ROBOTIC CELL SCHEDULING WITH OPERATIONAL FLEXIBILITY

A THESIS

SUBMITTED TO THE DEPARTMENT OF INDUSTRIAL ENGINEERING

AND THE INSTITUTE OF ENGINEERING AND SCIENCES

OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

By

Hakan Gültekin

June 2002

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

Assoc. Prof. M. Selim Aktürk(Principal Advisor)

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

Asst. Prof. Oya Ekin Karaşan

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

Assoc. Prof. Sinan Kayalıgil

Approved for the Institute of Engineering and Sciences:

Prof. Mehmet Baray
Director of Institute of Engineering and Sciences

ABSTRACT

ROBOTIC CELL SCHEDULING WITH OPERATIONAL FLEXIBILITY

Hakan Gültekin
M.S. in Industrial Engineering
Supervisor: Assoc. Prof. M. Selim Aktürk
June 2002

In this thesis, we study the problem of two-machine, identical parts robotic cell scheduling with operational flexibility. We assume that every part to be processed has a number of tasks to be completed in these two machines and both machines are capable of performing all of the tasks. The decision to be made includes finding the optimal robot move cycle and the optimal allocation of tasks to these two machines corresponding to this robot move cycle that jointly minimize the cycle time. We proved that 1-unit robot move cycles are not necessarily optimal with this definition of the problem any more and that according to given parameters either one of the 1-unit robot move cycles or a 2-unit robot move cycle is optimal. We proposed a new robot move cycle, which is a result of the assumption of operational flexibility. This cycle is not only simple and practical but also dominates all of the common cycles reported in the literature. Finally, we considered the change of layout and showed that the cycle time of the proposed cycle can be further reduced by a change in the layout while the cycle times of all other cycles remain the same.

Keywords: Robotic cell, cyclic scheduling, automated manufacturing.

ÖZET

OPERASYONEL ESNEKLİKLİ ROBOTİK HÜCRE ÇİZELGELEME PROBLEMİ

Hakan Gültekin
Endüstri Mühenlisliği Yüksek Lisans
Tez Yöneticisi: Doç. Dr. M. Selim Aktürk
Haziran 2002

Bu tezde iki makinalı, operasyonel esneklikli ve tek tip parça üreten robotik hücreler konu alınmıştır. Üretilecek her parçanın bu iki makinada belirli sayıda işlemden geçmesi gerekliliği ve her iki makinanın da bütün işlemleri yapabilecek kapasitede olması kabul edilmiştir. Verilmek istenen karar, döngü zamanını en aza indiren robot hareket döngüsünü ve işlemlerin makinalara atamasını bulmak. Bu problem tanımı için 1-parça üreten hareket döngülerinin artık mutlaka en iyi olmadığını, verilen parametrelere göre bazen 1-parça bazen de 2-parça üreten hareket döngülerinin en iyi olabileceğini ispatladık. Operasyonel esneklik kabullenmesinin sonucu olarak ortaya çıkan yeni bir robot hareket döngüsü önerdik. Önerilen bu yeni döngü sadece basit ve kullanışlı olarak kalmadı, aynı zamanda bilinen bütün hareket döngülerinden daha kısa bir döngü zamanı verdi. Son olarak, robot ve makinaların yerleşimini değiştirerek önerilen yeni hareket döngüsünün zamanını daha da azaltabileceğimizi ve bunu yaparken bilinen diğer döngülerin zamanlarında bir değişme olmayacağını gösterdik.

Anahtar Sözcükler: Robotik hücre, döngüsel çizelgeleme, otomasyonlu üretim.

To my angels and big lovely family...

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to Assoc. Prof. M. Selim Aktürk for all the encouragement and trust during my graduate study. He has been supervising me with patience and his great helps brings this thesis to an end.

I am grateful to Assist. Prof. Oya Ekin Karaşan for her invaluable guidance, remarks and recommendations. Her understanding and sympathy was a great encouragement for me.

I am also indebted to Assoc. Prof. Sinan Kayalıgil for accepting to read and review this thesis and for his suggestions.

I would like to thank to Selçuk Gören for his friendship. Without his academic and most importantly morale support I would not be able to bear all.

I also would like to thank to Taylan İlhan and Hüseyin Bilal Pandul for their friendship during six years of time. I am grateful for their support.

Finally, I would like to express my deepest gratitude to my wife, Feyza Gültekin for everything she brought to my life.

Contents

List of Figures

Chapter 1

Introduction

In order to be successful in today's highly competitive world, increasing productivity is an essential factor for manufacturing systems. Recent technological improvements opened new perspectives for industries. Since setup times are reduced to improve flexibility, as stated by Kamoun et al. [20] material handling time and cost become bottlenecks, and efficient material handling becomes very important. Depending on the type of manufacturing facility, estimates ranging from 10% to 80% of total cost have been attributed to material handling [38]. Hartley [15] suggests that a cellular manufacturing is ideally suited to material handling by a robot. Today, robots are used extensively in automotive, electrical, electronic and mechanical engineering industries to perform tasks ranging from assembly to testing and inspection [3]. Miller and Walker [32] cite an application where a robot was installed to load and unload a 500 pound workpiece to a series of machines and finally to discharge the workpiece at an output station. Browne et al. [3] states that robots are installed in order to;

1- Reduce labor cost,

2- Increase output,

3- Provide more flexible production system,

4- Replace people working in dangerous or hazardous conditions,

5- Compensate for local shortage of skilled labor and

6- Achieve more consistent control of quality.

The manufacturing cells in which the loading and unloading of the parts are done by a robot are called robotic cells [23]. There may be 2, 3 ..., m machines in a robotic cell with or without buffers. The buffers also maybe finite capacity or infinite capacity buffers. Three different layouts are considered for the robotic cells: robot-centered cells (where the robot movement is rotational), in-line robotic cells (where the robot moves linearly) and mobile-robot cells (generalization of in-line robotic cell and robot-centered cell) [31]. The objective can be to minimize the makespan or to minimize the cycle time. Since the robot follows a computer program, there must be a finite activity sequence for the robot that it repeats to produce the parts. Thus the robot activities must be cyclic because of its nature and minimizing this cycle time or in other words maximizing the throughput is a relevant objective.

There may be identical parts to be processed, in which there is no part sequencing problem, but only the robot activity sequence is to be found. Once this sequence is found, then the production of these identical parts consists of repetition of this activity sequence in required numbers. Also the multiple parts case can be considered in which case the problem is to find the robot move sequence and the part input sequence jointly. However, again because of the nature of the industrial robots, the multiple parts case also must follow a cycle and this cycle can use the concept of minimal part sets (MPS). The current trend towards just-in-time (JIT) manufacturing with cyclic production schedules require producing a quantity known as minimal part set (MPS). The MPS for a production environment can be obtained, if the forecasted demand L_i is given for each part type i over planning horizon. If d is the largest common divisor of the integers, $L_1, L_2, \ldots, L_k$, the integer ratio ($r_i = L_i/d$) of part types can be represented as $r = (r_1, r_2, \ldots, r_k)$. The vector is the minimal production ratio (MPR) and the part set corresponding to this ratio is known

as the MPS. Thus the problem in multiple parts case is to find the robot move sequence and the part input sequence of the MPS.

In this study, we will consider a 2-machine robotic cell with in-line robotic cell layout. We will consider identical parts to be produced. There are no buffers at or between the machines. In the current literature, the allocation of the tasks to each machine is assumed to be constant and for given processing times the optimum robot move cycle minimizing the cycle time is to be determined. Since the processing stations in a robotic cell are predominantly CNC machines, they possess *operation flexibility* by definition. Operation flexibility can be defined as the ability to interchange the ordering of several operations for each part type (Browne et al. [3]). Therefore, assuming that processing times are fixed on each CNC machine is rather unrealistic and limits the number of alternatives unnecessarily. In this study we assume that each part has a number of different tasks to be completed. Then, the problem to be considered in this thesis is to allocate the tasks to the machines and is to find the robot move cycle that corresponds to this allocation of tasks in order to jointly minimize the cycle time or in other words, maximize the long run average throughput rate.

