140	133.32	52.04	36.85
25	60	0.29	0.18	0.15	96	2.43	1.35	0.84	140	181.04	45.58	34.24
50	70	0.20	0.14	0.14	96	2.85	0.92	0.75	140	101.71	29.58	18.20
75	95	0.23	0.18	0.14	99	1.76	0.84	0.73	140	105.16	18.97	11.70
100	120	0.23	0.23	0.15	124	2.18	0.78	0.31	140	64.61	10.18	5.39
125	145	0.25	0.18	0.17	149	2.01	0.93	0.75	153	26.26	8.08	4.82
150	170	0.21	0.17	0.14	174	2.53	0.92	0.45	178	59.62	5.36	3.62
175	195	0.21	0.14	0.12	199	1.90	0.96	0.75	203	48.48	8.23	4.17
200	220	0.18	0.17	0.15	224	1.64	0.48	0.43	228	39.36	5.87	4.71
225	245	0.18	0.15	0.15	249	1.90	0.45	0.40	253	33.14	5.02	4.60
250	270	0.20	0.15	0.14	274	1.70	0.98	0.36	278	22.64	8.32	5.51

According to the results in Table 4, the proposed model has less CPU time than the model of Gultekin et al., and the reduced model has always the shortest CPU time among all these three models. The reduced model is the only model that can solve a six-machine cell in about 5 hours, which is about 2 hours faster than the performance of the complete model; and the model of Gultekin et al. could not be solved as the computer stopped the execution with an error message. Moreover, it is worth to state that the difference of the CPU times for these three models increases when the number of machines increases (see Figure 6, 7 and 8).

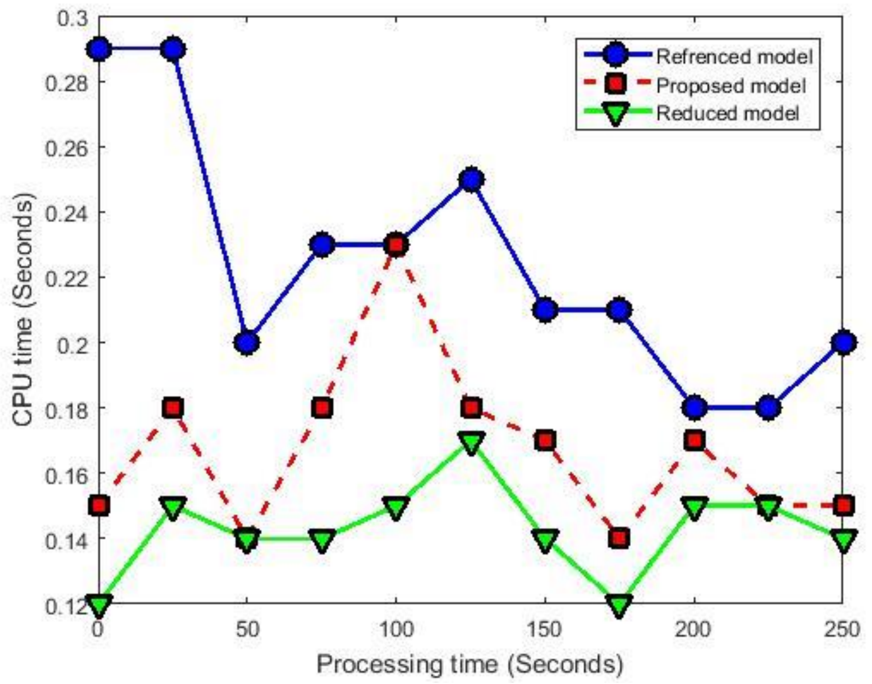

Figure 6. CPU times of the models for 3-machine test instances.

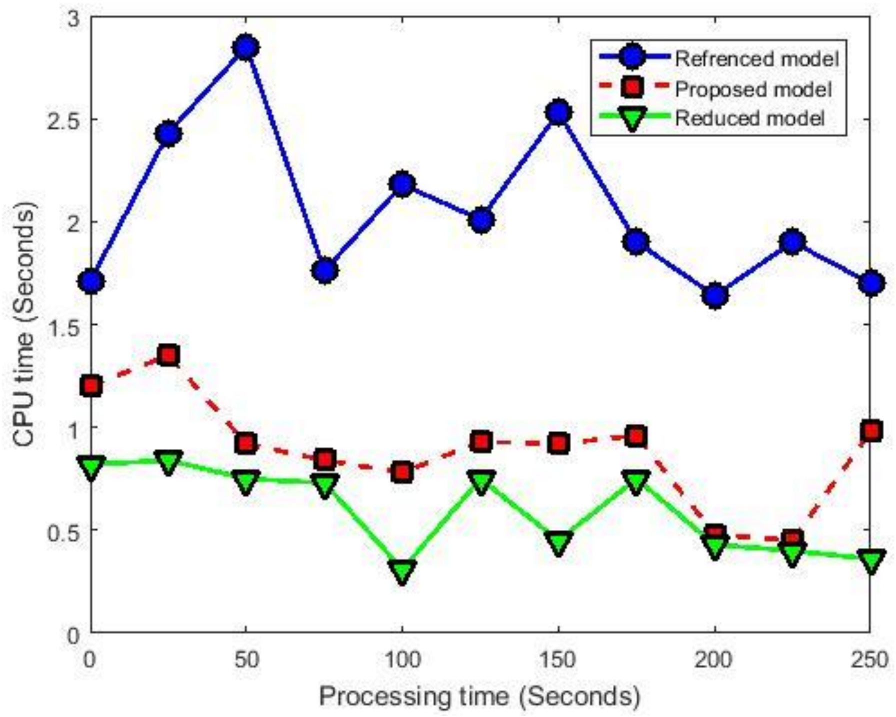

Figure 7. CPU times of the models for 4-machine test instances.

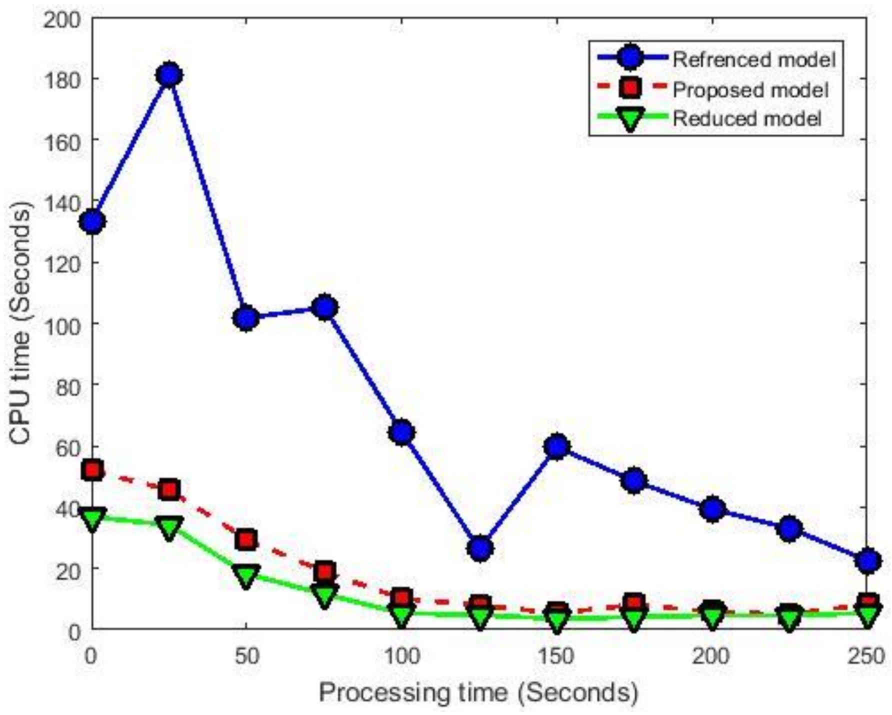

Figure 8. CPU times of the models for 5-machine test instances.

As depicted in the figures in general, the computation times of the models decreases or in some cases remains steady when the processing time of the machines increases. Additionally, the proposed universal and reduced mathematical models solve the problem in shorter times in compare to the model presented in the literature, where it is obvious that the performance of the reduced model is the best. It seems that, as process time in the machines increases solution times of the models converging (see Figure 8). In this testing scenario, the solution time of the models have completely similar trends.

Figure 9 shows the sequence of optimal robot movements for a four-machine cell obtained by the proposed mathematical model when $p = 22$, $\delta = 2$, and $\varepsilon = 1$ time unit. The numbers on arrows show the sequence of the robot movements.

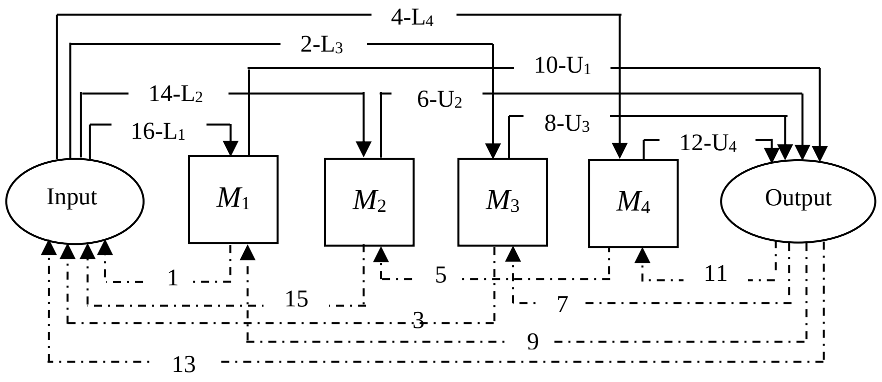

Figure 9. The robot optimal moves sequence for $L_1L_3L_4U_2U_3U_1U_4L_2L_1$ cycle.

3.6.2 Computational result of the return time

In this case, we considered a four-machine FMC with small processing times and 96 time units of the cycle time which is the minimum cycle time according to Theorem 3. This model was solved to find how large the processing times can be, and we found that the minimum return time is 66 in the cycle $\mathrm{L_1L_4U_3L_3U_2L_2U_1U_4}$. This means that if the processing time for each machine is less than or equal to 66, there will be no increase in the minimum cycle time and it is equal to 96. The minimum return time for the general case is discussed in Section 3.9.3. After this example, we tried to maximize the minimum return time for a 4-machine FMC when there is no limit for the cycle time; however, all the waiting times are zero. The solution shows that the return time for all of the machines is the same and equal to 84 when the cycle time is 106 for the cycle $\mathrm{L_1U_4L_4U_3L_3U_2L_2U_1}$.

The generalization of the optimal solution based on maximizing the minimum return time for a two- to six-machine cell shows that the optimal solution for an m-machine robotic cell will be $\mathrm{L_1U_mL_mU_{m-1}L_{m-1}} \ldots \mathrm{U_2L_2U_1}$.

3.7 An improved lower bound for the optimal cycle time in the general case

Gultekin et al. [5] proved two lower bounds using a single formula. The first lower bound is the time that a part stays in the system. The second lower bound is the minimum total moving time of the robot in a cycle which is independent of the processing times. They presented short and straightforward proofs for both the lower bounds. The new proof of the first lower bound is as follows:

Let's consider each part stays at the input station at first. The robot carries each part to a machine (M_j), the machine processes the part, and the robot carries it to the output station. This process consists of the following time elements. The robot is loaded at the input station (ε time units) as it moves to $M_j(j\delta)$, the robot is unloaded at $M_j(\varepsilon)$, the processing time is p_j, the robot is loaded at $M_j(\varepsilon)$ as it travels to the output station ($(m+1-j)\delta$), the part is unloaded at (ε), and finally to start the cycle again the robot must return to the input station ($(m+1)\delta$). The total time calculated is $4\varepsilon + 2(m+1)\delta + p_j$. Hence, the value of $4\varepsilon + 2(m+1)\delta + \max_{j=1 \text{ to } m} p_j$ is considered a lower bound of the cycle time. If the processing times are very long then they are the dominating factors for determining the cycle time. However, the value of $4\varepsilon + 2(m+1)\delta + \max_{j=1 \text{ to } m} p_j$ can be equal to the cycle time only if there is only a single machine or there is a long processing time that can complete other processes. In what follows, all of the cases of the two processes are discussed. Assume that both the processes are performed on machines *k* and *l*. There are three significant different relative positions for the two processes:

1. Process of part *k* is completely finished and then the process of part *l* is started (see Figure 10(a)). To unload part k, $4\varepsilon + (m+1)\delta + p_k$ time units are required. Then,

the robot must move back to the input station to carry part l, this activity takes $(m+1)\delta$ time units. Similarly, to finish the process of part l, $4\varepsilon+(m+1)\delta+p_l$ is required. At the end, the robot must move back to the input station once more $(m+1)\delta$, thus the total time will be $8\varepsilon+4(m+1)\delta+p_k+p_l$.

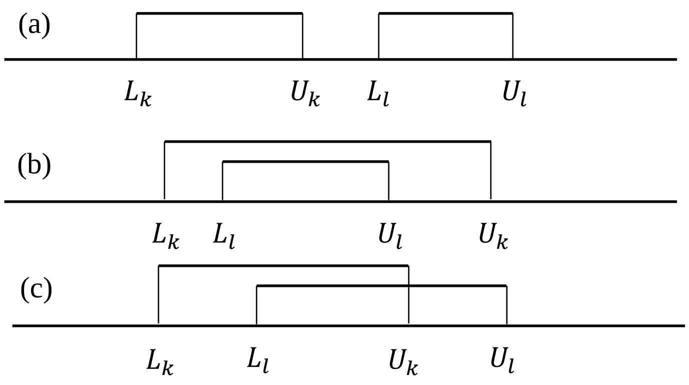

Figure 10. Different positions of parts in a cycle that affects the cycle time.

2. Another possibility is that parts k and l are loaded consecutively and then part l is unloaded first (see Figure 10(b)). To load part k, $2\varepsilon+k\delta$ time units are required for the robot to carry the part from the input station and to load machine k. To load part l, the robot must return to the input station ($k\delta$) and carry the part to load machine l ($2\varepsilon+l\delta$). The robot must wait for completing the process of part l (p_l). It must unload machine l ($(m+1-l)\delta+2\varepsilon$), return to machine k ($(m+1-k)\delta$), unload machine k ($(m+1-k)\delta+2\varepsilon$), and finally return to the input station. The total time is at least $4(m+1)\delta+8\varepsilon+p_l$ if there is no activity between these activities.

3. The last relative position is when parts k and l are loaded consecutively and then part k is unloaded first (see Figure 10(c)). In this position, the activity time of either L_l or U_k can be completed by the process time. First assume that U_k is completed. Hence, the time elements are as follows: loading of machine k ($2\varepsilon+k\delta$), returning

to the input station in ($k\delta$), loading of machine l ($2\varepsilon + l\delta$), processing of part l (p_l), unloading of part l ($(m + 1 - l)\delta + 2\varepsilon$), and returning to the input station ($(m + 1)\delta$). Therefore, the total time is $\left(2k + 2(m + 1)\right)\delta + 6\varepsilon + p_l$). If L_l is completed by the process of part k then the time elements are as follows: loading of part k ($2\varepsilon + k\delta$), processing of part k (p_k), unloading of part k ($(m + 1 - k)\delta + 2\varepsilon$), returning to machine l ($(m + 1 - l)\delta$), unloading of part l ($(m + 1 - l)\delta + 2\varepsilon$), and returning to the input station ($(m + 1)\delta$). Therefore, the total time is $(4(m + 1) - 2l)\delta + 6\varepsilon + p_k$. Both the cases provide the lower bound equal to $2(m + 2)\delta + 6\varepsilon + \min_{k=1 \text{ to } m} p_k$ that can be concluded in the following theorem.

Theorem 2: Assume that $m \geq 2$.