For this problem definition, we proved that the optimal solution is not necessarily a 1-unit cycle as in the case of Sethi et al. [34] and showed that the optimal solution could be a 2-unit cycle for some parameter input, where an n-unit cycle can be defined as a robot move cycle which produces exactly n units and ends up with the same state of the cell as the starting state. We proposed a new robot move cycle, which is a result of assuming operational flexibility, and proved that this cycle has a smaller cycle time than all of the common ones reported in the literature. As Han et al. [14] proposed, cell formation may increase efficiency of the cell, thus, we considered changing the layout of the cell from in-line robotic cell to robot-centered cell and proved that this change further improves the cycle time of the proposed robot move cycle while all of the others remain the same.

We also considered the tooling constraints. The CNC machines are capable of performing different operations, but in order to perform these operations they require different tools. These tools are placed and stored in the tool magazines of these machines. However, these tool magazines have limited capacity and sometimes because of this capacity constraint not all of the tools required to produce a part can be placed to the tool magazines. Thus, these tools are placed to an other machine and the tasks that must use these tools can only be processed on this machine. We considered this problem and proved that the optimal solution is either one of the 1-unit robot move cycles or a 2-unit robot move cycle and provided the optimality regions for these robot move cycles.

The remainder of this study is organized as follows: In the next chapter an extensive review of the literature is provided. In Chapter 3, the task allocation problem is defined and the solution procedure is given. In Chapter 4 we consider the problem with tooling constraints and provide the solution for this problem, whereas the last chapter is devoted to concluding remarks and future research directions.

Chapter 2

Literature Review

With the rapid increase of using robots in manufacturing, the problems to be solved in order to increase efficiency in these systems also increase. As a result of this, there are many different types of problems that are considered in the literature. Before reviewing these problems let us first introduce notations, some definitions and terminology that will be used throughout this study.

We will use the following parameters and decision variables:

$t_i =$ Processing time of task i on either machine 1 or 2, $i = 1, 2, \ldots, p$.

$P =$ Total processing time required to finish the production of a part, that is the sum of the processing times of all the p tasks, $t_1 + t_2 + \ldots + t_p$.

$\epsilon =$ The load and unload time of workstations by the robot.

$\delta =$ Time taken by the robot to travel between two consecutive stations.

$C_t =$ Cycle time, i.e., the time that is required to perform each robot activity exactly once and produce one part.

The following definitions on robot activities and n-unit robot move cycles are borrowed from [4].

Definition 2.1 A_i is the robot activity defined as; robot unloads machine i, transfers part from machine i to machine $i+1$, loads machine $i+1$. The machines are numbered as 1 and 2, the input buffer is numbered as 0 and the output buffer is numbered as 3. So we have activities A_0, A_1, A_2.

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

Our problem definition allows for the possibility of the parts of a given robot move cycle to have different allocation of tasks to the machines. Hence, we shall make use of the following definition.

Definition 2.3 A given robot move cycle uses k-allocation types if in a cyclic manner, the first k parts in the queue have differing allocation of tasks to the machines and after the k^{th} one the cycle returns to the beginning and the next k parts have the same allocation types as the previous k ones.

Definition 2.4 *Consider the robot move cycle $S2$ using i-allocation types. Let P_{ij} be the processing time of a part which uses the j^{th} allocation type on machine 1 where $j \leq i$. Note that under this setting, the processing time of this part on machine 2 is simply $P - P_{ij}$.*

Sethi et al. [34] proved that there are two feasible 1-unit robot move cycles:

$S1 \equiv$ This can be represented by the robot activities as $A_0A_1A_2$. Initially the system is empty and the robot is in front of the input buffer. After the activities are performed, the robot returns to the input buffer.

$S2 \equiv$ This cycle is represented as $A_0A_2A_1$. Initially, only the second machine is loaded and the robot is in front of the input buffer. Then the robot sequences these activities and returns to the input buffer.

After presenting the definitions and the terminology we can now review the problems considered in the literature. Mainly we will divide these problems as no-buffer problems and infinite or finite-buffer problems. Under this main heading there are subtopics where these problems differ from each other. Crama et al. [6] lists the distinguishing features of various types of flowshops as follows:

1. Processing Requirements:

$[L_i^j, U_i^j]$ is the processing window which gives the processing requirements of part j on machine M_i. The meaning of the processing window is that the time spent by part j on machine M_i must be at least L_i^j and may not exceed U_i^j. This representation of the processing requirements is very general and contains many special cases. For example, the case where each part has a precisely defined processing time on each machine and can wait on the machine indefinitely long after it has been processed, can be handled by setting all upper bounds U_i^j to $+\infty$. This case is referred as *unbounded processing windows* [6]. Another type of processing requirement referred as *with blocking* is one, in which the machine becomes blocked if the part is not removed from the machine after the processing is finished. Another model is referred to as *no-wait* problems and in these problems it is assumed that the parts must be removed from the machines immediately after the processing is finished. These two problems can be modelled by setting $L_i^j = U_i^j$. This setting of processing windows is referred to as *zero-width processing windows* [6].

2. Time models for the robot:

This feature considers the loading-unloading times of the machines by the robot and the travel times of the robot between the machines. Several authors considered the loading and unloading times to be machine depended, that is, it takes ϵ_i time to load or unload a part to machine M_i. The simpler models

assume that the loading and unloading times are all equal, that is $\epsilon_i = \epsilon$ or sometimes they are assumed to be all zero.

Another characteristic of the robot is the travel speed of the robot. Again simpler models assume that the travel speed of the robot is constant in all cases, while some other authors assumed that the travel time between machines M_i and M_j denoted by δ_{ij}. In many situations these times are assumed to be symmetric, that is, $\delta_{ij} = \delta_{ji}$ and to satisfy the triangle inequality, that is, for $i < j < k, \delta_{ij} + \delta_{jk} \geq \delta_{ik}$. In some cases, it is assumed that the travel times are symmetric and the following relation holds: for $i < j < k, \delta_{ij} + \delta_{jk} = \delta_{ik}$, this case is referred as the *triangle equality* [6].

3. Number of machines and parts:

In the literature, 2-machine, 3-machine and m-machine robotic flowshops are considered. Since, as the number of machines increases, the problems becomes more complex, most of the analytical results are derived for 2-machine or 3-machine flowshops. Another factor that determines the complexity and the size of the problem is the number of parts. Some models consider identical parts in which the part scheduling vanishes, while some other considers multiple parts. In the multiple parts case it is assumed that a *minimal part set* (MPS) is being processed.

4. Objective functions:

There are two objective functions that are commonly used in most of the research on this area. The first one is maximizing the throughput or in other words minimizing the cycle time. The other one is minimizing the makespan of the schedule.

Another difference among the models is the layout of the robotic cells.

Han et al. [14] proposed that cell formation may increase efficiency of the cell. Three different robotic cell layouts are considered in the literature: robot-centered cells (RCC) (where the robot movement is rotational), in-line robotic cells (IRC) (where the robot moves linearly on linear tracks) and mobile-robot cells (MRC) (generalization of robot-centered cell and in-line robotic cells) [31]. The findings with the no-buffer case will be presented in §2.1 and with the finite or infinite-buffer case in §2.2. In §2.3 we will present some other interesting configurations of robotic cell flowshops and in the last section, the motivations for this study will be explained. First of all, let us give the classification scheme for robotic flowshops, which will help us through the rest of this section. The standard classification scheme for scheduling problems (Graham et al. [10]) can be denoted $\psi_1|\psi_2|\psi_3$ where ψ_1 indicates the scheduling environment, ψ_2 indicates the job characteristics or restrictive requirements, and ψ_3 defines the objective function to be minimized. Hall et al. [11] extended this scheme to provide for the scheduling problems arising in robotic cells, as follows:

Under ψ_1, we have:

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

Under ψ_2, we have:

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

Under ψ_3, we have:

C_t = the average steady state cycle time for the repetitive manufacture of an MPS.

C_{max} = the makespan for the manufacture of a given set of jobs.

2.1 The no-buffer case

Most of the research made on robotic cell scheduling assumes that there are no buffers at or between the machines. Sethi et al. [34] considered $RCC2|k = 1, \delta_i = \delta, \epsilon_i = \epsilon|C_t$ and proved that 1-unit cycles give the optimal results for this problem. Furthermore, they presented the following theorem which shows the regions of optimality for these two 1-unit robot move cycles.