(i) If there are two parts, say k and l, in a cycle such that their loading and unloading activities take place in disjoint time intervals, then the cycle time is at least $4(m + 1)\delta + 8\varepsilon + p_k + p_l$.

(ii) If there are two parts, say k and l, such that the time interval of the loading and unloading activities of part k completely covers the time interval of the loading and unloading activities of part l, then the cycle time is at least $4(m + 1)\delta + 8\varepsilon + p_l$.

(iii) In any other case, the cycle time is at least $2(m + 2)\delta + 6\varepsilon + \min_{k=1 \text{ to } m} p_k$.

□

3.8 A lower bound explained by an assignment problem

The logic of the other lower bound of Gultekin et al. [5] is that the cycle time cannot be shorter than the minimum total moving time of the robot such that it serves all of the machines and returns to its initial position. Here, a new proof of the lower bound is provided. This proof explores the structure of the problem and carries an optimality analysis for several possible cycles.

Theorem 3: The cycle time is at least $2(m^2 + m)\delta + 4m\varepsilon$.

Proof: The robot performs each activity exactly once in each cycle. Thus, the description of a cycle includes one element from every row and column of Table 3. Hence, a lower bound on the cycle time can be obtained if a minimization assignment problem is solved based on the matrix shown in Table 3. Every cell of Table 3 includes 2ε as the loading and unloading activities. Therefore, if we decrease each cell by 2ε, the optimal solution of the robot movement cycle remains the same. If p_i and w_i are expressed in the form of $\frac{p_i}{\delta}\delta$ and $\frac{w_i}{\delta}\delta$, then only coefficients of δ remain, δ can be deleted from each cell, and Table 3 can be revised as Table 5. Note that *j* can only be less or greater than *k* in this table.

Table 5. Distance matrix in terms of coefficient of δ.

j \ k	L_1	…	L_k	…	L_m	U_1	…	U_k	…	U_m	Dual Coefficient
L_1	-	…	1+k	…	1+m	m+$\frac{p_1}{\delta}$	…	m+$\frac{w_k}{\delta}$	…	m+$\frac{w_m}{\delta}$	4-m
⋮	⋮	…	⋮	…	⋮	⋮	…	⋮	…	⋮	⋮
L_j	j+1	…	j+k	…	j+m	j+m-1+$\frac{w_1}{\delta}$	…	|j-k|+m-k+1+$\frac{w_k}{\delta}$	…	m-j+1+$\frac{w_m}{\delta}$	3-m+j
⋮	⋮	…	⋮	…	⋮	⋮	…	⋮	…	⋮	⋮
L_k	k+1	…	-	…	k+m	k+m-1+$\frac{w_1}{\delta}$	…	m-k+1+$\frac{p_k}{\delta}$	…	m-k+1+$\frac{w_m}{\delta}$	3-m+k
⋮	⋮	…	⋮	…	⋮	⋮	…	⋮	…	⋮	⋮
L_m	m+1	…	m+k	…	-	2m-1+$\frac{w_1}{\delta}$	…	2(m-k)+1+$\frac{w_k}{\delta}$	…	1+$\frac{p_m}{\delta}$	3
U_1	m+2	…	m+k+1	…	2m+1	-	…	2(m-k+1)	…	2	4
⋮	⋮	…	⋮	…	⋮	⋮	…	⋮	…	⋮	⋮
U_j	m+2	…	m+k+1	…	2m+1	2m	…	2(m-k+1)	…	2	4
⋮	⋮	…	⋮	…	⋮	⋮	…	⋮	…	⋮	⋮
U_m	m+2	…	m+k+1	…	2m+1	2m	…	2(m-k+1)	…	-	4
Dual Coefficient	m-2	…	m+k-3	…	2m-3	2m-4	…	2(m-k-1)	…	-2	

If all waiting times are considered to be zero, the optimal value of the assignment problem can be reduced further. It is possible to obtain a lower bound of the optimal value by solving the assignment problem defined by the coefficient matrix. The dual variables are shown in Table 5 (see the row and column of "dual coefficient"). Note that these variables are considered feasible in the dual of the assignment problem.

To be considered feasible in the dual problem, each coefficient of the matrix must be greater than or equal to the sum of the dual variables of its row and column. If the dual variables are feasible and there is a feasible solution of the assignment problem such that if a variable of the assignment problem is 1 then the related constraint of the dual problem is satisfied by the equation and both the primal and dual solutions are optimal. The sum of the dual variables associated with $L_jL_k, L_jU_k, L_kU_k, U_jL_k,$

and U_jU_k are $j+k, m+j-2k\ +1, m-k+1, m+k+1,$ and $2m–2k+2,$ respectively. The constraints for L_jL_k,U_jL_k, L_kU_k, and U_jU_k are satisfied. To check the feasibility of L_jU_k, the following formula must be applied:

L_jU_k: $m+j-2k\ +1\ \leq\ |j-k|+m-k+1+\frac{w_k}{\delta}$.

There are two different cases $j>k$ and $j<k$. If $j>k$, then $m+j-2k+1 \leq |j-k|+m-k+1+\frac{w_k}{\delta} = j-k+m-k+1+\frac{w_k}{\delta} = m+j-2k+1+\frac{w_k}{\delta}$since $w_k \geq 0$. If $j<k$, then $m+j-2k+1 \leq |j-k|+m-k+1+\frac{w_k}{\delta} = k-j+m-k+1+\frac{w_k}{\delta} = m-j+1+\frac{w_k}{\delta}$. By eliminating $m+1$ from both sides of the inequality and taking all the parameters to one side of the inequality, it will be simplified as 0 $sik-j)+\frac{w_k}{\delta}$, and because $k>j$, this inequality is always true.

If the elements of Table 5 are reduced by the row and column of dual variables, then Table 5 becomes Table 6.

Table 6. The last table of the assignment problem.

jk	L$_1$	L$_2$	…	L$_k$	…	L$_m$	U$_1$	U$_2$	…	U$_k$	…	U$_m$
L$_1$	-	$\boxed{0}$	…	0	…	0	$\frac{p_1}{\delta}$	2+$\frac{w_2}{\delta}$	…	2(k-1)+$\frac{w_k}{\delta}$	…	2(m-1)+$\frac{w_m}{\delta}$
⋮	⋮	⋮	…	⋮	…	⋮	⋮	⋮	…	⋮	…	⋮
L$_{k-1}$	0	0	…	$\boxed{0}$	…	0	$\frac{w_1}{\delta}$	$\frac{w_2}{\delta}$	…	2+$\frac{w_k}{\delta}$	…	2(m-k+1)+$\frac{w_m}{\delta}$
L$_k$	0	0	…	-	…	0	$\frac{w_1}{\delta}$	$\frac{w_2}{\delta}$	…	$\frac{p_k}{\delta}$	…	2(m-k)+$\frac{w_m}{\delta}$
⋮	⋮	⋮	…	⋮	…	⋮	⋮	⋮	…	⋮	…	⋮
L$_{m-1}$	0	0	…	0	…	$\boxed{0}$	$\frac{w_1}{\delta}$	$\frac{w_2}{\delta}$	…	$\frac{w_k}{\delta}$	…	2+$\frac{w_m}{\delta}$
L$_m$	0	0	…	0	…	0	$\boxed{\frac{w_1}{\delta}}$	$\frac{w_2}{\delta}$	…	$\frac{w_k}{\delta}$	…	$\frac{p_m}{\delta}$
U$_1$	0	0	…	0	…	0	-	$\boxed{0}$	…	0	…	0
⋮	⋮	⋮	…	⋮	…	⋮	⋮	⋮	…	⋮	…	⋮
U$_{k-1}$	0	0	…	0	…	0	0	0	…	$\boxed{0}$	…	0
U$_k$	0	0	…	0	…	0	0	0	…	-	…	0
⋮	⋮	⋮	…	⋮	…	⋮	⋮	⋮	…	⋮	…	⋮
U$_{m-1}$	0	0	…	0	…	0	0	0	…	0	…	$\boxed{0}$
U$_m$	$\boxed{0}$	0	…	0	…	0	0	0	…	0	…	-

In Table 6, the rectangles show an optimal solution assuming $w_1 = 0$. In Table 7, the rectangles show the solution of the assignment problem in terms of δ coefficients. Table 6 also shows that there are alternative optimal solutions without containing any cell with w_k values. Thus, the optimal solutions can be determined from the cells of rectangles assuming that $w_1 = 0$.

Table 7. Solution of the assignment problem in terms of δ coefficients.

jk	L_1	L_2	…	L_k	…	L_m	U_1	U_2	…	U_k	…	U_m
L_1	-	$\boxed{3}$	…	1+k	…	1+m	$m+\frac{p_1}{\delta}$	$m+\frac{w_2}{\delta}$	…	$m+\frac{w_k}{\delta}$	…	$m+\frac{w_m}{\delta}$
⋮	⋮	⋮	…	⋮	…	⋮	⋮	⋮	…	⋮	…	⋮
L_{k-1}	k	k+1	…	$\boxed{2k-1}$	…	k+m-1	$m+k-2+\frac{w_1}{\delta}$	$k+m-4+\frac{w_2}{\delta}$	…	$m-k+2+\frac{w_k}{\delta}$	…	$m-k+2+\frac{w_m}{\delta}$
L_k	k+1	k+2	…	-	…	k+m	$m+k-2-\frac{w_1}{\delta}$	$k+m-3+\frac{w_2}{\delta}$	…	$m-k+1+\frac{p_k}{\delta}$	…	$m-k+1+\frac{w_m}{\delta}$
⋮	⋮	⋮	…	⋮	…	⋮	⋮	⋮	…	⋮	…	⋮
L_{m-1}	m	m+1	…	m+k-1	…	$\boxed{2m-1}$	$2m-2+\frac{w_1}{\delta}$	$2m-4+\frac{w_2}{\delta}$	…	$2(m-k)+\frac{w_k}{\delta}$	…	$2+\frac{w_m}{\delta}$
L_m	m+1	m+2	…	m+k	…	2m	$\boxed{2m-1+\frac{w_1}{\delta}}$	$2m-3+\frac{w_2}{\delta}$	…	$2(m-k)+1+\frac{w_k}{\delta}$	…	$1+\frac{p_m}{\delta}$
U_1	m+2	m+3	…	m+k+1	…	2m+1	-	$\boxed{2(m-1)}$	…	2(m-k+1)	…	2
⋮	⋮	⋮	…	⋮	…	⋮	⋮	⋮	…	⋮	…	⋮
U_{k-1}	m+2	m+3	…	m+k+1	…	2m+1	2m	2(m-1)	…	$\boxed{2(m-k+1)}$	…	2
U_k	m+2	m+3	…	m+k+1	…	2m+1	2m	2(m-1)	…	-	…	2
⋮	⋮	⋮	…	⋮	…	⋮	⋮	⋮	…	⋮	…	⋮
U_{m-1}	m+2	m+3	…	m+k+1	…	2m+1	2m	2(m-1)	…	2(m-k+1)	…	$\boxed{2}$
U_m	$\boxed{m+2}$	m+3	…	m+k+1	…	2m+1	2m	2(m-1)	…	2(m-k+1)	…	-

The cycle times considering the rectangles of Table 7 are calculated as follows:

$$T = 4m\varepsilon + \big((m+2) + \sum_{k=2}^{m}(2k-1) + (2m-1) + \sum_{k=2}^{m} 2(m-k+1)\big)\delta.$$

Because $\sum_{k=2}^{m}(2k-1) = m^2 - 1$ and $\sum_{k=2}^{m} 2(m-k+1) = m^2 - m$, the cycle time can be calculated as follows:

$$T = 4m\varepsilon + (m + 2 + m^2 - 1 + 2m - 1 + m^2 - m)\delta = 4m\varepsilon + (2m^2 + 2m)\delta. \ \square$$

3.9 Optimal cycles of different structures in the general case

The optimal solution of the assignment problem provides a lower bound for the minimum cycle time. If the processing times are such that all waiting times are zero, then the lower bound is equal to the cycle time and the cycle is optimal. Since the assignment problem has many optimal solutions, we determine the conditions for optimal solutions in this section.

Theorem 4: A cycle is an optimal solution of the assignment problem if and only if there are no machine indices k and l such that L_k is immediately followed by U_l when $k \leq l$.

Proof: Table 6 contains positive values for these activities in the right upper triangle of the matrix. $\square$

There are many cycles that satisfy the condition of the theorem. For example, the cycle $L_1 L_2 \dots L_m U_1 U_2 \dots U_m$that can be generalized as $L_{i_1} L_{i_2} \dots L_{i_m} U_{i_1} U_{i_2} \dots U_{i_m}$. These cycles are discussed in Sections 3.9.2 and 3.9.3.

3.9.1 Basic tools

To find the optimal solution(s) of the problem, we first solve the problem when the processing times are small enough so that they can be excluded from the calculations ($p_i = 0$). Then, the solution is fixed and under this condition the minimum of p_i, considering that all the waiting times (w_i) are still zero, is maximized.

All loaded robot movements consist of the same elements: carry a part from the input station and load machine k (which needs $k\delta + 2\varepsilon$ time units) or unload the same machine and put that part at the output station (that needs $(m + 1 - k)\delta + 2\varepsilon$ time units). Therefore, the total time is equal to $(m + 1)\delta + 4\varepsilon$ time units for every part in one pure cycle. Hence, to minimize the cycle time for one pure cycle, only the unloaded robot movement can be considered and must be minimized. Thus, it is sufficient to determine the optimal (minimum) solution of the assignment problem defined by the matrix consisting of the time distances between two machines as a unit (δ). Table 8 shows the coefficients of δ for only unloaded robot movements.

Table 8. Matrix of δ coefficients for only unloaded robot movements.

ik	L1	…	Li	…	Lk	…	Lm	U1	…	Uk	…	Um
L1	-	…	1	…	1	…	1	0	…	k-1	…	m-1
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
Li	i	…	-	…	i	…	i	i-1	…	|k-i|	…	m-i
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
Lk	k	…	k	…	-	…	k	k-1	…	0	…	m-k
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
Lm	m	…	m	…	m	…	-	m-1	…	m-k	…	0
U1	m+1	…	m+1	…	m+1	…	m+1	-	…	m-k+1	…	1
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
Ui	m+1	…	m+1	…	m+1	…	m+1	m	…	m-k+1	…	1
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
Uk	m+1	…	m+1	…	m+1	…	m+1	m	…	-	…	1
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
Um	m+1	…	m+1	…	m+1	…	m+1	m	…	m-k+1	…	-

Table 7 shows an optimal solution for the assignment problem considering δ coefficients that are the same as provided in Table 8. The optimality of this solution is discussed in the next section. The primary tool of the analysis is the *return time* that is discussed in Section 3.4.

3.9.2 The $L_{i_1}L_{i_2}\dots L_{i_m}U_{i_1}U_{i_2}\dots U_{i_m}$ cycle

Lemma 2: The minimum return time is not less than $2(m-1)\varepsilon+\left(\frac{m^2-1}{2}\right)\delta$ for any of the solutions of type $L_{i_1}L_{i_2}\dots L_{i_m}U_{i_1}U_{i_2}\dots U_{i_m}$.

Proof: To prove that the statement is correct, at first the return time to machines i_1 and i_2 is calculated. Then, the return time to machine i_k is calculated and its minimum value is computed. Similarly, the return time to machine i_m is calculated and after comparing all of them the minimum value is computed.

Return time to machine i_1: The return time to machine $i_1$for this cycle is the time of movement from L_{i_1} to U_{i_1}which includes $L_{i_2} \dots L_{i_m} U_{i_1}$. Such a robot movement sequence is called the *return cycle* and the return time is the time required from loading a machine until the robot reaches the same machine to unload it. It is clear that all the waiting times of the minimum return time are equal to zero. To calculate the return time to machine i_1, the loaded and unloaded robot movement times must be considered. Table 9 shows the loaded and unloaded robot movement time distances.