Theorem 2.1 *$S1$ is optimal if and only if $P \leq 2\delta$, and $S2$ is optimal if and only if $P \geq 2\delta$. In the limiting case $P = 2\delta$, both policies perform equally well.*

For the 3 machine identical parts problem, Sethi et al. [34] considered only the 1-unit cycles and prepared a decision tree for these 1-unit cycles. They conjectured that 1-unit cycles yield optimal production rates for m-machine robotic flowshops, $m \geq 2$. Hall et al. [11] proved that, for 3-machine identical part types case, the repetition of 1-unit cycles dominates more complicated policies that produce 2 units. Later on Crama et al. [5] established the validity of the conjecture for 3-machine robotic flowshops. Brauner and Finke [2] simplified the proof of Crama et al. [5].

Crama et al. [4], extended the results of Sethi et al. [34] and proved that when there is only one type of parts to be produced, the problem can be solved in (strongly) polynomial time, even if the number of machines is viewed as an input parameter of the problem. This is achieved by proving that the set of pyramidal permutations necessarily contains an optimal solution of the problem. Pyramidal permutations have been previously

introduced in the framework of the travelling salesman problem; see e.g. Gilmore et al. [9]. Crama et al. [4] proved that every permutation of the robot activities starting with A_0 corresponds to a 1-unit cycle. Let $\pi = (A_0, A_{i_1}, \ldots, A_{i_k}, A_{i_{k+1}}, \ldots, A_{i_m})$. Then π is pyramidal if $1 \leq i_1 < \ldots < i_k = m$ and $m > i_{k+1} > \ldots > i_m \geq 1$. An algorithm is given which computes the cycle time of a schedule described by a pyramidal permutation. Lastly, a dynamic programming approach is presented that solves the identical parts cyclic scheduling problem with the restriction that one unit be produced in each cycle in $O(m^3)$ time where m is the number of machines in the cell.

For the multiple parts case, Sethi et al. [34] also considered $RCC2|k \geq 2, \delta_i = \delta, \epsilon_i = \epsilon, S_1(S_2)|C_t$. For this problem they found a strongly polynomial procedure that generates the optimal job sequence. This result is achieved by showing that the problem of finding the optimal job input sequence can be reduced to a solvable case of the travelling salesman problem. They also showed that for any given m-machine cell, there are only $m!$ 1-unit cycles.

Sriskandarajah et al. [36] extended the results of Sethi et al. [34] for the two-machine multiple parts problem. They indicated that the optimal cycle time is not necessarily attained by repeating the same 1-unit cycle even when $m = 2$.

Hall et al. [11] proved that for 2-machine cells, producing multiple part types, 1-unit cycles generally do not give the optimal solution. Furthermore, they provide an example to clarify it. An algorithm is provided for this case which gives the robot move cycle and part input sequence that jointly minimize the cycle time.

Kise et al. [26] consider 2-machine multiple parts problem with the objective of minimizing the makespan. They propose an $O(n^3)$ procedure, where n is the number of different parts in the queue, that solves the problem based on the Gilmore and Gomory algorithm (Gilmore and Gomory [8]). For the same problem, if the transportation time between the machines is job dependent, then the problem is equivalent to an asymmetric travelling salesman problem, which is shown to be NP-hard in the strong sense (Stern and Vitner

[37]).

Again for the 2-machine, multiple parts problem, Logendran and Sriskandarajah [31] considered three different layouts and established optimal robot sequences for these layouts. The problem of determining the optimal sequence of multiple parts types has been shown to be equivalent to a two machines no-wait flow shop problem, and has been solved by Gilmore and Gomory's algorithm. Besides the analysis of a single MPS, they also analyzed the production of multiple MPSs.

For 3-machine multiple part types case, Hall et al. [11] showed that the optimal part sequencing problems associated with 4 of the 6 potentially optimal robot move cycles for producing 1-unit are polynomially solvable. For the remaining 2 cycles, they showed that the recognition version of the part sequencing problem is NP-complete. In this paper they also provide the conditions on the relative lengths of processing times compared to robot move times under which the last 2 cycles are dominated by the other 4.

Later on Sriskandarajah et al. [36] extended this result. They classified 1-unit robot move cycles in an m-machine cell, for $m \geq 2$, according to the tractability of their associated part sequencing problems. The classification is as follows:

$ClassU$ = Sequence independent (trivially solvable),
$ClassV1$ = Capable of formulation as a TSP but polynomially solvable
$ClassV2$ = Unary NP-hard TSP models,
$ClassW$ = Unary NP-hard, but not having TSP structure.

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

Sriskandarajah et al. [36] proved that 3-machine multiple parts problem is intractable. Thus, Kamoun et al. [20] designed and tested simple heuristic

procedures for the part sequencing problem under different robot move cycles. That is, they fixed the robot move cycle and designed a heuristic that solves the part sequencing problem for this fixed robot move cycle. This enabled them to develop a heuristic procedure for a general 3-machine cell without fixing any robot move cycle. In order to extend this study from 3-machines to 4-machines, they described and tested a methodology.

2.2 The finite and infinite-buffer case

As a general rule, the additional freedom introduced by buffers tends to complicate scheduling problems [33]. For example, buffers may allow to consider schedules in which parts are processed in different orders on different machines. For 2-machines, in the case of minimizing the makespan, Kise [25] showed that the robot can be looked as a third machine and proved that this problem is NP-hard. Hurink and Knust [17] also considered the same objective with n jobs consisting of m operations which have to be processed in the same order on m machines. They derived complexity results for these problems. Hitomi and Yashimura [16] considered a problem with m machines, $m-1$ intermediate buffers, one robot and two conveyors (one incoming, one outgoing). They have developed a branch-and bound based algorithm for obtaining an optimum robot transport sequence that minimizes the makespan of completing a given set of jobs. For the same objective, King et al. [24] considered 2-machine multiple parts problem where each machine has a finite input buffer but no output buffer (i.e. a machine becomes blocked if a completed job is not removed). They assumed a fixed input sequence, and then determined the sequence of robot moves by using a branch-and bound procedure.

For single capacity buffers, Finke et al. [7] identified optimal robot move sequences, while for infinite intermediate buffers with 2-machines, where each machine has its own servicing robot Levner et al. [28] and Kogan and Levner

[27] showed that the problem can be solved using a polynomial-time Johnson-type algorithm.

For multiple parallel machines, Jeng et al. [19] assumed that the input sequence of a given set of jobs is fixed and proposed a branch-and-bound algorithm to find the optimal schedule of robot moves. The objective was to minimize the total flow time or $\sum C_j$. The shortest processing time (SPT) rule is used to derive an initial sequence and an upper bound for the search.

2.3 Other Robotic Cell Configurations

There are many other interesting robotic cell configurations in the literature. An important class of problems addresses robotic cells involving more than one robot. Problems of this type can be found in Karzanov and Livshits [21], and Lieberman and Turksen [30]. Kats and Levner [22] consider minimization of cycle time over the 1-unit robot move cycles with the no-wait constraint and identical parts. They solved the problem for single and several robots. They assumed that the set of machines served by each robot is known in advance. They solved the problem in $O(m^3 logm)$ time. They also showed that the cyclic multi-robotic problem considered in the paper can be interpreted as a new polynomially solvable case of the cyclic no-robot job-shop scheduling problem.

A special case of robotic cell scheduling is when all of the machines are identical and working in parallel. Hall et al. [13] considered these systems with different objectives (e.g. C_{max}, L_{max}, $\sum T_j$, etc.) and derived complexity results for these problems.

Some research on this area considered different objective functions other than minimizing the makespan or the cycle time. For instance, Song et al. [35] and Jeng et al. [19] tried to minimize the sum of completion times. On the other hand, Levner and Vlach [29] consider an objective which minimizes some penalty function of the maximum lateness.

Hurink and Knust [18] consider a single machine scheduling problem which arises as a subproblem in a job shop environment where the jobs have to be transported between the machines by a single transport robot. They presented a tabu search algorithm for this problem, where they considered it as a generalized TSP.

Lastly, Han and Cook [14], developed a mathematical model and a solution algorithm for solving a robot acquisition and cell formation problem. They formulated the problem as a multi-type two-dimensional bin packing problem, a pure 0-1 integer program, which is known to be NP-hard. They developed and implemented a heuristic algorithm. The computational results showed that the problems solved with a gap of less than 1% and over 70% of all solutions (334 out of 450) were optimal.