Table 9. Loaded and unloaded robot movement time distances.

Activity	L_{i_2}	L_{i_3}	…	L_{i_m}	U_{i_1}	U_{i_2}	…	$U_{i_{m-1}}$	U_{i_m}
Unloaded movement	i_1	i_2	…	i_{m-1}	$\|i_m - i_1\|$	$m+1-i_2$	…	$m+1-i_{m-1}$	$m+1-i_m$
Loaded movement	i_2	i_3	…	i_m	$m+1-i_1$	$m+1-i_2$	…	$m+1-i_{m-1}$	

In the first row of Table 9, the order of the robot movements' cycle of $L_{i_1} L_{i_2} \dots L_{i_m} U_{i_1} U_{i_2} \dots U_{i_m}$ has been shown. The second and third rows of Table 9 show the unloading and loading times of the robot movement in detail (in terms of coefficients of δ), respectively. For example, to load machine i_2 just after loading machine i_1, the robot requires $i_1\delta$ time units to reach the input station to carry a new part (unloaded robot movement) and then it requires $i_2\delta$ time units to move from the input station to machine i_2. To calculate the summation of the loaded and unloaded robot movements for this case, all of the robot movements from every machine to the input station except machine i_m must be considered as follows:

In the second row of Table 9, there is no robot movement from the last loaded machine (i_m) to the input station. Instead, there is a robot movement to the first machine for which the return cycle is started with (i_1). This robot movement requires

$|i_m - i_1|\delta$ time units. Moreover, in the third row of Table 9, there is no robot movement from the input station to the machine for which the return cycle is started with (i_1). It is clear that to calculate such robot movements we must consider $2(m-1)\varepsilon$ time units for taking $m-1$ new parts from the input station and loading them on the machines (considering that at the beginning of the cycle machine i_1 has been loaded). Therefore, the return time will be calculated as follows:

Return time $= 2(m-1)\varepsilon + \left(\left(\sum_{j=1}^{m-1} i_j\right) + |i_m - i_1| + \left(\sum_{j=2}^{m} i_j\right)\right)\delta =$

$2(m-1)\varepsilon + \left(\sum_{j=1}^{m} j - i_m + |i_m - i_1| + \sum_{j=1}^{m} j - i_1\right)\delta = 2(m-1)\varepsilon +$

$(m(m+1) + |i_m - i_1| - i_1 - i_m)\delta.$

Here, $2(m-1)\varepsilon + m(m+1)\delta$ is a constant value. The minimum of $(|i_m - i_1| - i_1 - i_m)\delta$ is equal to $\begin{cases} -2i_1 \text{ if } i_m > i_1 \\ -2i_m \text{ if } i_m < i_1 \end{cases}$ and its minimum value is $-2(m-1)\delta$ if $i_m = m$ and $i_1 = m-1$ or $i_m = m-1$ and $i_1 = m$. Therefore, in this case the minimum return time is $2(m-1)\varepsilon + (m^2 + m - 2)\delta$.

Return time to machine *i₂*: To calculate the return time to machine L_{i_2}, the values of the L_{i_2}column in Table 9 must be subtracted from the calculations of return time to machine i_1. Also, the times of the robot movements from machine i_1 to the output station and from the output station to machine i_2 must be added to its calculations. Therefore, the return time to machine i_2 is calculated as follows:

Return time to i_2 $=$ Return time to $i_1 - (i_1 + i_2)\delta + (2(m+1) - i_1 - i_2)\delta =$

$2(m-1)\varepsilon + (m(m+1) + |i_m - i_1| - i_1 - i_m)\delta - (i_1 + i_2)\delta + (2(m+1) - i_1 - i_2)\delta = 2(m-1)\varepsilon + (m(m+1) + 2(m+1) + |i_m - i_1| - 3i_1 - 2i_2 - i_m)\delta.$

Similar to machine i_1, the coefficient of δ can be calculated as follows:

$(m+2)(m+1)+\begin{cases}-4i_1-2i_2 & \text{if } i_m>i_1\\ -2i_1-2i_2-2i_m & \text{if } i_m<i_1\end{cases}$. If $i_m>i_1$, then in this case the minimum value of the coefficient of δ for this case is $(m+2)(m+1)-4(m-1)-2(m-2)$ when $i_m=m$, $i_1=m-1$, and $i_2=m-2$. However, if $i_m<i_1$, the minimum value of the coefficient of δ is $(m+2)(m+1)-2m-2(m-1)-2(m-2)$ when $i_1=m-1$, $i_2=m$, and $i_m=m-2$ or when $i_1=m$ either $i_2=m-1$ and $i_m=m-2$ or $i_2=m-2$ and $i_m=m-1$.

The comparison of these two values shows that the minimum return time to machine i_2 is$2(m-1)\varepsilon+\big((m+2)(m+1)-6(m-1)\big)\delta=2(m-1)\varepsilon+(m^2-3m+8)\delta$.

Return time to machine i_k: Starting from the return time of i_1 some new elements must be added and some other elements must be excluded as follows:

Return time to $i_k=$ Return time to $i_1-(i_1+2i_2+2i_3+\cdots+2i_{k-1}+i_k)\delta+(2(k-1)(m+1)-i_1-2i_2-2i_3-\cdots-2i_{k-1}-i_k)\delta=2(m-1)\varepsilon+(m(m+1)+|i_m-i_1|-i_1-i_m)\delta+\big(2(k-1)(m+1)-2i_1-4\sum_{j=2}^{k-1}i_j-2i_k\big)\delta$.

To minimize the return time to machine i_k, not considering the constant part of it, $2(m-1)\varepsilon+\big(m(m+1)+2(k-1)(m+1)\big)\delta$,the negative parts,$\big(-2i_1-4\sum_{j=2}^{k-1}i_j-2i_k-i_1-i_m\big)\delta$, should be maximized in absolute value and the positive parts, $(|i_m-i_1|)\delta$, should be minimized at the same time as the following maximization objective function:

$$\text{Maximize } 4\sum_{j=2}^{k-1} i_j + 3i_1 + 2i_k + i_m - \begin{cases} i_m - i_1 \text{ if } i_m > i_1 \\ i_1 - i_m \text{ if } i_m < i_1 \end{cases}. \tag{18}$$

Considering i_1 and i_m, objective function (18) can be divided into two parts as follows:

$$\text{Maximize } 4\sum_{j=1}^{k-1} i_j + 2i_k \text{ if } i_m > i_1 \tag{19}$$

and

$$\text{Maximize } 4\sum_{j=2}^{k-1} i_j + 2i_1 + 2i_k + 2i_m \text{ if } i_m < i_1. \tag{20}$$

To maximize objective function (19), the highest value of $4\sum_{j=1}^{k-1} i_j$ is obtained when i_j is related to the last $k-1$ machines that are from machine $m-k+2$ to machine m, regardless of the order of machines that the robot visits. Also, the highest value of $2i_k$, after excluding the last $k-1$ machines in the line, is obtained when i_k is machine $m-k+1$. Therefore, the maximum value of objective function (19) will be $4\sum_{j=m-k+2}^{m} j + 2(m-k+1) = 2(-k^2 + (2m+2)k - m - 1)$.

To find the maximum value of objective function (20), similar to the calculations related to objective function (19), $4\sum_{j=2}^{k-1} i_j$ can be in any sequence of the last $k-2$ machines in the line layout, and i_1, i_k, and i_m can be in any order of the last three machines before machine $m-k+3$ for which $i_m < i_1$. The maximum value of objective function (20) is

$4\sum_{j=m-k+3}^{m} j + 2\sum_{j=m-k}^{m-k+2} j = 2(-k^2 + (2m+2)k - m - 3)$.

Hence, the objective function (19) is greater than the objective function (20) and the return time to machinei_k is at least $2(m-1)\varepsilon + \big((m+1)(m+2k-2) -$

$2(-k^2 + (2m + 2)k - m - 1)\big)\delta = 2(m - 1)\varepsilon + ((m - k)^2 + (k - 1)^2 + m - 1)\delta = 2(m - 1)\varepsilon + (m^2 - 2mk - 2k + m + 2k^2)\delta.$

To find the minimum return time to machine i_k, $2(m - 1)\varepsilon + (m^2 + m)\delta$ depends only on m but the remaining part, which is $(2k^2 - 2mk - 2k)\delta$, must be minimized. Therefore, the derivation $\frac{\partial(2k^2-2mk-2k)}{\partial k} = 4k - 2m - 2 = 0$, where $k = \frac{m+1}{2}$. By putting $\frac{m+1}{2}$ instead of k in $2(m - 1)\varepsilon + (m^2 - 2mk - 2k + m + 2k^2)\delta$, the minimum return time to machine $i_k$will be equal to $2(m - 1)\varepsilon + \left(\frac{m^2-1}{2}\right)\delta$.

Return time to machine $\boldsymbol{i_m}$: Considering Table 9, the return time to machine $i_m$is calculated as follows:

Return time to $i_m = 2(m - 1)\varepsilon + \left(|i_m - i_1| + (m - 1)(m + 1) - \left(\sum_{j=2}^{m} i_j\right)\right)\delta + \left((m - 1)(m + 1) - \left(\sum_{j=1}^{m-1} i_j\right)\right)\delta = 2(m - 1)\varepsilon + (|i_m - i_1| + (m - 2)(m + 1) + i_1 + i_m)\delta.$

Its minimum value is $2(m - 1)\varepsilon + \big((m - 2)(m + 1) + 2\big)\delta = 2(m - 1)\varepsilon + (m^2 - m)\delta$. It is obtained when either $i_1 = 1$ or $i_m = 1$.

The comparison among the minimum values of i_1, i_2, i_k, and i_m which are equal to $2(m - 1)\varepsilon + (m^2 + m - 2)\delta$, $2(m - 1)\varepsilon + (m^2 - 3m + 8)\delta$, $2(m - 1)\varepsilon + \left(\frac{m^2-1}{2}\right)\delta$, and $2(m - 1)\varepsilon + (m^2 - m)\delta$, respectively, shows that the return time to machine $k = \frac{m+1}{2}$ when m is an odd number and $k = \frac{m}{2}$ or $k = \frac{m}{2} + 1$ when m is an even number. Therefore, we can obtain the minimum possible return time in the cycle of the type $L_{i_1}L_{i_2}\dots L_{i_m}U_{i_1}U_{i_2}\dots U_{i_m}$that is equal to $2(m-1)\varepsilon+\left(\frac{m^2-1}{2}\right)\delta$. This value is obtained when the last $k-1$ machines have the indices from $m-k+2$ to m (regardless of the order of how the robot services them) and also for the machines $i_{m-k+1}=m-k+2$. □

3.9.3 The *$L_1L_mU_{m-1}L_{m-1}\dots U_2L_2U_1U_m$* cycle

Lemma 3: The minimum return time is not greater than $(4m-6)\varepsilon+(2m^2-4)\delta$ for any of the solutions of the type $L_1L_mU_{m-1}L_{m-1}\dots U_2L_2U_1U_m$. This value is obtained for the return time to machines 1 and m.

Proof: Considering Table 6, the cycle time of $L_1L_mU_{m-1}L_{m-1}\dots U_2L_2U_1U_m$ is one of the minimum cycle times when all the waiting times are zero. According to Theorem 3, it is equal to $4m\varepsilon+2(m^2+m)\delta$. To calculate the return times of all the machines, we can subtract the complementary of the return time for each machine from the cycle time presented in Theorem 2. The complementary of a return time, let say for machine i, is the robot movements' time from when the robot is beside machine i to unload it to the end of loading the same machine. For example, to calculate the complementary of the return time to machine k when U_k is followed by L_k, only unloading machine k (when the robot is beside it) and loading the same machine must be considered. This duration of time is the summation of the time for unloading machine k (which is equal to $2\varepsilon+(m+1-k)\delta$), the robot movement from the output station to the input station ($(m+1)\delta$), and loading machine $k(2\varepsilon+k\delta)$,. Therefore, the total time is $4\varepsilon+2(m+1)\delta$. Since the complementary of return times for all the machines except machines 1 and m of the given cycle is equal to $4\varepsilon+2(m+1)\delta$, therefore the return times for these machines are calculated as follows:

$$(4m\varepsilon + 2(m^2 + m)\delta) - (4\varepsilon + 2(m + 1)\delta) = (4m - 4)\varepsilon + (2m^2 - 2)\delta.$$

In the same way, it can be shown that the return times to machines 1 and m are equal to $(4m - 6)\varepsilon + (2m^2 - 4)\delta$. Thus, the minimum return time for the cycle $L_1 L_m U_{m-1} L_{m-1} \dots U_2 L_2 U_1 U_m$ is obtained for machines 1 and m. □

Chapter 4

THE FMC WITH INDIVIDUAL BUFFER FOR EACH MACHINE

4.1 Preface

In this chapter, we consider m parallel and identical CNC machines placed on a line. Different parts with different processes can be performed by each machine and consequently the process time of each machine can be different. Each machine contains individual input buffer. There are an input station in which the items to be processed are kept and an output station in which the finished items are kept. When an item is processed by any of the machines, it becomes a finished item and it must be taken to the output station. There is a robot that performs the loading/unloading activities and transports the items.

4.2 Sequential part production matrix

To determine the optimal cycle time in an FMC, all of the potential cycles of the system should be considered. To represent the cycles, the sequential part production (SPP) matrix for an m-machine FMC when machines 1 to m produce $n_1, n_2, \dots, n_m$ parts, respectively, is presented in the following.

The SPP matrix contains two rows. The first row represents the loading operations and the second row represents the unloading operations. Every column indicates one production; therefore, there are $n_1 + n_2 + \cdots + n_m$ columns. The columns related to the production of machines are separated with the vertical lines in the matrix and the sequence of the operations is distinguished by numbers. For example, suppose there are two machines in a cell and each machine produces two parts, there will be a 2*4 SPP matrix like $\begin{bmatrix} - & - & | & - & - \\ - & - & | & - & - \end{bmatrix}$. Figure 11 presents the cycle $L_1L_2L_1L_2U_1U_2U_1U_2$. This cycle is shown by using the SPP matrix. The first activity is loading machine 1, so in the first step the matrix will be $\begin{bmatrix} 1 & - & | & - & - \\ - & - & | & - & - \end{bmatrix}$ in the first step. In the next step, machine 2 is loaded. Therefore, the matrix will become $\begin{bmatrix} 1 & - & | & 2 & - \\ - & - & | & - & - \end{bmatrix}$. Then, to load the buffers of machines 1 and 2, respectively, the matrix will become $\begin{bmatrix} 1 & 3 & | & 2 & 4 \\ - & - & | & - & - \end{bmatrix}$. To complete the cycle, machines 1 and 2 will be unloaded, respectively. Thus, the matrix will be represented as $\begin{bmatrix} 1 & 3 & | & 2 & 4 \\ 5 & 7 & | & 6 & 8 \end{bmatrix}$.

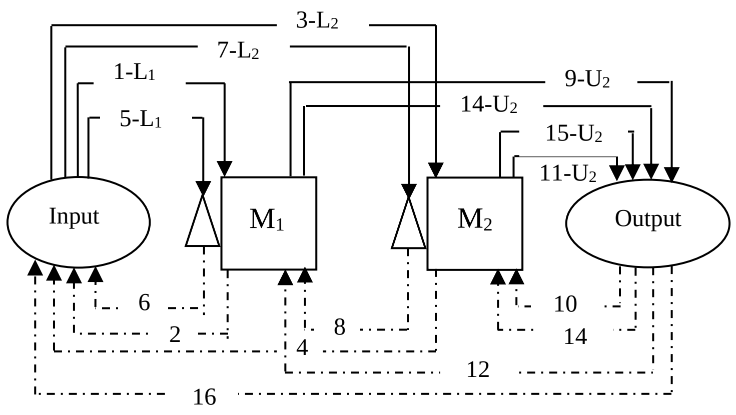

Figure 11. Robot movement sequence related to the $L_1L_2L_1L_2U_1U_2U_1U_2$ cycle.

To determine the number of different cycles in an m-machine FMC without any individual buffer before machines, since each machine must be loaded and unloaded one time, there are two activities related to each machine and totally there are 2m unique activities in a cycle. To prevent the repetitive permutation, such as $L_1L_2U_1U_2$ and $U_1U_2L_1L_2$ that are different representations of the same cycle, without loss of generality it can be assumed that all of the cycles start with activity L_1. Hence, there are (2m−1)! different cycles in such a cell. Based on this idea, to determine all of the cycles for an m-machine FMC with an input buffer before each machine (when each buffer has only the capacity of storing one part), and assuming that 2m parts are produced in a cycle in which each machine produces exactly two parts, there are four activities related to each machine including two loading and two unloading activities. In this case, the SPP matrix is represented as $\left[\begin{array}{cc|cc|c|cc} - & - & - & - & \ldots & - & - \\ - & - & - & - & & - & - \end{array}\right]$. In this matrix, there are m−1 vertical lines that separate all of the activities related to each of the machines. Generally, an m-machine cell consists $\frac{(4m-1)!}{(2!)^{2m-1}}$ different cycles in which the first loading of machine 1 is the first activity and the other L_1 can be in any place from the second to the last.

4.3 Definitions, parameters and sets

The loading of machine i includes taking an item from the input station, robot movement from the input station to machine i, and putting the item to the input buffer of the machine. It is assumed that when a machine is idle and there is an item in its input buffer, the machine takes this item from its input buffer and starts processing it automatically. It means there is no need to use the robot for this operation. The unloading of machine i includes taking the finished item from the machine by the robot, the robot movement from the machine i to the output station, and putting the item to the output station. Note that the robot stays at machine i at the end of the loading of machine i, and stays at the output station after the unloading of any machine.

It is assumed that the system repeats the same cycle in the long run. If the system is at a specific state at the beginning of a cycle, it reaches the same state at the end of the cycle and repeats the same activities in the following cycle. The duration of a cycle is called the cycle time. It is assumed that each machine processes only two items in each cycle. Let L_{ik} be the loading and U_{ik} be the unloading of the kth item of machine i in each cycle. Let L_i be the set of loading activities of machine i, i.e., $L_i = \{L_{ik} | \ k=1,2\}$, and U_i be the set of unloading activities of machine i, i.e., $U_i = \{U_{ik} | \ k=1,2\}$. Let A be the set of all loading and unloading activities, i.e., $A=\{L_{ik}, U_{ik} | \ i=1,2,\ldots, m; k=1,2\}$.

Let ε be the time for taking an item from the input station or from a machine, or the time for putting an item to the input buffer of a machine or to the output station. Let δ be the travel time of the robot for a one-unit distance. The distances between the input station and the first machine, between any two consecutive machines, and between the last machine and the output station are assumed to be one unit of distance. So, the time that the robot needs to perform activity b after finishing activity a (dab) is

$$d_{ab} = \begin{cases} 2\varepsilon + (i+j)\delta & \text{if } a \in L_i \text{ and } b \in L_j \\ 2\varepsilon + (|i-j| + m + 1 - j)\delta & \text{if } a \in L_i \text{ and } b \in U_j \\ 2\varepsilon + 2(m+1-j)\delta & \text{if } a \in U_i \text{ and } b \in U_j \\ 2\varepsilon + (m+1+j)\delta & \text{if } a \in U_i \text{ and } b \in L_j \end{cases}$$

On the other hand, the robot may need to wait before starting the unloading of an item if its process has not finished. The time between the completion times of activities a, and b related to machine i cannot be less than the following values (MT^i_{ab}):

$$MT^i_{ab} = \begin{cases} 2\varepsilon + (m+1-i)\delta + p_i & \text{if } a = L_{i1} \text{ and } b = U_{i1} \\ 2\varepsilon + (m+1-i)\delta + p_i & \text{if } a = L_{i2} \text{ and } b = U_{i2} \\ p_i & \text{if } a, b \in U_i \\ d_{ab} & \text{otherwise} \end{cases}$$

In order to start such a cyclic production, the system needs a setup. All of the loading and unloading activity orders of machine i at the beginning of the first cycle are as follows:

- The machine is idle and its input buffer is empty:

1. L_{i1}, L_{i2}, U_{i1}, U_{i2}

2. L_{i1}, U_{i1}, L_{i2}, U_{i2}

- The machine has a processed part and its input buffer is empty:

3. L_{i2},U_{i1}, U_{i2}, L_{i1}

4. L_{i2}, U_{i1}, L_{i1}, U_{i2}

5. U_{i1}, L_{i2}, U_{i2}, L_{i1}

6. U_{i1}, L_{i2}, L_{i1}, U_{i2}

- The machine has a processed part and its input buffer is full:

7. U_{i1}, U_{i2}, L_{i1}, L_{i2}

8. U_{i1}, L_{i1}, U_{i2}, L_{i2}

The order of the loading and unloading activities of each machine can be one of the eight above-mentioned orders in the cyclic production. According to the orders of the activities, each machine should be setup before starting the cyclic production and then the system may repeat the same operations continuously.

Since the system will repeat the same cycle continuously, each activity may be considered as the first activity. We consider L_{11} as the first activity in this study. Therefore, the time between two L_{11} is called the cycle time. A reduction in the cycle time means an increase in the production rate of such a system. The problem is to determine the order of all loading and unloading activities for minimizing the cycle time. We have developed the following mixed integer programming formulation to resolve this problem where the decision variables are as follows:

T: cycle time

t_a: completion time of activity $a \in A$

$$x_{ab} = \begin{cases} 1 & \text{if robot performs activity b after activity a} \\ 0 & \text{o.w.} \end{cases}$$

$$z_{ik} = \begin{cases} 1 & \text{if k}^{\text{th}} \text{ order is applied for the actvities of machine i; k} = 1, 2, ..., 8 \\ 0 & \text{o.w.} \end{cases}$$

4.4 Mathematical model

The mathematical formulation for the provided problem is the following:

Minimize T (21)

$$\sum_{b \in A-a} x_{ab} = 1,\ \forall a \in A, \qquad (22)$$

$$\sum_{a \in A-b} x_{ab} = 1,\ \forall b \in A, \qquad (23)$$

$$t_b \geq t_a + d_{ab} - M(1 - x_{ab}),\ \forall b \in A - L_{11}, a \in A - b, \qquad (24)$$

$$T \geq t_a + d_{aL11} x_{aL11},\ \forall a \in A - L_{11}, \qquad (25)$$

$$\sum_{j=1}^{8} z_{ij} = 1,\ i = 1, ..., m, \qquad (26)$$

$$t_{Li1} - t_{Ui1} \leq M(z_{i3} + z_{i4} + z_{i5} + z_{i6} + z_{i7} + z_{i8}),\ i = 1, ..., m, \qquad (27)$$

$$t_{Li1} - t_{Ui2} \leq M(z_{i3} + z_{i5} + z_{i7}),\ i = 1, ..., m, \qquad (28)$$

$$t_{Li1} - t_{Li2} \leq M(z_{i3} + z_{i4} + z_{i5} + z_{i6}),\ i = 1, ..., m, \qquad (29)$$

$$t_{Ui1} - t_{Li1} \leq M(z_{i1} + z_{i2}),\ i = 1, ..., m, \qquad (30)$$

$$t_{Ui1} - t_{Li2} \leq M(z_{i1} + z_{i3} + z_{i4}),\ i = 1, ..., m, \qquad (31)$$

$$t_{Li2} - t_{Ui1} \le M(z_{i2} + z_{i5} + z_{i6} + z_{i7} + z_{i8}),\ i = 1,\dots,m, \tag{32}$$

$$t_{Li2} - t_{Ui2} \le M(z_{i7} + z_{i8}),\ i = 1,\dots,m, \tag{33}$$

$$t_{Li2} - t_{Li1} \le M(z_{i1} + z_{i2} + z_{i7} + z_{i8}),\ i = 1,\dots,m, \tag{34}$$

$$t_{Ui2} - t_{Li1} \le M(z_{i1} + z_{i2} + z_{i4} + z_{i6} + z_{i8}),\ i = 1,\dots,m, \tag{35}$$

$$t_{Ui2} - t_{Li2} \le M(z_{i1} + z_{i2} + z_{i3} + z_{i4} + z_{i5} + z_{i6}),\ i = 1,\dots,m, \tag{36}$$

$$t_{Ui1} \ge t_{Li1} + MT^{i}_{L1,U1} - M(1 - z_{ik}),\ i = 1,\dots,m; k = 1,2, \tag{37}$$

$$t_{Li2} \ge t_{Ui1} + MT^{i}_{U1,L2} - M(1 - z_{ik}),\ i = 1,\dots,m; k = 2,5,6, \tag{38}$$

$$t_{Ui2} \ge t_{Li2} + MT^{i}_{L2,U2} - M(1 - z_{ik}),\ i = 1,\dots,m; k = 2,4,5,6, \tag{39}$$

$$t_{Ui2} \ge t_{Li2} + \varepsilon + MT^{i}_{L2,U2} - M(1 - z_{ik}),\ i = 1,\dots,m; k = 1,3, \tag{40}$$

$$t_{Li1} \ge t_{Ui2} + MT^{i}_{U2,L1} - M(1 - z_{ik}),\ i = 1,\dots,m; k = 3,5,7, \tag{41}$$

$$t_{Li2} \ge t_{Li1} + MT^{i}_{L1,L2} - M(1 - z_{ik}),\ i = 1,\dots,m; k = 1,7, \tag{42}$$

$$t_{Li1} \ge t_{Ui1} + MT^{i}_{U1,L1} - M(1 - z_{ik}),\ i = 1,\dots,m; k = 4,8, \tag{43}$$

$$t_{Ui2} \ge t_{Li1} + MT^{i}_{L1,U2} - M(1 - z_{ik}),\ i = 1,\dots,m; k = 4,6,8, \tag{44}$$

$$t_{Li1} \ge t_{Li2} + MT^{i}_{L2,L1} - M(1 - z_{ik}),\ i = 1,\dots,m; k = 6, \tag{45}$$

$$t_{Li2} \ge t_{Ui2} + MT^{i}_{U2,L2} - M(1 - z_{ik}),\ i = 1,\dots,m; k = 8, \tag{46}$$

$$t_{Ui1} \ge t_{Li2} + MT^{i}_{L2,U1} - M(1 - z_{ik}),\ i = 1,\dots,m; k = 1,3,4, \tag{47}$$

$$t_{Ui2} \ge t_{Ui1} + MT^{i}_{U1,U2} - M(1 - z_{ik}),\ i = 1,\dots,m; k = 1,3,4,7,8, \tag{48}$$

$$t_{Li1} - t_{Ui1} \le T - MT^{i}_{L1,U1} z_{ik},\ i = 1,\dots,m; k = 3,4,5,6,7,8, \tag{49}$$

$$t_{Li2} - t_{Ui2} \le T - MT^{i}_{L2,U2} z_{ik},\ i = 1,\dots,m; k = 8, \tag{50}$$

$$t_{Li2} - t_{Ui2} \le T - (\varepsilon + MT^{i}_{L2,U2}) z_{ik},\ i = 1,\dots,m; k = 7, \tag{51}$$

$$t_a \geq 0,\ \forall a \in A, \tag{52}$$

$$T \geq 0 \tag{53}$$

$$x_{ab} \in \{0,1\},\ \forall a \in A, b \in A - a, \tag{54}$$

$$z_{ij} \in \{0,1\},\ i = 1,\dots,m;\ j = 1,\dots,6, \tag{55}$$

The objective function of this model is to minimize the cycle time. Constraint (22) guarantees that the robot passes from each activity to another activity. Constraint (23) guarantees that the robot passes to each activity from another activity. Constraints (22) and (23) together guarantee that each activity is performed by the robot. Constraint (24) computes the completion times of the activities considering the robot moving time to perform successive activities. Constraint (24) also eliminates the sub-cycles. Constraint (25) computes the cycle time considering the completion time of the last activity and the robot moving time until just before performing the first activity of the cycle. Constraint (26) guarantees that for each machine one of the eight possible orders is applied. Constraints (27)–(36) compare the completion times of the activities of each machine and force the related z variables to be 1. Since in all of the eight orders, U_{i1} is before U_{i2}; considering constraint (26), there is no need to use the similar constraints for U_{i2}–U_{i1}. Constraints (37)–(51) compute the completion times of the activities considering the process times of the machines and the time that the robot needs to perform these tasks.

Two of the main concerns of the researchers are the execution time reduction of the exact methods and to solve the larger problems. In the present study, for decreasing the CPU time to solve the small-size problems, a linear relaxation approach is employed. Using this approach, some of the fractional solutions are eliminated from the solution space. The linear relaxation is expressed as follows:

$$T \leq Ub\,, \tag{56}$$

where *Ub* is an upper bound for the cycle time. To find *Ub*, a system when only machine 1 has been loaded and the robot is besides it is considered. Generally, the activity orders for this cycle are $L_1L_2L_2\ldots L_mL_mU_1U_1\ldots U_mL_1$. To find *Ub* by using a mathematical model, constraints (57)–(61) must be added to the proposed mathematical model (PMM) as follows:

$$t_{Li1} < t_{Li2}\,,\ i = 1,\ldots,m\,, \tag{57}$$

$$t_{Li2} < t_{Lj1}\,,\ i = 1,\ldots,m-1,\ j = i+1, \tag{58}$$

$$t_{Lm2} < t_{U11}, \tag{59}$$

$$t_{Ui1} < t_{Ui2}\,,\ i = 1,\ldots,m\,, \tag{60}$$

$$t_{Ui2} < t_{Uj1}\,,\ i = 1,\ldots,m-1,\ j = i+1\,. \tag{61}$$

The cycle time, which is obtained in less than 3 s, is considered as *Ub* and must be added to the original mathematical model for the second run. This method is called the time reduction method (TRM) in this study.

4.5 Numerical results

To compare the performance of TRM and PMM, several small-size problems with identical parts and processing times are considered. Both the exact models are coded in CPLEX 12.6 software and executed on an Intel(R) Pentium(R) Dual CPUE 2180 at 2.00 GHz CPU with a RAM of 2.0 GB. Table 10 shows the objective functions, lower bounds, and the CPU times for solving the two-machine problems using both the proposed exact models.

Table 10. The results of the mathematical models for two-machine cells.

Processing times	Objective function	Lower Bound	*CPU* time (in seconds)	
			PMM	*TRM*
30	76	112	0.83	0.77
35	86	127	1.14	0.51
40	96	142	0.98	0.79
45	106	157	0.92	0.64
50	116	172	1.00	0.85
55	126	187	1.25	0.92
60	136	202	0.90	0.71
65	146	217	1.07	0.76
70	156	232	0.79	0.78
75	166	247	0.98	0.81
80	176	262	1.03	1.01
85	186	277	1.06	1.20
90	196	292	0.81	0.89
95	206	307	1.11	0.78
100	216	322	1.00	0.71

From Table 10, it can be observed that when the processing times of both the machines are the same and equal to 30 time units, the cycle time will be 76 units. This objective function is obtained by PMM in 0.83 s using the lower bound of 112 time units. TRM obtains the optimal solution in 0.77 s. To compare the performance of both the methods, their CPU times are plotted in Figure 12.