2.4 Motivations for this study

In the current literature, the allocation of the tasks to each machine is assumed to be constant. For this problem definition with 2-machines, Sethi et al. [34] proved that 1-unit cycles give the optimal solution. They also conjectured that 1-unit cycles give optimal solution for m-machine case. Later Crama et al. [4] extended the results of Sethi et al. and proved that 1-unit cycles also give the optimal solution for the 3-machine case. Hall et al. [11] proved that for 2-machines multiple parts case 1-unit cycles are not necessarily optimal.

However, the workstations in a robotic cell are predominantly CNC machines and they possess an operational flexibility by definition. Therefore, assuming that the allocation of tasks are fixed on each CNC machine limits the number of solution alternatives. In this paper we considered the problem of finding the optimal robot move cycle while simultaneously allocating the tasks to the machines in order to jointly minimize the cycle time. We proved that the optimal robot move cycle is not necessarily a 1-unit cycle as Sethi et al [34] proved but a 2-unit cycle may also be optimal for some parameter input.

We also presented the regions of optimality for the robot move cycles. We showed that the assumption of fixed allocation of tasks on each CNC machine limits the number of solution alternatives. We proposed a new robot move cycle which is a result of the assumption of operational flexibility. We proved that this cycle gives a smaller cycle time than the common robot move cycles considered in the literature.

Chapter 3

Robotic Task Allocation

3.1 Preliminaries and Problem Definition

In this section we give a formal definition of our problem and analyze the restricted case of 1-unit robot move cycles.

We consider an in-line robotic cell of two identical machines which repeatedly produces one type of product. There is no buffer between machines. The layout of the cell can be seen in Figure 3.1.

Each of the identical parts has a number of tasks to be performed. The processing times are assumed to be identical for every task in both of the machines. Consistent with the existing literature (see [34]), the loading and unloading times and the travel time of the robot from one station to a consecutive station are all assumed to be constant. Furthermore, the robot travel times satisfy the triangular equality, that is, for $i < j < k, \delta_{ij} + \delta_{jk} = \delta_{ik}$. So, according to these definitions and assumptions, by using the classification scheme of Hall et al. [11] we can define our problem as IRC2$|k = 1, \delta_i = \delta, \epsilon_i = \epsilon|C_t$ with operational flexibility.

In the remainder of this section, we will show how to find the cycle times

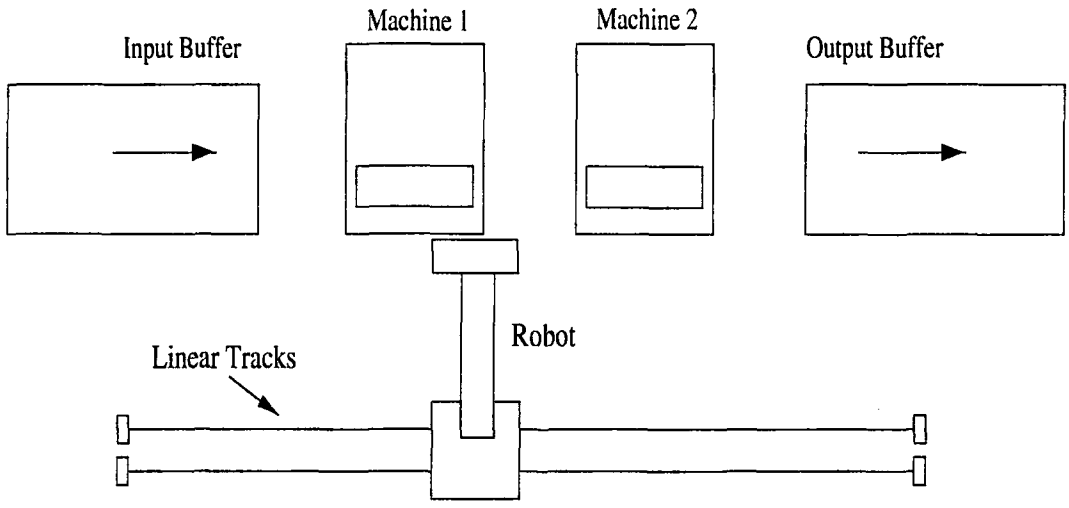

Figure 3.1: Inline Robotic Cell Layout

for 1-unit robot move cycles, and how to model the allocation problem as an integer linear program.

3.1.1 Cycle Time Calculations

Firstly, let us recall the two 1-unit robot move cycles defined by Sethi et al. [34]:

$S1 \equiv$ This can be represented by the robot activities as $A_0A_1A_2$. Initially the system is empty and the robot is in front of the input buffer. After the activities are performed, the robot returns to the input buffer.

$S2 \equiv$ This cycle is represented as $A_0A_2A_1$. Initially, only the second machine is loaded and the robot is in front of the input buffer. Then the robot sequences these activities and returns to the input buffer.

Consider the robot move cycle $S2$ using i-allocation types. As defined formerly, P_{ij} is the processing time of a part which uses the j^{th} allocation type on machine 1 where $j \leq i$. Note that under this setting, the processing time of this part on machine 2 is simply $P - P_{ij}$.

Then the cycle time for $S1$ can be found as follows:

Initially the two machines are empty and the robot is in front of the input buffer. The cycle starts with this state of the system and must end up with the same state as well. The robot first takes a part from the input buffer (ϵ), transfers the part to machine 1 (δ), loads this machine (ϵ), waits in front of the machine to finish the operation (P_{11}), unloads the machine, transfers the part to second machine and loads it ($\epsilon + \delta + \epsilon$), waits in front of the machine to finish the processing, ($P - P_{11}$), unloads machine 2, transfers the part to output station ($\epsilon + \delta + \epsilon$)and turns back to the input buffer (3δ). Thus the total cycle time is equal to:
$C_t^{S1} = 6\epsilon + 6\delta + P_{11} + P - P_{11}$
$\Rightarrow C_t^{S1} = 6\epsilon + 6\delta + P$

The calculation of the cycle time of the robot move cycle $S2$ is somewhat different than this one:
For this robot move cycle, initially the robot is in front of the input buffer, however the first machine is empty but the second machine is loaded. The robot takes a part from the input buffer, transports it to the first machine and loads the first machine ($\epsilon + \delta + \epsilon$). In this cycle, the robot does not wait for the machine to finish the operation but immediately after loading the first machine goes to the second machine (δ). Since the machine is initially loaded, the robot has two alternatives: If the processing of the part is finished, robot unloads the machine else if the processing is not finished yet, the robot waits for the machine. So the waiting time for the robot in front of the second machine in the first cycle is denoted by w_{12}. After this waiting (or 0-waiting) time, the robot unloads the second machine, transports the part to the output buffer and places the part to the output buffer ($\epsilon + \delta + \epsilon$). Then it returns to the first machine in order to unload the machine (2δ). Remember that the cycle is defined as starting with an initial state of the system, the robot performs each activity exactly once and ends up with the same state of the system with the initial state. Thus the cycle is not completed yet. Again the robot either waits for the machine in order to complete the processing and unloads the machine afterwards or unloads the machine without waiting (w_{11}: waiting time for the

first machine in the first cycle) (ϵ). Afterwards, the robot transports the part to the second machine, loads the machine, and returns to the input buffer ($\delta + \epsilon + 2\delta$). So the cycle is completed, and the cycle time is the following:

$C_t^{S2} = 6\epsilon + 8\delta + w_{11} + w_{12}$

Now let us find w_{11} and w_{12}. After loading the first machine robot makes the following activities to return back to unload the first machine. It goes to the second machine (δ), it might wait (w_{12}), unloads the second machine (ϵ), transports the part to output buffer, places the part to the output buffer and returns back to the first machine ($\delta + \epsilon + 2\delta$). Thus the robot spends $w_{12} + 2\epsilon + 4\delta$ time and if P_{11}, the processing time of the first machine is less than this time then the robot does not wait for the machine, else it waits a time of $P_{11} - w_{12} - (2\epsilon + 4\delta)$. So we can represent this as follows:

$w_{11} = max\{0, P_{11} - w_{12} - (2\epsilon + 4\delta)\}$

In the same manner we can find w_{12} as follows:

$w_{12} = max\{0, P_{12} - (2\epsilon + 4\delta)\}$

As we can see, in calculation of w_{11} we require w_{12}, but we do not require w_{11} in calculation of w_{12}. When we add up w_{11} and w_{12} by using the properties of the max term we obtain the following relationship:

$w_{11} + w_{12} = max\{w_{12}, w_{12} + P_{11} - w_{12} - (2\epsilon + 4\delta)\}$

$= max\{max\{0, P_{12} - (2\epsilon + 4\delta)\}, P_{11} - (2\epsilon + 4\delta)\}$

$= max\{0, P_{12} - (2\epsilon + 4\delta), P_{11} - (2\epsilon + 4\delta)\}$

Thus the cycle time of $S2$ is the following:

$C_t^{S2} = 6\epsilon + 8\delta + max\{0, P_{12} - (2\epsilon + 4\delta), P_{11} - (2\epsilon + 4\delta)\}$

The calculation of all other robot move cycles are made in the same way.

3.1.2 Modeling the Problem

Now consider the problem of allocating the tasks to the machines for these two 1-unit robot move cycles. For $S1$ there is no allocation problem since whatever the allocation of the tasks is, the cycle time is the same. However, for $S2$ there is an allocation problem to be solved. Since the number of allocation types to be used in producing the parts in the queue is not known a priori, this

makes the problem more complicated to solve. There may be one-allocation type, which means that all of the parts will be processed according to the same allocation type or there may be as much allocation types as the number of parts in the queue, which means that each part will be processed according to a different allocation type. If the problem is to find the allocation of tasks given that one-allocation type will used, the model can be constructed as follows:

$$\begin{aligned} \min \quad & C_t = 6\epsilon + 8\delta + max\{0, P_{11} - (2\epsilon + 4\delta), (P - P_{11}) - (2\epsilon + 4\delta)\} \\ \text{st} \quad & x_{i1} + x_{i2} = 1 \qquad i = 1, 2, \ldots, p \\ & x \in \{0, 1\} \end{aligned}$$

where

$$\begin{aligned} & P_{11} = t_1 x_{11} + t_2 x_{21} + \ldots + t_p x_{p1}, \\ & (P - P_{11)} = t_1 x_{12} + t_2 x_{22} + \ldots + t_p x_{p2} \end{aligned}$$

mboxand

$$x_{ij} = \begin{cases} 1, & \text{if task } i \text{ is allocated to } j^{th} \text{ machine} \\ 0, & \text{otherwise} \end{cases} \qquad i = 1, 2, \ldots p \quad \text{and} \quad j = 1, 2.$$

The objective function minimizes the cycle time while the only constraint states that the same task for one part cannot be allocated to both machines at the same time.

Although this is not a linear programming model we can linearize it as follows:

Let

$$z = 6\epsilon + 8\delta + max\{0, P_{11} - (2\epsilon + 4\delta), (P - P_{11}) - (2\epsilon + 4\delta)\}$$

We should adjust the constraints as follows:

$z - 6\epsilon - 8\delta \geq 0$

$z - 6\epsilon - 8\delta \geq P_{11} - (2\epsilon + 4\delta)$

$z - 6\epsilon - 8\delta \geq (P - P_{11}) - (2\epsilon + 4\delta)$

Resulting is the following ILP model:

$$
\begin{aligned}
\min \quad & z \\
\text{st} \quad & z \geq 6\epsilon + 8\delta \\
& z \geq P_{11} + 4\epsilon + 4\delta \\
& z \geq (P - P_{11}) + 4\epsilon + 4\delta \\
& x_{i1} + x_{i2} = 1 \quad i = 1, 2, \ldots, p \\
& x \in \{0, 1\}
\end{aligned}
$$

Now let us show how to model the problem for two-allocation type $S2$. The sequence of activities is still $A_0A_2A_1$, however, since the allocation type changes from one cycle to other, the processing time on the first and the second machines will change. Thus, in order to find the cycle time for this case we simply multiply $6\epsilon + 8\delta$ by 2, however, the rest depends on the allocation type and is as follows:

$$
\begin{aligned}
C_t = 12\epsilon + 16\delta \quad & + max\{0, (P - P_{22}) - (2\epsilon + 4\delta), P_{21} - (2\epsilon + 4\delta)\} \\
& + max\{0, (P - P_{21}) - (2\epsilon + 4\delta), P_{22} - (2\epsilon + 4\delta)\}
\end{aligned}
$$

where

$$
\begin{aligned}
& P_{21} = t_1x_{11} + t_2x_{21} + \ldots + t_px_{p1}, \\
& (P - P_{21}) = t_1x_{12} + t_2x_{22} + \ldots + t_px_{p2}, \\
& P_{22} = t_1y_{11} + t_2y_{21} + \ldots + t_py_{p1}, \text{ and} \\
& (P - P_{22}) = t_1y_{12} + t_2y_{22} + \ldots + t_py_{p2}
\end{aligned}
$$

So the mathematical model for this case is the following and a similar linearization will translate the model into a linear program.

$$
\begin{aligned}
\min \quad & C_t \\
\text{st} \quad & x_{i1} + x_{i2} = 1 \quad i = 1, 2, \ldots, p \\
& y_{i1} + y_{i2} = 1 \quad i = 1, 2, \ldots, p \\
& x \in \{0, 1\} \text{ and } y \in \{0, 1\}
\end{aligned}
$$

For using three or more allocation types, the cycle times and the corresponding linear programs can be found in the same manner. One way of solving this problem is to consider all possible robot move cycles and for

each robot move cycle consider all possible number of allocation types. For one robot move cycle the number of allocation types to be considered is as much as the number of parts in the queue. Thus, we have to develop and solve a different LP model for all possible number of allocation types, for each robot move cycle. Then by comparing the cycle times of all robot move cycles for which the optimal allocations are found, we can find the optimal allocation of tasks and the corresponding robot move cycle. However, since there may be a great number of parts in the queue and since there are a great number of robot move cycles when we include higher unit ones, solving the problem by this way is burdensome and furthermore, if we assume the number of parts in the queue to be infinite, solving the problem by this way is not possible. Thus, we need another way and in the next section we will describe our solution procedure for this problem.

The remainder of the chapter is organized as follows. In the next section, the task allocation problem is described. In section 3.3, we will introduce a new robot move cycle which is a result of assuming operational flexibility. In section 3.4, we comment on the results.

3.2 Solution procedure for task allocation problem

In this section we will review two additional robot move sequences from the literature so that we can represent higher unit robot move cycles. Next, we will prove Lemmas 3.1, 3.2 and 3.3, which will jointly show that the optimal solution to the problem of robotic cell scheduling with operational flexibility is one of the three robot move cycles which will be defined in the following paragraphs. In Theorems 3.1 and 3.2 we find the optimal number of allocation types for these three robot move cycles. The main result of the study will be provided with Theorem 3.3. This theorem will provide intervals where each of the three robot move cycles are optimal.

In addition to the 1-unit robot move cycles, $S1$ and $S2$, Hall et al. [11] define two more robot move sequences. They are not feasible robot move cycles by themselves, since the starting and the ending states depend on the preceding and following robot movements. Unless we know the preceding and the following robot movements, we cannot represent these cycles by the help of robot activities.

$S_{12} \equiv$ The transition movement of the robot from performing cycle $S1$ to $S2$. The robot uses cycle $S1$ during processing of part i on first machine, and cycle $S2$ during processing on the second machine.

$S_{21} \equiv$ The transition movement of the robot from performing cycle $S2$ to $S1$, in which, the robot uses cycle $S2$ during processing of part i on the first machine, and cycle $S1$ during processing on the second machine.

Hall et al. [11] show that for 2-machine case, we can represent any robot move cycle by the four robot move sequences: $S1$, $S2$, S_{12} and S_{21}. For example, a 3-unit robot move cycle can be represented as $S1S_{12}S_{21}$ and in terms of the robot activities this is equivalent to the sequence $A_0A_1A_2A_0A_1A_0A_2A_1A_2$. S_{12} and S_{21} are not robot move cycles by themselves and they can not be represented as a fixed sequence of activities. The activity sequence for these two depends on the preceding and the following robot move sequences. Thus, in this activity sequence we do not know which activity belongs to which cycle. The same cycle can also be represented as $S_{12}S_{21}S1$. The difference is the starting and the ending points of the cycle but both give the same cycle time. However, we are not entirely free in representing robot move cycles by these four sequences. For example, $S1$ cannot be followed by $S2$ since at the end of cycle $S1$ both machines are empty and the robot is in front of the input buffer in order to take a part and load machine 1. If $S1$ is followed by $S2$ then, while performing $S2$ the robot will try to unload machine 2 which is already empty and this violates the basic assumption of Crama et al. [5]. By the same methodology we can derive the following:

i- $S1$ can be followed by either $S1$ or S_{12}.