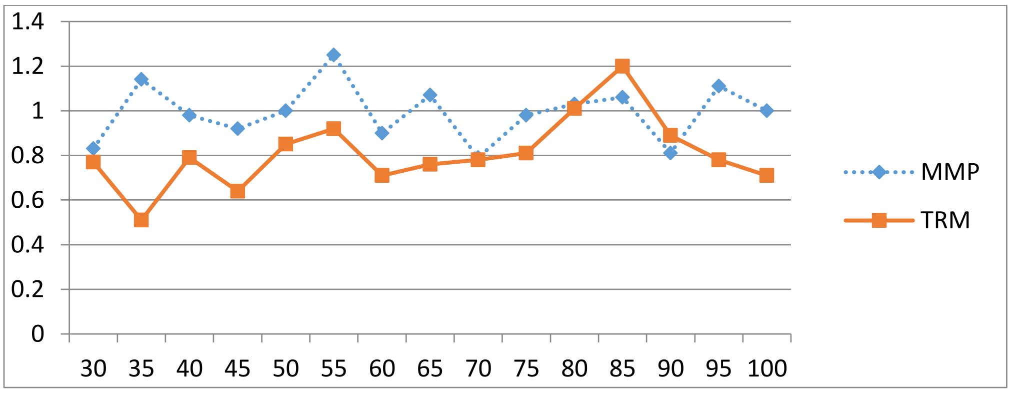

Figure 12. CPU times of the mathematical models for two-machine cells.

Comparing the CPU times of TRM and PMM for the two-machine cell problems demonstrates that the performance of the TRM is significantly better than the PMM.

Table 11, similar to Table 10, presents the process of solving the three-machine cell problems by using both the proposed models.

Table 11. The results of the mathematical models for three-machine cells.

Processing times	Objective function	Lower Bound	*CPU* time (in seconds)	
			PMM	*TRM*
30	120	180	395.90	124.30
35	120	195	160.90	63.16
40	120	210	33.39	49.49
45	120	225	41.68	20.13
50	120	240	9.26	20.80
55	130	256	109.90	8.53
60	140	276	42.17	7.89
65	150	296	37.74	12.20
70	160	316	17.50	13.60
75	170	336	17.69	11.64
80	180	356	25.87	14.44
85	190	376	70.51	14.87
90	200	396	44.94	11.54
95	210	416	46.50	31.79
100	220	436	33.09	16.30

Table 11 shows that when the processing times are less than 50, the objective functions are the same and equal to 120; and by increasing each five units to the processing times, the objective functions are increased by 10 units. To obtain a better comparison between the models for the three-machine cell problems, the CPU times of solving the problems are plotted in Figure 13.

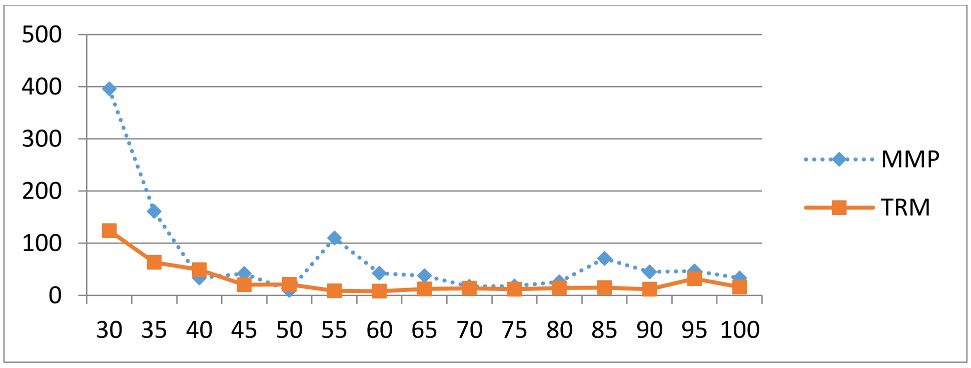

Figure 13. CPU times of the mathematical models for three-machine cells.

The graph of PMM demonstrates that for small processing times, the CPU times are very large but by increasing the processing times the CPU times decrease significantly and fluctuate around 40.7 s. The graph of TRM shows a more logical trend in which the CPU times for small processing times are large but by increasing the processing times the CPU times decrease gradually and fluctuate around 14.5 s. Considering Figure 13, it can be seen that for the three-machine cell problems TRM is more efficient than PMM in most of the cases.

4.6 A lower bound explained by an assignment problem

In a cycle, the minimal total time of the robot movements provides a lower bound for the cycle time. The cycle time is longer than the total movement time if and only if there is at least one positive waiting time for the robot. The presence of the positive waiting time depends on the length of the processing time.

Theorem 5. The minimum cycle time to produce *2*m parts when each machine produces two parts is $(4m^2 + 4m)\delta + 8m\varepsilon$.

Proof: When the total movement time of the robot is determined, all waiting times can be considered zero. Table 12 shows the time distance matrix for every two activities in an m-machine cell.

Table 12. Time distance matrix for an *FMC* when each machine has an input buffer.

j \ k	L1	L2	…	Lm-1	Lm	U1	U2	…	Um-1	Um
L1	$2\delta+2\varepsilon$	$3\delta+2\varepsilon$	…	$m\delta+2\varepsilon$	$(m+1)\delta+2\varepsilon$	$m\delta+2\varepsilon+p$	$m\delta+2\varepsilon+w_2$	…	$m\delta+2\varepsilon+w_{m-1}$	$m\delta+2\varepsilon+w_m$
L2	$3\delta+2\varepsilon$	$4\delta+2\varepsilon$	…	$(m+1)\delta+2\varepsilon$	$(m+2)\delta+2\varepsilon$	$(m+1)\delta+2\varepsilon+w_1$	$(m-1)\delta+2\varepsilon+p$	…	$(m-1)\delta+2\varepsilon+w_{m-1}$	$(m-1)\delta+2\varepsilon+w_m$
.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.
Lm-1	$m\delta+2\varepsilon$	$(m+1)\delta+2\varepsilon$	…	$(2m-2)\delta+2\varepsilon$	$(2m-1)\delta+2\varepsilon$	$(2m-2)\delta+2\varepsilon+w_1$	$(2m-4)\delta+2\varepsilon+w_2$	…	$2\delta+2\varepsilon+p$	$2\delta+2\varepsilon+w_m$
Lm	$(m+1)\delta+2\varepsilon$	$(m+2)\delta+2\varepsilon$	…	$(2m-1)\delta+2\varepsilon$	$(2m)\delta+2\varepsilon$	$(2m-1)\delta+2\varepsilon+w_1$	$(2m-3)\delta+2\varepsilon+w_2$	…	$3\delta+2\varepsilon+w_{m-1}$	$\delta+2\varepsilon+p$
U1	$(m+2)\delta+2\varepsilon$	$(m+3)\delta+2\varepsilon$	…	$(2m)\delta+2\varepsilon$	$(2m+1)\delta+2\varepsilon$	$(2m)\delta+2\varepsilon+w_1$	$(2m-2)\delta+2\varepsilon+w_2$	…	$4\delta+2\varepsilon+w_{m-1}$	$2\delta+2\varepsilon+w_m$
U2	$(m+2)\delta+2\varepsilon$	$(m+3)\delta+2\varepsilon$	…	$(2m)\delta+2\varepsilon$	$(2m+1)\delta+2\varepsilon$	$(2m)\delta+2\varepsilon+w_1$	$(2m-2)\delta+2\varepsilon+w_2$	…	$4\delta+2\varepsilon+w_{m-1}$	$2\delta+2\varepsilon+w_m$
.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.
Um-1	$(m+2)\delta+2\varepsilon$	$(m+3)\delta+2\varepsilon$	…	$(2m)\delta+2\varepsilon$	$(2m+1)\delta+2\varepsilon$	$(2m)\delta+2\varepsilon+w_1$	$(2m-2)\delta+2\varepsilon+w_2$	…	$4\delta+2\varepsilon+w_{m-1}$	$2\delta+2\varepsilon+w_m$
Um	$(m+2)\delta+2\varepsilon$	$(m+3)\delta+2\varepsilon$	…	$(2m)\delta+2\varepsilon$	$(2m+1)\delta+2\varepsilon$	$(2m)\delta+2\varepsilon+w_1$	$(2m-2)\delta+2\varepsilon+w_2$	…	$4\delta+2\varepsilon+w_{m-1}$	$2\delta+2\varepsilon+w_m$

Since each cell in Table 12 includes 2ε for the loading/unloading activities, if we decrease each cell by 2ε, the optimal robot movement sequence remains the same. Then, every cell in the table will be a multiple of δ (see Table 13). Therefore, it should be noted that for those cells that include processing times, p_k will be changed to $\frac{p_k}{\delta}$ to be a multiple of δ. Moreover, since each machine produces two parts in a cycle, there are two loading and two unloading activities for each machine. Thus, there are four rows and columns for each machine in which the first row or column is related to the first part which is processed by machine k.

Table 13. Time distance matrix in terms of the δ coefficient.

ij	L1	…	Lk	Lk	…	Lm	U1	…	Uk	Uk	…	Um
L1	-	…	1+k	1+k	…	1+m	$m+\frac{p_1}{\delta}$	…	m	m	…	m
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
Lj	j+1	…	-	j+k	…	j+m	j+m-1	…	|j-k|+m-k+1	|j-k|+m-k+1	…	m-j+1
Lj	j+1	…	j+k	-	…	j+m	j+m-1	…	|j-k|+m-k+1	|j-k|+m-k+1	…	m-j+1
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
Lk	k+1	…	-	2k	…	k+m	k+m-1	…	$m-k+1+\frac{p_k}{\delta}$	m-k+1	…	m-k+1
Lk	k+1	…	2k	-	…	k+m	k+m-1	…	m-k+1	$m-k+1+\frac{p_k}{\delta}$	…	m-k+1
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
Lm	m+1	…	m+k	m+k	…	-	2m-1	…	2(m-k)+1	2(m-k)+1	…	$1+\frac{p_m}{\delta}$
U1	m+2	…	m+k+1	m+k+1	…	2m+1	-	…	2(m-k+1)	2(m-k+1)	…	2
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
Uj	m+2	…	m+k+1	m+k+1	…	2m+1	2m	…	2(m-k+1)	2(m-k+1)	…	2
Uj	m+2	…	m+k+1	m+k+1	…	2m+1	2m	…	2(m-k+1)	2(m-k+1)	…	2
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
Uk	m+2	…	m+k+1	m+k+1	…	2m+1	2m	…	-	2(m-k+1)	…	2
Uk	m+2	…	m+k+1	m+k+1	…	2m+1	2m	…	2(m-k+1)	-	…	2
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
Um	m+2	…	m+k+1	m+k+1	…	2m+1	2m	…	2(m-k+1)	2(m-k+1)	…	-

Each feasible solution has an element in every row and column. To find the optimal solution, an assignment problem is solved as shown in Table 13. The dual variables of this problem are shown in Table 14 (see the row and the column of "Dual coefficient"). It must be shown that these dual variables are feasible in the dual of the assignment problem.

Table 14. Distance matrix in terms of the δ coefficient and dual solutions.

ij	L1	…	Lk	Lk	…	Lm	U1	…	Uk	Uk	…	Um	Dual
L1	-	…	1+k	1+k	…	1+m	$m+\frac{p_1}{\delta}$	…	m	m	…	m	2
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	…
Lj	j+1	…	j+k	j+k	…	j+m	j+m-1	…	|j-k|+m-k+1	|j-k|+m-k+1	…	m-j+1	j+1
Lj	j+1	…	j+k	j+k	…	j+m	j+m-1	…	|j-k|+m-k+1	|j-k|+m-k+1	…	m-j+1	j+1
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	
Lk	k+1	…	-	2k	…	k+m	k+m-1	…	$m-k+1+\frac{p_k}{\delta}$	m-k+1	…	m-k+1	k+1
Lk	k+1	…	2k	-	…	k+m	k+m-1	…	m-k+1	$m-k+1+\frac{p_k}{\delta}$	…	m-k+1	k+1
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	
Lm	m+1	…	m+k	m+k	…	-	2m-1	…	2(m-k)+1	2(m-k)+1	…	$1+\frac{p_m}{\delta}$	m+1
U1	m+2	…	m+k+1	m+k+1	…	2m+1	-	…	2(m-k+1)	2(m-k+1)	…	2	m+2
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	
Uj	m+2	…	m+k+1	m+k+1	…	2m+1	2m	…	2(m-k+1)	2(m-k+1)	…	2	m+2
Uj	m+2	…	m+k+1	m+k+1	…	2m+1	2m	…	2(m-k+1)	2(m-k+1)	…	2	m+2
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	
Uk	m+2	…	m+k+1	m+k+1	…	2m+1	2m	…	-	2(m-k+1)	…	2	m+2
Uk	m+2	…	m+k+1	m+k+1	…	2m+1	2m	…	2(m-k+1)	-	…	2	m+2
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	
Um	m+2	…	m+k+1	m+k+1	…	2m+1	2m	…	2(m-k+1)	2(m-k+1)	…	-	m+2
	0	…	k-1	k-1	…	m-1	m-2	…	m-2k	m-2k	…	-m	

To examine the feasibility in the dual problem, each coefficient in the matrix of the assignment problem must be greater than or equal to the sum of the dual variables of its row and column. When a variable of the assignment problem is 1, if its dual variables are feasible in the dual problem and there is a feasible solution for the assignment problem, the related constraint of the dual problem is satisfied by the following equations and both of the primal and dual solutions are optimal. The sums of the dual variables are as follows:

L_jL_k: $(j+1) + (k-1) = j+k$

L_kL_k: $(k+1) + (k-1) = 2k$

L_jU_k: $(j+1) + (m-2k) = m+j-2k+1$

L_kU_k: $(k+1) + (m–2k) = m–k+1$

U_jL_k: $(m+2) + (k–1) = m+k+1$

U_kL_k: $(m+2) + (k–1) = m+k+1$

U_jU_k: $(m+2) + (m–2k) = 2m–2k+2$

U_kU_k: $(m+2) + (m–2k) = 2m–2k+2$

Therefore, to examine all of these feasibilities, the following equations must be evaluated with their conjunctions of their rows and columns of Table 14:

L_jL_k: $j+k \leq j+k$

L_kL_k: $2k \leq 2k$

L_jU_k: $m+j–2k +1 \leq |j–k|+m–k+1$

There are two different cases as $j > k$ and $j < k$. If $j > k$, then $m+j–2k+1 \leq |j–k|+m–k+1 = j–k+m–k+1 = m+j–2k+1$. If $j < k$, then $m+j–2k+1 \leq |j–k|+m–k+1 = k–j+m–k+1 = m–j+1$. By eliminating $m+1$ from both sides of the inequality and taking all the parameters to one side, it will be simplified as $0 \leq 2(k-j)$; and as k is greater than j, this inequality is always true. Therefore,

L_kU_k: $m–k+1 \leq m–k+1$ and also $m-k+1 \leq m-k+1+\frac{p_k}{\delta}$. Considering every value of p in the second case, both the inequalities are correct:

U_jL_k: $m+k+1 \leq m+k+1$

U_kL_k: $m+k+1 \leq m+k+1$

U_jU_k: $2m–2k+2 = 2(m–k+1)$

U_kU_k: $2m–2k+2 = 2(m–k+1)$

If the elements of Table 14 are reduced by the row and column of dual variables, then it becomes Table 15.

Table 15. The last table for the assignment problem.

Ij	L1	L1	…	Lk	Lk	…	Lm	U1	…	Uk	Uk	…	Um
L1	-	$\boxed{0}$	…	0	0	…	0	$\frac{p_1}{\delta}$	…	2(k-1)	2(k-1)	…	2(m-1)
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
Lj	0	0	…	0	0	…	0	0	…	2(k-j)	2(k-j)	…	2(m-j)
Lj	0	0	…	0	0	…	0	0	…	2(k-j)	2(k-j)	…	2(m-j)
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
Lk	0	0	…	-	$\boxed{0}$	…	0	0	…	$\frac{p_k}{\delta}$	0	…	2(m-k)
Lk	0	0	…	0	-	…	0	0	…	0	$\frac{p_k}{\delta}$	…	2(m-k)
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
Lm	0	0	…	0	0	…	-	$\boxed{0}$	…	0	0	…	$\frac{p_m}{\delta}$
U1	0	0	…	0	0	…	0	-	…	0	0	…	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
Uj	0	0	…	0	0	…	0	0	…	0	0	…	0
Uj	0	0	…	0	0	…	0	0	…	0	0	…	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
Uk	0	0	…	0	0	…	0	0	…	-	$\boxed{0}$	…	0
Uk	0	0	…	0	0	…	0	0	…	0	-	…	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
Um	0	0	…	0	0	…	0	0	…	0	0	…	$\boxed{0}$
Um	$\boxed{0}$	0	…	0	0	…	0	0	…	0	0	…	-

In Table 15, the rectangles indicate an optimal solution of the assignment problem. Therefore, considering the related δ coefficients of the rectangles in Table 14, the total time of producing $2m$ parts in a cycle using an FMC can be calculated as follows:

$$T = 8m\varepsilon + \left(\sum_{k=2}^{2m} k + (2m-1) + (2m) + 2\sum_{k=0}^{m} 2(m-k+1) + (m+2)\right)\delta.$$

Since $\sum_{k=2}^{2m} k = 2m^2 + m - 1$and $\sum_{k=0}^{m} 2(m-k+1) = m^2 - m$, the cycle time can be calculated as follows:

$T = 8m\varepsilon + \left((2m^2 + m - 1) + (2m - 1) + (2m) + (2m^2 - 2m) + (m + 2)\right)\delta =$

$8m\varepsilon + (4m^2 + 4m)\delta.$ □

4.7 Optimal cycles of different structures in a general case

The optimal solution of the assignment problem, which is the minimal total time, the robot moves in a cycle, provides a lower bound for the minimal cycle time. Table 15 shows that there are numerous different optimal solutions for the assignment problem. If the processing times are such that all waiting times are zero, then the lower bound is equal to the minimal cycle time. Otherwise, the minimal cycle time becomes significantly greater than the lower bound. To find the optimal solutions of the problem, first we solve the problem when the processing times are small enough, i.e., which can be neglected from the calculations (P = 0). Then, we find the solution, and under that condition we maximize p such that all of the waiting times (w_i) are still zero.

4.7.1 Return time

The *return time* to a machine is the time between loading the machine and the moment that the robot returns to the same machine to unload it. The total working time of the robot during a return time is a lower bound of the return time. It is possible that some of the waiting times during the return time can be positive. The total working times of the robot contain both transportations and the loading/unloading operations. Therefore, the return time can be reduced by calculating the working times of the robot. The waiting time on machine k is zero if the return time is longer than its processing time. In this study, we have considered two return times for each machine because each machine is loaded twice. For example, assume that a cycle has an order like $L_1L_1...L_mL_mU_1U_1...U_mU_m$. Then, the first return time to machine *k* is the time from machine *k*, just after loading machine *k* for the first time, and passing the route of $L_k...L_mL_mU_1U_1...U_{k-1}$ until the robot reaches besides machine *k* to unload it for the first time. To calculate this return time, all the robot movements are divided into *loaded movements*, when the robot is carrying a part, and *unloaded movements*, when the robot is moving when it is empty. The δ coefficients of all the robot movements for return to machine *k* are provided in Table 16.

Table 16. Loaded and unloaded robot movement times for machine *k.*

Activity	L_k	L_{k+1}	L_{k+1}	…	L_{m-1}	L_m	L_m	U_1	U_1	…	U_{k-1}	U_k
Unloaded movement	k	k	k+1	…	m-1	m-1	m	m-1	m	…	m-k+2	m-k+1
Loaded movement	k	k+1	k+1	…	m-1	m	m	m	m	…	m-k+2	

The times related to the unloaded robot movements are the summation of the second row of Table 16, and it is calculated as follows:

$$2\left(\sum_{j=k}^{m} j\right) + (m-1) + 2\left(\sum_{j=m-k+2}^{m-1} j\right) + (m-k+1) = (m^2 + m - k^2 + k) + (m-1) + (2mk - k^2 - 4m + 3k - 2) + (m-k+1) = m^2 - m - 2k^2 + 2km + 3k - 2.$$

The times related to the loaded robot movements are the summation of the third row of Table 16, and it is calculated as follows:

$$k + 2\left(\sum_{j=k+1}^{m} j\right) + 2\left(\sum_{j=m-k+2}^{m} j\right) = k + (m^2 + m - k^2 - k) + (2km - k^2 - 2m + 3k - 2) = m^2 - m - 2k^2 + 2mk + 3k - 2.$$

Therefore, the first return time of the robot for $L_k...L_mL_mU_1U_1...U_k$ is equal to $2(m^2-2k^2+2km-m+3k-2)\delta+2(2m-1)\varepsilon$. The properties of the δ coefficient are as follows:

1- It is a quadratic function in terms of k.

2- Its maximum point is equal to $\frac{2m+3}{4}$ and the maximum value of this function is $(3m^2+m-\frac{7}{4})\delta+2(2\text{m}-1)\varepsilon$.

3- It is symmetric to the vertical line of $k=\frac{2m+3}{4}$.

4- The minimal value of this function in the [1, m] interval is $2\text{m}^2+2\text{m}-2$.

To calculate the second return time to machine k, the loaded and the unloaded robot movements of activity L_k must be neglected from Table 16, and the loaded robot movement of the first U_k and the unladed robot movement of the second U_k must be added (see Table 17).

Table 17. Loaded and unloaded robot movement times for $L_{k+1}...L_mL_mU_1U_1...U_kU_k$.

Activity	L_{k+1}	L_{k+1}	…	L_{m-1}	L_{m-1}	L_m	L_m	U_1	U_1	…	U_{k-1}	U_k	U_k
Unloaded movement	k	k+1	…	m-2	m-1	m-1	m	m-1	m	…	m-k+2	m-k+1	m-k+1
Loaded movement	k+1	k+1	…	m-1	m-1	m	m	m	m	…	m-k+2	m-k+1	

As it is obvious from Table 17, by adding $2(m-k+1)\delta$ to the total robot movement time of the first return time to machine k and subtracting $2k\delta$ from it, the total robot movement time of the second return time to machine k is as follows:

$2(m^2 - 2k^2 + 2km - m + 3k - 2)\delta + 2(2m - 1)\varepsilon + (2(m - k + 1)\delta - 2k\delta) =$

$2(m^2 - 2k^2 + 2km + k - 1)\delta + 2(2m - 1)\varepsilon.$

The properties of the δ coefficient for the second return time to machine k are as follows:

1- It is a quadratic function of k.

2- Its maximum point is equal to $\frac{2m+1}{4}$ and its maximum value is the same as the first return time to machine k.

3- It is symmetric to the vertical line of $k = \frac{2m+1}{4}$.

4- The minimal value of this function in the [1,m] interval is the same as the minimum value of the first return time to machine k.

A comparison between the first and the second return time to machine k shows that the first return time to machine k is shorter if $k < \frac{m+1}{2}$. On the other hand, these calculations show that the following lemma is true.

Lemma 4. In the $L_1L_1...L_mL_mU_1U_1...U_mU_m$ cycle, the minimal return time is at least $2(m^2 + m - 1)\delta + 2(2m - 1)\varepsilon$.

Proof. The first return time to machine 1 and the second return time to machine m are the same and equal to $2(m^2 + m - 1)\delta + 2(2m - 1)\varepsilon$. If there are positive waiting times, then the minimal return time can be larger but not shorter. □

Assume that p_k is the processing time of machine k to process the parts. The first and the second return time define the lemma as follows:

Lemma 5. If for each machine k in which $k < \frac{m+1}{2}, p_k \leq 2(m^2 - 2k^2 + 2km - m + 3k - 2)\delta + 2(2m-1)\varepsilon$, then all waiting times are zero. Also, if for each machine k in which $k \geq \frac{m+1}{2}$, $p_k \leq 2(m^2 - 2k^2 + 2km + k - 1)\delta + 2(2m-1)\varepsilon$, then all waiting times are zero.

Corollary. In an FMC with identical parallel machines, if $p \leq 2(m^2 + m - 1)\,\delta + 2(2m-1)\varepsilon$, then all waiting times are zero.

Proof. The statement is a direct consequence of the last two lemmas. □

4.7.2 Optimal cycles

All loaded robot movements contain the same elements: taking a part from the input buffer and loading machine k which needs $k\delta+2\varepsilon$ time units, unloading it, and putting it into the output buffer which needs $(m+1-k)\delta+2\varepsilon$; and the total time is the same and equal to $(m+1)\delta+4\varepsilon$ time units for each part. Therefore, to minimize the working times of the robot, only the unloaded robot movements should be minimized. For this reason, it is sufficient to determine the optimal solution of the assignment problem defined by the matrix consisting of the distances between two machines as a unit (δ). Table 18 provides the distances in terms of the δ coefficient for only unloaded robot movements.

Table 18. Matrix of the δ coefficient for unloaded robot movements.

k	L1	…	Li	Li	…	Lm	U1	…	Uk	Uk	…	Um
L1	-	…	1	1	…	1	0	…	k-1	k-1	…	m-1
L1	1	…	1	1	…	1	0	…	k-1	k-1	…	m-1
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
Li	i	…	-	i	…	i	i-1	…	|k-i|	|k-i|	…	m-i
Li	i	…	i	-	…	i	i-1	…	|k-i|	|k-i|	…	m-i
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
Lk	k	…	k	k	…	k	k-1	…	0	0	…	m-k
Lk	k	…	k	k	…	k	k-1	…	0	0	…	m-k
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
Lm	m	…	m	m	…	-	m-1	…	m-k	m-k	…	0
U1	m+1	…	m+1	m+1	…	m+1	-	…	m-k+1	m-k+1	…	1
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
Ui	m+1	…	m+1	m+1	…	m+1	m	…	m-k+1	m-k+1	…	1
Ui	m+1	…	m+1	m+1	…	m+1	m	…	m-k+1	m-k+1	…	1
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
Uk	m+1	…	m+1	m+1	…	m+1	m	…	-	m-k+1	…	1
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
Um	m+1	…	m+1	m+1	…	m+1	m	…	m-k+1	m-k+1	…	-

If we assume that $p_i = 0$, then we can obtain the optimal solution by minimizing the assignment problem. There are a number of optimal solutions for $p_i = 0$. Some of the solutions are as follows:

1. According to Table 18, the total time for the unloaded robot movement cycle like $L_1L_1...L_mL_mU_1U_1...U_mU_m$ in terms of the δ coefficient is equal to $\left(2(\sum_{k=1}^{m} k) - 1 + 2(\sum_{k=1}^{m}(m-k+1)) - m + (m+1)\right)\delta = (2m^2 + 2m)\delta$.

2. The cycle that contains any order of loading activities like L_iL_j... L_m at first and continues by any order of unloading activities that will be started with U_1.

3. Any order of the robot movement cycle which includes loading and unloading activities of each machine consecutively as $L_iU_iL_jU_j\ldots L_mU_m$.

4.7.3 Comparison between three similar cycles

Let's consider the $L_1U_1L_1U_1L_2U_2\ldots L_mU_m$ cycle in which all the machines are empty at the beginning of the cycle (the first case). The other cycle is $U_1L_1U_1L_1U_2L_2\ldots U_mL_m$ with two different situations. In the first situation, each machine has been loaded two times in the previous cycle without any unloading between them until the end of the cycle (the second case). In the second situation, the cycle is started when there is only one processed part in each machine (the third case).

In the first case, all of the data related to loaded/unloaded robot movements and the waiting times of the robot are presented in Table 19. It should be noted that the loading/unloading times of the robot are considered in the calculations.

Table 19. Robot movement times for the $L_1U_1L_1U_1L_2U_2\ldots L_mU_m$ cycle.

Activity	U_1	L_1	U_1	L_2	U_2	…	L_k	U_k	…	L_m	U_m	L_1
Unloaded movement	-	m+1	-	m+1	-	…	m+1	-	…	m+1	-	m+1
Robot waiting time	p_1	-	P_1	-	P_2	…	-	p_k	…	-	p_m	-
Loaded movement	m	1	m	2	m-1	…	k	m-k+1	…	m	1	1

The summations of the unloaded and loaded robot movement times in terms of the δ coefficient are equal to $2m(m+1)$ and $2m(m+1)$, respectively, from Table 19. By adding $8m\varepsilon$ for the loading and unloading of the machines, the total cycle time is equal to $(4m^2+4m)\delta + 2(\sum_{i=1}^{m} p_i) + 8m\varepsilon$, which is similar to the result of Theorem 5, considering $p_i > 0$. Generally, this cycle is considered as $L_kU_kL_kU_k\ldots L_mU_mL_1U_1\ldots L_{k-1}U_{k-1}L_{k-1}U_{k-1}$.

In the second case, at the beginning of the cycle each machine has one processed part that is ready to unload and one unprocessed part in its input buffer. In this case, we assume that the processing time is less than the return time for each machine. Table 20 shows the data of this cycle.

Table 20. Cycle times related to the $U_1L_1U_1L_1U_2L_2 \ldots U_mL_m$ cycle.

Activity	L_1	U_1	L_1	U_2	L_2	U_2	L_2	…	U_m	L_m	U_m	L_m	U_1
Unloaded movement	m+1	-	m+1	1	m+1	-	m+1	…	1	m+1	-	m+1	m-1
Loaded movement	1	m	1	m-1	2	m-1	2	…	1	m	1	m	m

Table 20 shows that the unloaded and loaded robot movements take $2(m^2 + 2m - 1)\delta$ and $2(m^2 + m)\delta$ time units, respectively, where the robot needs $8m\varepsilon$ time units to load and unload all of the machines. To reach to the initial state, from where the cycle was started, when the robot is ready to unload machine 1, the processing time of machine 1 should be considered. Thus, all of the calculated times are considered as the return time to machine 1 that is equal to $2(2m^2 + 3m - 1)\delta + 8m\varepsilon$. Considering this return time, the cycle time of this case will become $\max\{2p_i, 2(2m^2 + 3m - 1)\delta + 8m\varepsilon\}$.

In the third case, in addition to the loaded and unloaded robot movements, the waiting time of the robot for the second unloading of the machines must be considered. Table 21 shows the data for calculating the cycle time in this case.

Table 21. Cycle times of $U_1L_1U_1L_1U_2L_2 \ldots U_mL_m$ cycle considering waiting times.

Activity	L_1	U_1	L_1	…	U_k	L_k	U_k	L_k	…	U_m	L_m	U_1
Unloaded movement	m+1	-	m+1	…	1	m+1	-	m+1	…	-	m+1	m-1
Robot waiting time	-	w_1	-	…	-	-	w_k	-	…	w_m	-	-
Loaded movement	1	m	1	…	m-k+1	k	m-k+1	k	…	1	m	m

Considering Table 21 and following the similar procedure to calculate the cycle time show that the total cycle time in this case is equal to $2(2m^2+3m-1)\delta+\sum_{i=1}^{m} w_i+8m\varepsilon$.

A comparison between the cycle time for the first and the second case shows that if $\sum_{i=1}^{m} p_i \geq (m-1)\delta$, $\forall i$, then the cycle time of the second case is always shorter; otherwise, the cycle time of the first case is shorter. The cycle time of the third case is greater than the cycle time of the second case unless all waiting times are equal to zero or the processing times are large enough. Also, a comparison between the first and the third cycle time shows that if $\sum_{i=1}^{m} p_i \geq (m-1)\delta+\frac{\sum_{i=1}^{m} w_i}{2}$, then the cycle time of the third case is less than the cycle time of the first one.