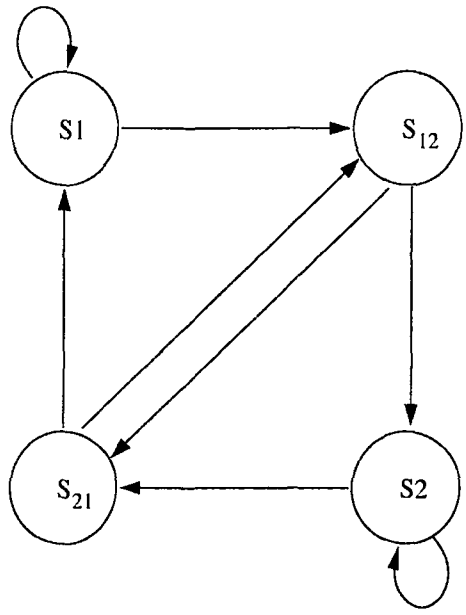

Figure 3.2: Transition Graph

ii- $S2$ can be followed by either $S2$ or S_{21}.

iii- S_{12} can be followed by either S_{21} or $S2$.

iv- S_{21} can be followed by either S_{12} or $S1$.

We can represent the set of all allowable moves by the transition graph shown in Figure 3.2.

Lemma 3.1 *(Hall et al. [11]) In any feasible robot move cycle the number of S_{12} sequences is equal to the number of S_{21} sequences.*

Proof. In Figure 2 we see that robot move sequence S_{12} must be preceded by $S1$ or S_{21}, and followed by $S2$ or S_{21}. Since the schedule is cyclic, it follows that the number of transitions from $S1$ into $S2$ (using S_{12}) must equal the number of transitions from $S2$ into $S1$ (using S_{21}) □

Before proceeding with the next lemma, let us note that though neither S_{12} nor S_{21} are complete cycles, both sequences $S_{12}S_{21}$ and $S_{21}S_{12}$ are 2-unit complete robot move cycles resulting from the robot activity sequence

$A_0A_1A_0A_2A_1A_2$.

Lemma 3.2 *Cycle time of any n-unit robot move cycle is the summation of the cycle times of three robot move cycles S_1, S_2 and $S_{12}S_{21}$ in corresponding numbers that form the n-unit robot move cycle.*

Proof. We previously mentioned that for 2-machine case, any n-unit robot move cycle is a repetitive sequence of $S1$, $S2$, S_{12} and S_{21}. Let us first analyze $S1$. The four possible sequences of $S1$ with the preceding and following robot move cycles with corresponding activity sequences are:
$S1S1S1$: $\ldots A_2(A_0A_1A_2)A_0\ldots$
$S1S1S_{12}$: $\ldots A_2(A_0A_1A_2)A_0\ldots$
$S_{21}S1S1$: $\ldots A_2(A_0A_1A_2)A_0\ldots$
$S_{21}S1S_{12}$: $\ldots A_2(A_0A_1A_2)A_0\ldots$
In all the cases the preceding activity of cycle $S1$ is A_2 and the following one is A_0. Then the initial and the final states of the system before and after performing cycle $S1$ are the same. Thus, if we remove the activities of $S1$, $(A_0A_1A_2)$, the remaining sequence will still correspond to a valid cycle. So we can remove the activities of all $S1$ cycles and consider the cycle time of the remaining activities and the total cycle time of the removed $S1$ cycles at the end to find the cycle time of the n-unit robot move cycle.

By following a similar sequence of arguments, the same result can be achieved for cycle $S2$ also. Thus, we can remove cycles $S1$ and $S2$, consider the rest, and add up the cycle times of the removed cycles at the end. After we remove cycles $S1$ and $S2$, we are left with an alternating sequence of S_{12} and S_{21}. Then the cycle time of the n-unit robot move cycle can be found by the summation of the cycle times of $S1$, $S2$ and $S_{12}S_{21}$ cycles that form the n-unit robot move cycle. □

Lemma 3.3 *At least one of the cycles $S1$, $S2$ or $S_{12}S_{21}$ has a cycle time that is less than or equal to the cycle time of any given n-unit robot move cycle.*

Proof. Let us consider any n-unit robot move cycle. This cycle can be expressed as a vector (b_1, b_2, b_3), where b_1 (respectively, b_2, b_3) is the number of $S1$ (respectively, $S2$, $S_{12}S_{21}$) cycles used to form the n-unit robot move cycle. Note that $S1$ and $S2$ are 1-unit robot move cycles and $S_{12}S_{21}$ is a 2-unit robot move cycle. Suppose that C_t^n is the total cycle time of the n-unit robot move cycle to produce n parts. So the cycle time to produce one part with this robot move cycle is $(\frac{C_t^n}{n})$. Using Lemma 3.2 we have:

$$\frac{C_t^n}{n} = \frac{(b_1)(\frac{C_t^{S1}}{1}) + (b_2)(\frac{C_t^{S2}}{1}) + (b_3)(\frac{C_t^{S_{12}S_{21}}}{2})}{b_1 + b_2 + b_3}$$

Let us choose among the robot move cycles forming the n-unit robot move cycle the one with minimum cycle time to produce one part. Assume without loss of generality that this cycle is $S1$. Then,
$C_t^{S1} \leq C_t^{S2}$ and $C_t^{S1} \leq \frac{C_t^{S_{12}S_{21}}}{2}$. Thus, we can derive the following inequality:

$$\frac{C_t^n}{n} = \frac{(b_1)(C_t^{S1}) + (b_2)(C_t^{S2}) + (b_3)(\frac{C_t^{S_{12}S_{21}}}{2})}{b_1 + b_2 + b_3} \geq \frac{(C_t^{S1})(b_1 + b_2 + b_3)}{b_1 + b_2 + b_3} \Rightarrow \frac{C_t^n}{n} \geq C_t^{S1}$$

□

Lemma 3.2 and Lemma 3.3 together state that the solution to IRC2$|k = 1, \delta_i = \delta, \epsilon_i = \epsilon|C_l$ is one of the three robot move cycles: $S1$, $S2$ or $S_{12}S_{21}$. In Theorem 3.3 we will extend this result and present regions of optimality for each of these three cycles. Prior to that, we shall prove Theorems 3.1 and 3.2, which will help us find the optimal number of allocation types to be used along with $S2$ and $S_{12}S_{21}$ cycles, respectively. Now let us consider an example which will shed light on why we need to find out the optimal number of allocation types.

Example 3.1 Let us consider using the 1-unit robot move cycle $S2$. Assume that we have 5 tasks to be completed and each of them has a processing time of 10. So $P = 50$. Take $\epsilon = 1$ and $\delta = 2$. If we assumed, as done in the current literature, that the allocation is fixed and first two tasks are allocated to the first machine and the next three are allocated to the second machine (the optimal allocation for this case) then the cycle time would be: $C_t =$

$6 \times 1 + 8 \times 2 + max\{0, (30 - 10), (20 - 10)\} = 42$. However, if two-allocation types are used: in the first cycle we allocate two tasks to the first machine and the next three to the second machine and in the second cycle we allocate three tasks to the first machine and the next two to the second machine. So the processing times in the first cycle will be $20 - 30$ and in the second cycle $30 - 20$. Then, the cycle time to produce two parts is:
$C_t = 12 \times 1 + 16 \times 2 + max\{0, (30-10), (30-10)\} + max\{0, (20-10), (20-10)\} = 74$
and thus, to produce one part the cycle time is 37 which is less than 42. This example clearly shows that fixing the processing times on each CNC machine unnecessarily limits the number of alternatives.