Chapter 5

DEVELOPED METAHEURISTIC ALGORITHM

5.1 Preface

Since the robotic flexible cell problems belong to NP-hard class of problems, optimizer software like CPLEX can only find the optimal solutions of small-size problems [29, 30]. Therefore, in order to solve large-size problems, metaheuristic algorithms are used. A simulated annealing algorithm is proposed for the problems with no individual buffers for each machine.

5.2 Representation

In our study, a solution is presented by an array having *2m* elements in which the numbers 1 to *m* correspond to the loading of the first machine to the m^{th} machine and the numbers *m+1* to *2m* correspond to the unloading of the first machine to the m^{th} machine, respectively. To prevent permutations, the first element of each array is always 1. For example, a four-machine flexible cell having a cycle $L_1L_3L_4U_2U_3U_1U_4L_2L_1$ is presented in Figure 14.

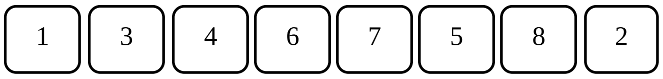

Figure 14. Presentation of $L_1L_3L_4U_2U_3U_1U_4L_2L_1$ cycle for a four-machine cell.

In this array, 1 is the first element and depicts L_1. L_3 and L_4 are shown by 3 and 4 respectively. Since in this example $m = 4$, all unloading activities are shown by 5 to

8, U_2 and U_3 are shown by 6 and 7, respectively. The same procedure is applied to demonstrate the rest of this array.

5.3 Initial solution

An initial solution can be obtained by generating a random solution or a reasonable solution; for the latter, there are two different methods. The first method is the cycle of $L_1L_2...L_mU_1U_2...U_mL_1$ (its array is shown in Figure 15 and discussed in Section 5.2).

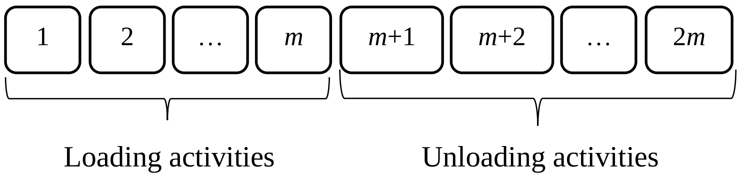

Figure 15. Array of $L_1L_2...L_mU_1U_2...U_mL_1$ cycle.

The second method is based on an iterative construction. For this, after loading the first machine, the activity that has the lowest robot moving time from the previously assigned activity among the remaining activities is selected until the cycle is complete.

After some preliminary experiments and testing different initial solutions in practice, to avoid considering a unique initial solution as the starting point of the algorithms, we randomly generated the initial solution. Generating random solutions help start from different initial solutions in each run that avoids entrapment in a local optimum.

5.4 Computing cycle time for a given solution

In finding of the objective value of each array, all of the parameters, except waiting times, are known. Let's consider a cycle $L_1L_3L_4U_2U_3U_1L_2U_4L_1$, assuming that loading/unloading times are 1, the robot move time between a pair of two consecutive machines is 2 and all processing times are the same and equal to 80 time units. To calculate the cycle time in this case, the duration time between a set of two activities is calculated by using d_{ab} formula in Section 3.4. For example, the duration time formula for L_1L_3 is $2\varepsilon + 4\delta,$ which is equal to 10 time units. By applying the same procedure, duration times of L_3L_4, L_4U_2, U_2U_3, U_3U_1, U_1L_2, L_2U_4 and U_4L_1 are calculated as 16, 12, 10, 18, 16, 8 and 14, respectively.

To compute waiting times, return time to each machine must be compared with its corresponding processing time. If the processing time is greater than the return time, the waiting time is positive, otherwise, it is equal to zero. There are two cases that should be considered; if loading of a machine occurs before unloading of the same machine in a cycle, like machine 1 in this example, the summation of the duration between a set of two activities from loading a machine to return to the same machine to unload it, in addition to the potential waiting times, should be considered. For example, the return time to machine 1 is the summation of the times from L_1 to U_1, which are $L_1L_3 + L_3L_4 + L_4U_2 + U_2U_3 + U_3U_1 = 66$, plus potential robot waiting time on machine 2 and machine 3. Since as mentioned in Section 3.4, before each unloading activity, a positive robot waiting time may exist. Consequently, the robot waiting time on machine 1 is equal to $max\{0, p_1 - (66 + w_2 + w_3)\}$.

On the other hand, if loading a machine is done after unloading the same machine in a cycle, like machine 2 in our example, the return time to the machine is the sum of duration from loading that machine to the end of the cycle, i.e., the loading of machine 1, and from loading machine 1 to return to the same machine to unload it in the next cycle. Obviously, potential waiting times should also be considered if there is any. For example, to calculate the return time to machine 2, at first, two cycles can be considered consecutively (see Figure 16). Then, the duration from the first loading of machine 2 to its unloading plus w_4 is the return time to machine 2, which is equal to $60 + w_4$. Therefore, $w_2 = max\{0, p_2 - (60 + w_4)\}$.

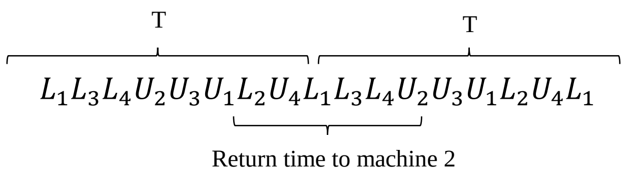

Figure 16. Return time to machine 2 in *L₁L₃L₄U₂U₃U₁L₂U₄L₁* cycle.

Applying the same procedure leads us to reach to the waiting times for machines 3 and 4 equal to $w_3 = max\{0, p_3 - (38 + w_2)\}$ and $w_4 = max\{0, p_4 - (64 + w_1 + w_2 + w_3)\}$ respectively. Thus, $T = 16 + 12 + 10 + 18 + 16 + 8 + 14 + w_1 + w_2 + w_3 + w_4 = 94 + w_1 + w_2 + w_3 + w_4$.

Since the aim is to minimize cycle time, the minimal amount of the total waiting times must be computed. On the other hand, increasing in any waiting time will affect other waiting times that are related to. Therefore, a linear programming model should be solved to minimize the cycle time. The linear programming model of our example is as follows:

$Minimize\ T = 94 + w_1 + w_2 + w_3 + w_4$

Subject to:

$w_1 \geq p_1 - (66 + w_2 + w_3)$

$w_2 \geq p_2 - (60 + w_4)$

$w_3 \geq p_3 - (38 + w_2)$

$w_4 \geq p_4 - (64 + w_1 + w_2 + w_3)$

$w_i \geq 0, i = 1, 2, 3, 4$

where all the processing times are known and must be substituted before solving the model.

5.5 Generating the next solution

To generate a new solution and to select a solution for the next generation, three different methods, based on local search algorithm, are considered. The algorithm generates neighborhood solutions and tries to find the local optimums, iteratively. Since this method accepts only a better solution as the current solution, it always modifies the last improved solution [31]. In this study, three different operators that are shift, swap, and reverse are employed. Figure 17 shows the shift operator for the robot move sequence in a four-machine cell, which removes one of the activities from the sequence and moves it at another place in the sequence, randomly (see Figure 17).

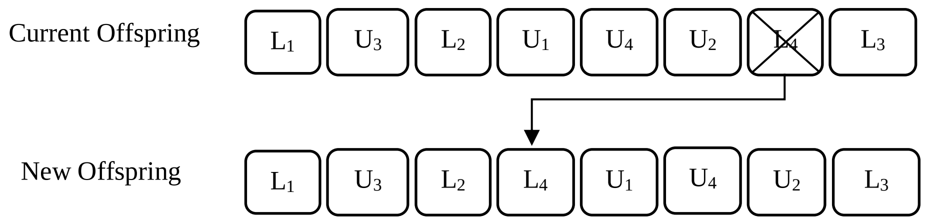

Figure 17. Shift operator.

The swap operator finds two activities randomly and switches them together (see Figure 18).

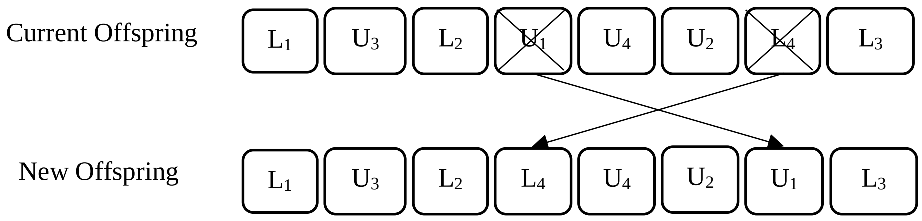

Figure 18. Swap operator.

The reverse operator selects two activities, which are not next to each other, and reverse all the activities among them as it can be seen in Figure 19.

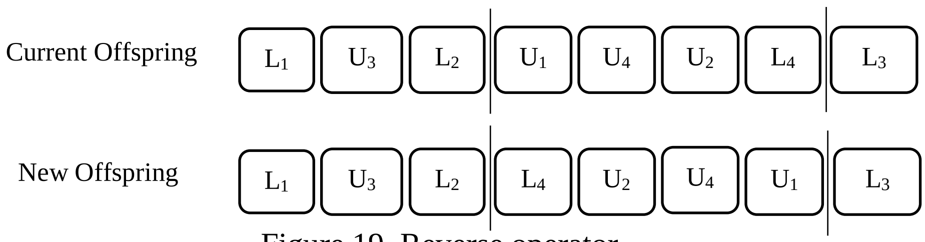

Figure 19. Reverse operator.

In each iteration of the developed SAA, three new solutions are generated from the current solution by using each of the shift, swap and reverse operators. The objective function of each is computed, and the best of these three new solutions is selected as the generated candidate neighboring solution. This solution is adopted as the next current solution if it is better than the current solution or it is accepted by using the acceptance probability function.

5.6 Cooling

The geometric cooling, which is the most common cooling method, is applied. According to this method, the developed SAA starts its search at an initial temperature. Then, after each iteration, a certain percent of T is counted as the value of T for the next iteration, i.e., $T=\alpha*T$ where $0<\alpha<1$.

5.7 Stopping criteria

Generally, if an approximate solution found by a metaheuristic algorithm is accurate enough, the iterative method should be stopped. In this thesis, for small-size problems that we know the optimal cycle time, when an optimal solution is found by the method the solving procedure is finished. For the large-size problems which the optimal cycle times are unknown for them, a limit on the solution time is used as the stopping criterion. The starting time is kept and the total time from the starting time to the current time is computed after each iteration. When it exceeds the limit, the search is stopped. The pseudocode of the developed simulated annealing algorithm is given below:

```
Step 0:  Set values of the parameters $T$, Time Limit, $\alpha$, $m$, $p$, $\varepsilon$ and $\delta$. Record the time:
         Tstart.
Step 1:  Generate the Initial Solution.
         Current Solution = Initial Solution. Best Solution = Current Solution.
Step 2:  Compute the cycle time of the Current Solution: F[Current Solution].
         F[Best Solution]=F[Current Solution]
Step 3:    a) Generate a neighboring solution of the current solution by using the swap
              mechanism.
           b) Generate a neighboring solution of the current solution by using the shift
              mechanism.
           c) Generate a neighboring solution of the current solution by using the
              reverse mechanism.
           d) Compute the cycle time of these neighboring solutions and select the best
              of them as the candidate solution.
Step 4:  If
         F[Candidate Solution]<F[Current Solution] or
         Rand(0,1) < Exp(-(F[Candidate Solution]-F[Current Solution])/T)
         then
         Current Solution = Candidate Solution and
         F[Current Solution]=F[Candidate Solution]
Step 5:  If
         F[Current Solution]<F[Best Solution]
         then
         Best Solution = Current Solution and
         F[Best Solution]=F[Current Solution]
Step 6:  Record the current time: Tnow.
         If the duration from Tstart to Tnow is more than Time Limit, then STOP and
         present the Best Solution, otherwise $T = \alpha * T$ and go to Step 3.
```

5.8 Experimental results

To evaluate the proposed metaheuristic algorithm, several problems are considered and executed on an Intel(*R*) Pentium(*R*) Dual *CPU E*2180 @ 2.00 *GHz CPU* with 2.0 *GB* of *RAM*. The algorithm are coded in *MATLAB R*2013a software. The objective functions (OBF) and the CPU times, are the average results of executions carried out 10 times for each problem.

The performance of the proposed SAA depends on the values of its parameters which are the initial value of T, Time Limit as the stopping criterion and α. To determine these values, some experiments are needed [32]. Since Taguchi orthogonal array design is a type of general fractional factorial experiment design, it is very effective in parameter setting. Based on noise minimization, this method selects the best level of the parameters. Using the following equation, the deviation of the response is examined, wherein Y designates the value of reply, and n characterizes the number of orthogonal ranges.

$$S/N = (-10) * \log 10(\mathrm{sum}(Y^2)/n)$$

In this thesis for each of initial value of T, Time Limit, and α, a range are determined. These ranges are [90-110] for the initial value of T, [1, 5] for Time Limit and [0.993-0.997] for α. For each of these parameters three different values are used: (1) the lower bound, (2) the average and (3) the upper bound of the corresponding range. Thus, nine orthogonal arrays has been considered of these values and tested. In these tests, the developed SA algorithm is used for solving 5 different instances which are (4, 75), (6, 150), (8, 250), (10, 500) and (12, 750) where the first entry shows the number of the machines and the second one shows the processing time (i.e., (m, p)). For each of the nine combinations of the parameter values, each of these five test instances is solved by the proposed SAA ten times. The average of the cycle times of the best solutions found by these ten runs recorded and presented in Table 22.

Table 22. Computational results for tuning SA parameters.

Combination	SA parameters			Response					
	(A) Initial T value	(B) Time limit	(C) α	(4,75)	(6,150)	(8,250)	(10,500)	(12,750)	Sum
1	90	1	0.993	107.2	197.6	325.6	549.6	812.0	1992.0
2	90	3	0.995	107.4	204.8	328.0	550.8	828.4	2019.4
3	90	5	0.997	108.6	197.6	327.2	550.8	806.0	1990.2
4	100	1	0.995	105.6	200.8	324.8	555.6	839.6	2026.4
5	100	3	0.997	107.2	200.8	322.0	557.2	817.2	2004.4
6	100	5	0.993	108.0	205.6	328.0	550.8	815.2	2007.6
7	110	1	0.997	107.6	202.8	326.0	556.0	813.6	2006.0
8	110	3	0.993	107.2	203.6	330.4	553.2	817.2	2011.6
9	110	5	0.995	107.4	202.8	329.6	554.4	828.4	2022.6

Then the S/N ratios are computed using the results in the last column of Table 22. Figure 20, shows the results of S/N ratios. According to these ratios, when the initial value of T is set to its lower bound (which is 90), Time Limit is set to its upper bound (which is 5 minutes), and α is set to its upper bound (which is 0.997) the best results are obtained. Thus, these values are used in the following tests.

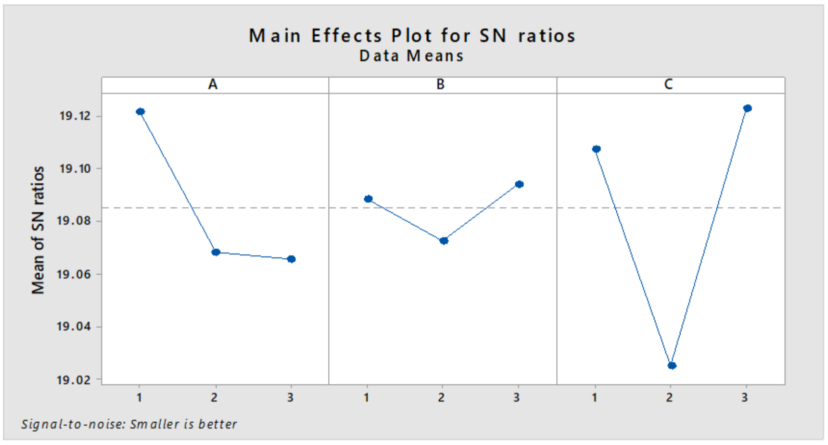

Figure 20. S/N ratio plot for SA parameters.