Theorem 3.1 *Under the robot move cycle $S2$, using two allocation types is better than using any number of allocation types. Moreover, the odd numbered allocation type alternatives follow a decreasing cycle time sequence. In the limit (assume unlimited number of parts in the queue), the cycle time of an odd numbered allocation type converges to that of using two-allocation type $S2$.*

Proof. Before we prove this theorem, recall that under the robot move cycle $S2$, the cycle time of using two allocation types is:

$$\begin{aligned} C_t = 12\epsilon + 16\delta \quad & + max\{0, (P - P_{22}) - (2\epsilon + 4\delta), P_{21} - (2\epsilon + 4\delta)\} \\ & + max\{0, (P - P_{21}) - (2\epsilon + 4\delta), P_{22} - (2\epsilon + 4\delta)\} \end{aligned} \quad (1)$$

We need to find the cycle time for k-allocation type $S2$. Consider the l^{th} robot move cycle of k-allocation type $S2$, $l \leq k$. Initially the second machine is loaded with a part which has $(l-1)^{th}$ allocation type. The robot activity sequence is $A_0 A_2 A_1$. Thus, the cycle time can be found as: $6\epsilon + 8\delta + w_{(l-1)2} + w_{l1}$
where w_{ij} is defined as the waiting time of the robot in front of machine j for the part which has i^{th} allocation type to be processed. More specifically:

$w_{(l-1)2} = max\{0, (P - P_{k(l-1)}) - (2\epsilon + 4\delta)\}$ and $w_{l1} = max\{0, P_{kl} - (2\epsilon + 4\delta + w_{(l-1)2})\}$

After a few manipulations the cycle time can be simplified as:

$$C_t = 6\epsilon + 8\delta + max\{0, (P - P_{k(l-1)}) - (2\epsilon + 4\delta), P_{kl} - (2\epsilon + 4\delta)\}$$

If $l = 1$, then $(l-1)^{th}$ allocation is the k^{th} allocation type since after the k^{th} allocation type, the first allocation type will be used again. Thus, if we add up the cycle times of all robot move cycles in a k-allocation type $S2$, the cycle time becomes:

$$\begin{aligned} C_t = 6(k)\epsilon + 8(k)\delta \quad & +max\{0, P_{k1} - (2\epsilon + 4\delta), (P - P_{kk}) - (2\epsilon + 4\delta)\} \\ & +max\{0, P_{k2} - (2\epsilon + 4\delta), (P - P_{k1}) - (2\epsilon + 4\delta)\} \\ & +max\{0, P_{k3} - (2\epsilon + 4\delta), (P - P_{k2}) - (2\epsilon + 4\delta)\} \\ & +\ldots \\ & +max\{0, P_{kk} - (2\epsilon + 4\delta), (P - P_{k(k-1)}) - (2\epsilon + 4\delta)\} \end{aligned} \tag{2}$$

Let us define Max_i as the i^{th} max term in the above formulation.

Our aim is to minimize C_t. We need to consider the following cases:

1 - In the ideal situation, it is possible to allocate the tasks to these two machines such that the sum of the task times that are allocated to first machine is equal to the sum of the task times allocated to second machine, i.e., $P_{ki} = \frac{P}{2}$ $\forall i \in [1 \ldots k]$. In this case the cycle time simplifies to:

$$C_t = \begin{cases} 6\epsilon + 8\delta & \text{if } \frac{P}{2} \leq 2\epsilon + 4\delta \\ 4\epsilon + 4\delta + \frac{P}{2} & \text{if } \frac{P}{2} \geq 2\epsilon + 4\delta \end{cases}$$

2 - Assume the tasks cannot be partitioned among the machines equally. We need to analyze the following sub-cases comparing each partition against the value of $2\epsilon + 4\delta$.

2.1 - If the tasks can be divided among the two machines such that each machine has a total task time of at most $2\epsilon + 4\delta$, i.e., $P_{ki} \leq 2\epsilon + 4\delta$ and $P - P_{ki} \leq 2\epsilon + 4\delta$ for all i, then each Max_i term vanishes in (2) and we have

$$C_t = 6\epsilon + 8\delta$$

2.2 - If there is a possible partition such that both $P_{ki} \geq 2\epsilon + 4\delta$ and $P - P_{ki} \geq 2\epsilon + 4\delta$ for all $i \in [1 \ldots n]$, then analyzing the cross relation between Max_i and Max_{i+1} in (2), the best allocation would simply

be to set $P_{ki} = P - P_{ki-1}\ \forall i \in [2 \ldots k]$ and $P_{k1} = P - P_{kk}$. With this allocation we have:

$$Max_i = \begin{cases} P_{k1} - (2\epsilon + 4\delta) & \text{for } i \text{ odd} \\ (P - P_{k1}) - (2\epsilon + 4\delta) & \text{for } i \text{ even} \end{cases} \quad 1 \leq i \leq k$$

Hence,

$$C_t = \begin{cases} 6\epsilon + 8\delta + \frac{P-(4\epsilon+8\delta)}{2} & \text{if } k \text{ is even} \\ 6\epsilon + 8\delta + \frac{k-1}{2k}(P - (4\epsilon + 8\delta)) + \frac{P_{k1}-(2\epsilon+4\delta)}{k} & \text{if } k \text{ is odd} \end{cases}$$

and

$$\lim_{k \to \infty,\, k \text{ odd}} C_t = 6\epsilon + 8\delta + \frac{P - (4\epsilon + 8\delta)}{2}$$

2.3 If the only possibility is one partition having a total task time of at least $2\epsilon + 4\delta$ and the other one at most $2\epsilon + 4\delta$, a similar analysis yields the same allocation as in case 2.2 optimum, i.e., $P_{ki} = P - P_{ki-1}\ \forall i \in [2 \ldots k]$ and $P_{k1} = P - P_{kk}$. Now assume without loss of generality that $P_{k1} \geq 2\epsilon + 4\delta$ and $(P - P_{k1}) \leq (2\epsilon + 4\delta)$. Then each max term in (2) becomes:

$$Max_i = \begin{cases} P_{k1} - (2\epsilon + 4\delta) & \text{for } i \text{ odd} \\ 0 & \text{for } i \text{ even} \end{cases} \quad 1 \leq i \leq k.$$

Hence, the cycle time of producing one unit is

$$C_t = \begin{cases} 6\epsilon + 8\delta + \frac{P_{k1}-(2\epsilon+4\delta)}{2} & \text{if } k \text{ is even} \\ 6\epsilon + 8\delta + \frac{k-1}{2k}(P_{k1} - (2\epsilon + 4\delta)) + \frac{P_{k1}-(2\epsilon+4\delta)}{k} & \text{if } k \text{ is odd} \end{cases}$$

and

$$\lim_{k \to \infty,\, k \text{ odd}} C_t = 6\epsilon + 8\delta + \frac{P_{k1} - (2\epsilon + 4\delta)}{2}$$

A simple comparison against equation (1) reveals that we achieve the statements of the theorem in all the cases. □

We shall now consider a numerical example for concreteness.

Example 3.2 Let us consider the same values as in Example 3.1. We have 5 tasks to be completed and each of them has a processing time of 10. So $P = 50$

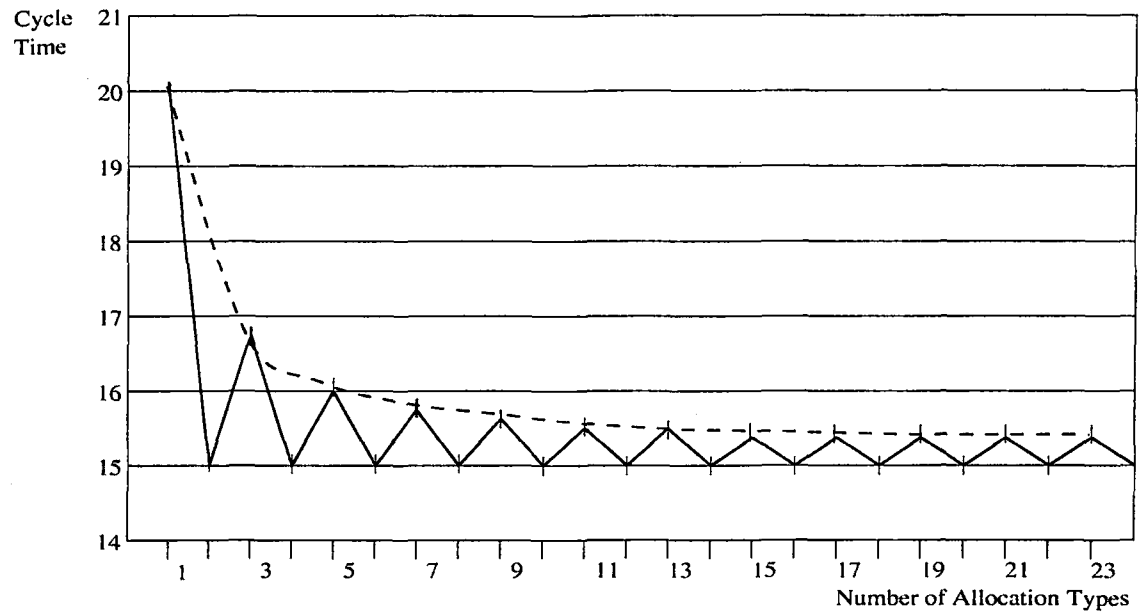

Figure 3.3: Convergence of odd number of different allocations to even number of different allocations for Example 10