First, the performance of the proposed SAA is tested on the above test instances whose optimal solutions are found by the mathematical models. Table 23 contains the cycle times and solution times of all the examined test problems for 4- to 6-machine cells found by the proposed SAA.

Table 23. Results of the SAA.

Process time	4-machine cell			5-machine cell			6-machine cell		
	Optimal cycle time	SAA cycle time	SAA solution time	Optimal cycle time	SAA cycle time	SAA solution time	Optimal cycle time	SAA cycle time	SAA solution time
0	96	96	0.033	140	140	0.128	192	192	0.134
25	96	96	0.441	140	140	0.405	192	192	0.321
50	96	96	2.043	140	140	1.260	192	192	0.953
75	99	108.6	2.859	140	140	2.319	192	192	2.215
100	124	124	2.257	140	143.2	4.785	192	192	2.269
125	149	149	1.976	153	160.1	2.741	192	192	4.574
150	174	174	2.049	178	178	2.759	192	197.6	6.203
175	199	199	1.462	203	203	4.087	207	222.7	4.684
200	224	224	2.646	228	228	4.559	232	233	5.778
225	249	249	1.974	253	253	3.344	257	257	5.839
250	274	274	2.536	278	278	4.401	282	282	6.328

According to the results in Table 23, and figures 21 to 23, the proposed SAA found almost all optimal solutions. Only 6 of the 33 instances could not be solved optimally. In these instances, the gap between the optimal cycle times and the best cycle times found by the proposed SAA is less than 10% of the optimal cycle times.

Thus, it may be concluded that the proposed SAA has a very good performance and it may be used to find good solutions to larger instances.

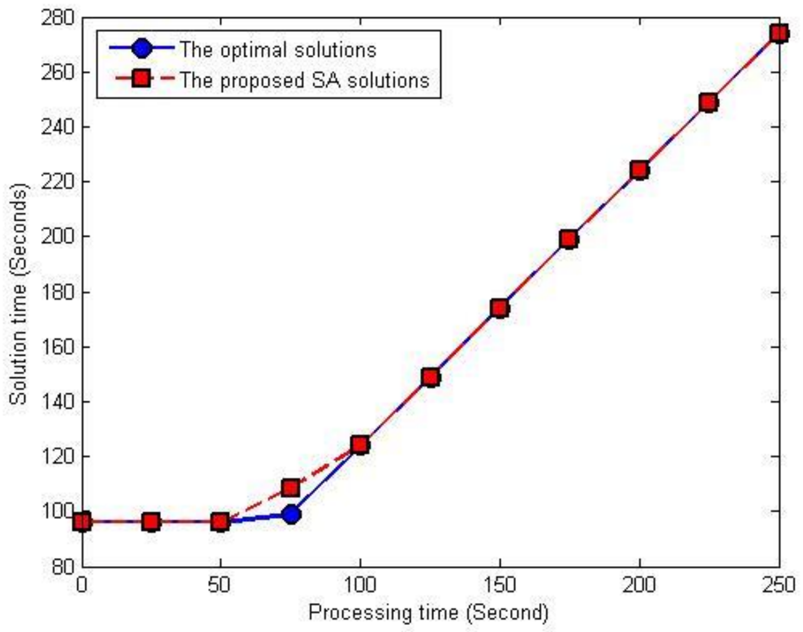

Figure 21. Solution time of the SAA for 4-machine test instances.

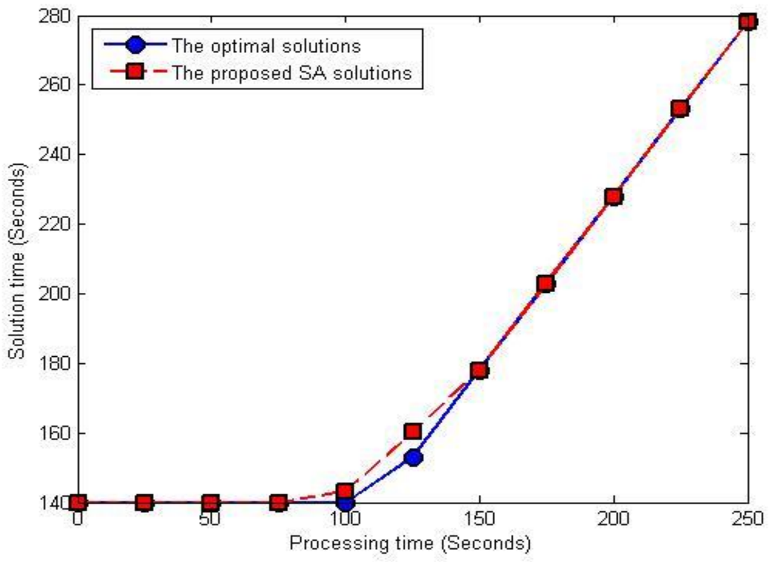

Figure 22. Solution time of the SAA for 5-machine test instances.

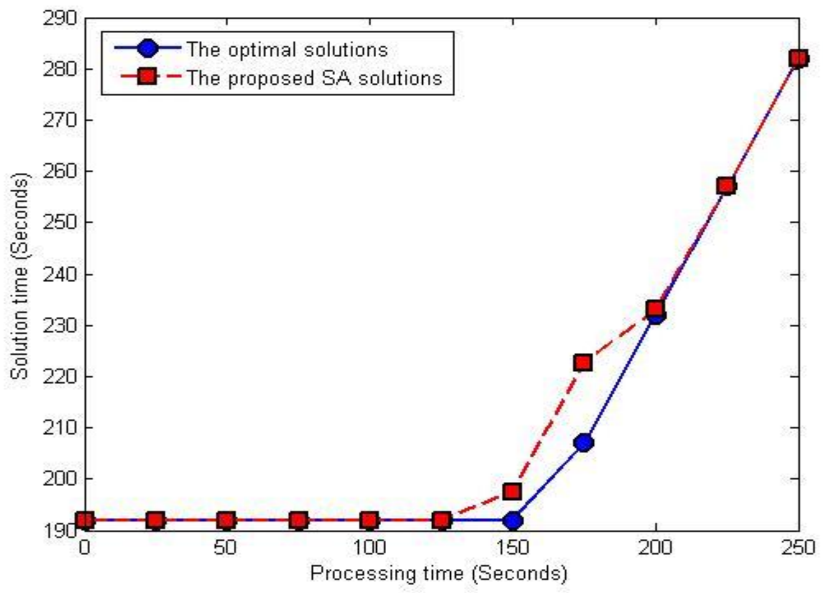

Figure 23. Solution time of the SAA for 6-machine test instances.

Chapter 6

CONCLUSIONS AND FUTURE RESEARCH

In this thesis, firstly the general case of a line layout flexible manufacturing system that includes m parallel machines for producing non-identical parts was considered. The mathematical model associated with it was explained in details and the reduced model was also presented. Furthermore, a new mathematical model was proposed for maximizing the minimum robot return time to each machine in a cycle and the optimal solution for the general case was calculated. A lower bound that has been proposed in the literature was improved and for another lower bound a new proof was provided. The proof was derived considering the optimality condition of an assignment problem. The new proof provided a deeper insight into the structure of the problem. Thus, it made possible to obtain several optimal solutions, and the processing times were used to describe their optimality conditions.

In the second part of this study, a novel problem related to robotic flexible manufacturing cells was discussed in which each machine has individual input buffer with one capacity and the machines are identical and parallel. In these cells, a robot transports the items from the input station that keeps unproduced items to the input buffer of machines and from machines to the output station in which the finished items are kept and loads/unloads the machines. The objective function was to determine the robot movements for minimizing the cycle time. In addition, a lower bound was provided based on the optimality condition of an assignment problem, and with some new definitions three similar cycles were compared in general.

In the last part of the thesis, a metaheuristic algorithms based on simulated annealing (SA) algorithm was proposed. This algorithm was proposed to solve the flexible robotic cell problems that were discussed in the first part of this study. To evaluate the algorithm, a number of problems have been generated and solved. The results demonstrated that the proposed SA algorithm is very efficient.

As a topic for further research, all of the calculations, theorems, and lemmas can be reformulated for a circular layout considering only one station as the input/output station. In addition, the concept of using more than one robot to handle the material in the cell can be a good area of future study. The development of the mathematical model for the problems with different capacities of the individual input buffer can be an interesting objective for the further research. The case that the machines have both individual input and output buffers may also be considered for future research. In addition, developing other metaheuristics for the given problem can be interesting in the further research. Furthermore, considering these problems under uncertainty of each parameter can be a future subject.

REFERENCES

[1] Mehrabi, M.G., Ulsoy, A.G., and Koren Y., *Reconfigurable manufacturing systems: Key to future manufacturing.* Journal of Intelligent Manufacturing. 11(4): p. 403-419.

[2] Ghadiri Nejad, M. and Mosallaeipour, S., *A New Approach to Optimize a Flexible Manufacturing Cell*, in *1st International Conference on New Directions in Business, Management, Finance and Economics*. 2013: Famagusta, Northern Cyprus. p. 38.

[3] Abdekhodaee, A.H., Wirth, A. and Gan, H.S., *Equal processing and equal setup time cases of scheduling parallel machines with a single server*. Computers & Operations Research, 2004. 31(11): p. 1867-1889.

[4] Dawande, M., Geismar, H.N., Sethi, S.P. and Sriskandarajah, Ch., *Sequencing and Scheduling in Robotic Cells: Recent Developments.* Journal of Scheduling, 2005. 8(5): p. 387-426.

[5] Gultekin, H., Ekin Karasan, O. and Akturk, M.S., *Pure cycles in flexible robotic cells.* Computers & Operations Research, 2009. 36(2): p. 329-343.

[6] Crama, Y., *Combinatorial optimization models for production scheduling in automated manufacturing systems.* European Journal of Operational Research, 1997. 99(1): p. 136-153.

[7] Agnetis, A., Pacciarelli, D. and Rossi, F., *Lot scheuling in a two-machine cell with swapping devices.* IIE Transactions (Institute of Industrial Engineers), 1996. 28(11): p. 911-917.

[8] Drobouchevitch, I.G., Sethi, S.P. and Sriskandarajah, C., *Scheduling dual gripper robotic cell: One-unit cycles.* European Journal of Operational Research, 2006. 171(2): p. 598-631.

[9] Sethi, S.P., Sriskandarajah, Ch., Sorger, G., Blazewicz, J. and Kubiak, W., *Sequencing of parts and robot moves in a robotic cell.* International Journal of Flexible Manufacturing Systems, 1992. 4(3-4): p. 331-358.

[10] Crama, Y. and Van de Klundert, J., *Cyclic scheduling in 3-machine robotic flow shops.* Journal of Scheduling, 1999. 2(1): p. 35-54.

[11] Hall, N.G., Potts, C.N. and Sriskandarajah, C., *Parallel machine scheduling with a common server.* Discrete Applied Mathematics, 2000. 102(3): p. 223-243.

[12] Brauner, N. and Finke, G., *Cycles and permutations in robotic cells.* Mathematical and Computer Modelling, 2001. 34(5–6): p. 565-591.

[13] Akturk, M.S., Gultekin, H. and Karasan, O.E., *Robotic cell scheduling with operational flexibility*. Discrete Applied Mathematics, 2005. 145(3): p. 334-348.

[14] Gultekin, H., Akturk, M.S. and Karasan, O.E., *Scheduling in a three-machine robotic flexible manufacturing cell.* Computers & Operations Research, 2007. 34(8): p. 2463-2477.

[15] Gultekin, H., Akturk, M.S., and Karasan, O.E., *Scheduling in robotic cells: process flexibility and cell layout.* International Journal of Production Research, 2008. 46(8): p. 2105-2121.

[16] Gultekin, H., Akturk, M.S., and Karasan, O.E., *Bicriteria robotic cell scheduling.* Journal of Scheduling, 2008. 11(6): p. 457-473.

[17] Ghadiri Nejad, M., Kovács, G., Vizvári, B. and Vatankhah Barenji, R., *An optimization model for cyclic scheduling problem in flexible robotic cells.* The International Journal of Advanced Manufacturing Technology, 2017: p. 1-11.

[18] Ghadiri Nejad, M., Güden, H., Vizvári1, B., and Vatankhah Barenji, R., *A Mathematical Model and Simulated Annealing Algorithm for Solving the Cyclic Scheduling Problem of a Flexible Robotic Cell.* Advances in Mechanical Engineering, 2018. 10: p. 1-12.

[19] Aktürk, M.S, Atamtürk, A. and Gürel, S.N., *A strong conic quadratic reformulation for machine-job assignment with controllable processing times.* Operations Research Letters, 2009. 37(3): p. 187-191.

[20] Yildiz, S., Karasan, O.E. and Akturk, M.S., *An analysis of cyclic scheduling problems in robot centered cells.* Computers & Operations Research, 2012. 39(6): p. 1290-1299.

[21] Uruk, Z., Gultekin, H., and Akturk, M.S., *Two-machine flowshop scheduling with flexible operations and controllable processing times.* Computers & Operations Research, 2013. 40(2): p. 639-653.

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

[23] Foumani, M. and Jenab, K., *Cycle time analysis in reentrant robotic cells with swap ability.* International Journal of Production Research, 2012. 50(22): p. 6372-6387.

[24] Foumani, M. and Jenab, K., *Analysis of flexible robotic cells with improved pure cycle.* International Journal of Computer Integrated Manufacturing, 2012. 26(3): p. 201-215.

[25] Jolai, F., Foumani, M., Tavakoli-Moghadam, R. and Fattahi, P., *Cyclic scheduling of a robotic flexible cell with load lock and swap.* J. Intell. Manuf., 2012. 23(5): p. 1885-1891.

[26] De Giovanni, L. and Pezzella, F., *An Improved Genetic Algorithm for the Distributed and Flexible Job-shop Scheduling problem.* European Journal of Operational Research, 2010. 200(2): p. 395-408.

[27] Batur, G.D., Karasan, O.E., and Akturk, M.S., *Multiple part-type scheduling in flexible robotic cells.* International Journal of Production Economics, 2012. 135(2): p. 726-740.

[28] Kim, H., Kim, H., Lee., J. and Lee, T., *Scheduling dual-armed cluster tools with cleaning processes.* International Journal of Production Research, 2013. 51(12): p. 3671-3687.

[29] Ghadiri Nejad, M., Shavarani, S. M., Vizvari, B. and Vatankhah Barenji, R., *Trade-off between process scheduling and production cost in cyclic flexible robotic cell.* The International Journal of Advanced Manufacturing Technology, 2018.

[30] Mosallaeipour, S., Ghadiri Nejad, M., Shavarani, S. M., and Nazerian, R., *Mobile robot scheduling for cycle time optimization in flow-shop cells, a case study*. Production Engineering, 2017: p. 1-12.

[31] Sabar, N.R. and Kendall, G., *An iterated local search with multiple perturbation operators and time varying perturbation strength for the aircraft landing problem.* Omega, 2015. 56: p. 88-98.

[32] Shavarani, S.M., Ghadiri Nejad, M., Rismanchian, F. and Izbirak, G., *Application of hierarchical facility location problem for optimization of a drone delivery system: a case study of Amazon prime air in the city of San Francisco.* The International Journal of Advanced Manufacturing Technology, 2017: p. 1-13.