and we cannot divide the tasks equally. $2\epsilon + 4\delta = 10$ and we can divide the tasks such that both partitions are greater than 10. So this definition leads us to case 2.2 in the proof of Theorem 3.1. The trivial solution is allocating 2 tasks to first machine and 3 tasks to second machine or vice versa. Let us consider how the results change when the number of allocation types is changed. Since $6\epsilon + 8\delta$ is the same for all of the terms, let us consider only the *max* terms.
Using one-allocation type S2: $30 - 10 = 20$
Using two : $\frac{50-20}{2} = 15$
Using three : $\frac{2(30-10)+50-30-10}{3} = \frac{50}{3}$
Using four : $\frac{50-20}{2} = 15$
Using five : $\frac{3(30-10)+2(50-30-10)}{5} = 16$
Using seven : $\frac{110}{7}$
Using nine : $\frac{140}{9}$
$\vdots$

Figure 3.3 depicts the fact that the cycle time of using odd number allocation type $S2$ converges to that of using two-allocation type $S2$.

In what follows we will prove a similar statement for 2-unit robot move

cycle, $S_{12}S_{21}$, however, we first need to define the decision variable for cycle $S_{12}S_{21}$ that parallels P_{ij} of $S2$.

Definition 3.1 *Consider the robot move cycle $S_{12}S_{21}$ in which i allocation types are available. Q_{ij} is the processing time of a part which has j^{th} allocation type on machine 1, where $j \leq i$. Note that under this setting, the processing time of this part on machine 2 is $P - Q_{ij}$.*

Theorem 3.2 *For 2-unit robot move cycle $S_{12}S_{21}$ using two allocation types is better than using any number of allocation types.*

Proof. The cycle time for one-allocation type $S_{12}S_{21}$ can be derived as:

$$C_t = 12\epsilon + 14\delta + Q_{11} + (P - Q_{11}) + w_{11} + w_{12}$$

where, $w_{11} = max\{0, Q_{11} - (2\epsilon + 4\delta + w_{12})\}$ and $w_{12} = max\{0, (P - Q_{11}) - (2\epsilon + 4\delta)\}$. In other words,

$$C_t = 12\epsilon + 14\delta + P + max\{0, Q_{11} - (2\epsilon + 4\delta), (P - Q_{11}) - (2\epsilon + 4\delta)\}$$

Assume without loss of generality that $Q_{11} \geq (P - Q_{11})$. So we have two cases:

1 - If we can partition the tasks such that $Q_{11} \leq 2\epsilon + 4\delta$ then the cycle time is
$C_t = 12\epsilon + 14\delta + P$

2 - If $Q_{11} \geq 2\epsilon + 4\delta$ then the cycle time is

$$C_t = 12\epsilon + 14\delta + P + \underbrace{Q_{11} - (2\epsilon + 4\delta)}_{\geq 0} \geq 12\epsilon + 14\delta + P$$

Now consider the case when we have two different allocation types and the parts are processed according to these two types in turn;

$$C_t = 12\epsilon + 14\delta + Q_{22} + (P - Q_{21}) + w_{11} + w_{22}$$

where $w_{11} = max\{0, Q_{21} - (2\epsilon + 4\delta + w_{22})\}$ and $w_{22} = max\{0, (P - Q_{22}) - (2\epsilon + 4\delta)\}$. In other words,

$$C_t = 12\epsilon + 14\delta + Q_{22} + (P - Q_{21}) + max\{0, Q_{21} - (2\epsilon + 4\delta), (P - Q_{22}) - (2\epsilon + 4\delta)\}$$

A trivial solution is found by letting $Q_{21} = (P - Q_{22}) = P$, thus $(P - Q_{21}) = Q_{22} = 0$. In words, in the first cycle all of the tasks are allocated to the first machine and in the second cycle all of the tasks are allocated to the second machine. The cycle time for this solution is:

$$C_t = \begin{cases} 12\epsilon + 14\delta & P \leq 2\epsilon + 4\delta \\ 10\epsilon + 10\delta + P & P \geq 2\epsilon + 4\delta \end{cases}$$

Comparing the using of one-allocation type with the using of two-allocation types, one can easily verify that for one-allocation type, in all of the cases C_t is always greater than or equal to $12\epsilon + 14\delta + P$ and for two-allocation type, in all of the cases C_t is always less than or equal to $12\epsilon + 14\delta + P$. Thus, we say that using two-allocation type always gives better results than using one-allocation type.

Now let us consider using more than two allocation types. For using an even number of allocation types, say $2k$, since $S_{12}S_{21}$ is a 2-unit robot move cycle, we will consider repeating the robot move cycle k times to produce all $2k$ parts. Let us consider the i^{th} repetition ($i \leq k$) of the robot move cycle. In this repetition allocation types $(2i - 1)$ and $2i$ will be considered. So the cycle time of the i^{th} cycle is:

$$\begin{aligned} C_t = \; & 12\epsilon + 14\delta + Q_{(2k)(2i)} + (P - Q_{(2k)(2i-1)}) \\ & + max\{0, Q_{(2k)(2i-1)} - (2\epsilon + 4\delta), (P - Q_{(2k)(2i)}) - (2\epsilon + 4\delta)\} \end{aligned}$$

The optimal allocation for this cycle is letting $Q_{(2k)(2i)=0}$ and $P - Q_{(2k)(2i-1)} = 0$ and this yields:

$$C_t = \begin{cases} 12\epsilon + 14\delta & P \leq 2\epsilon + 4\delta \\ 10\epsilon + 10\delta + P & P \geq 2\epsilon + 4\delta \end{cases}$$

This allocation is the same as two-allocation type $S_{12}S_{21}$. Thus we conclude that using $2k$-allocation type $S_{12}S_{21}$ is the same as using two-allocation type $S_{12}S_{21}$ and repeating it k times.

Now let us consider using an odd number of allocation types, say $2k-1$. The following allocation types will be used in each repetition of $S_{12}S_{21}$:

$\overbrace{1 \text{ and } 2}^{1}, \overbrace{3 \text{ and } 4}^{2}, \ldots, \overbrace{(2k-1) \text{ and } 1}^{k}, \overbrace{2 \text{ and } 3}^{(k+1)}, \ldots, \overbrace{(2k-2) \text{ and } (2k-1)}^{(2k-1)}$

The cycle times of cycles if we consider allocation types $(i-1)$ and i, and i and $(i+1)$ are as follows respectively:

$C_t = 12\epsilon + 14\delta + Q_{(2k-1)i} + (P - Q_{(2k-1)(i-1)}) + w_{(i-1)1} + w_{i2}$

$C_t = 12\epsilon + 14\delta + Q_{(2k-1)(i+1)} + (P - Q_{(2k-1)i}) + w_{i1} + w_{(i+1)2}$

If we add up these two cycle times the resulting one is:

$C_t = 24\epsilon + 28\delta + P + Q_{(2k-1)(i+1)} + (P - Q_{(2k-1)(i-1)}) + w_{(i-1)1} + w_{i1} + w_{i2} + w_{(i+1)2}$

As we can see, in this summation, the processing times of the part with the i^{th} allocation type on the first and second machines add up to P. Thus, if we take the summation of the cycle times of all cycles, all of the processing times of all allocation types on the first and second machine will add up to $(2k-1)P$. So if we calculate the cycle time to produce one part with $(2k-1)$-allocation type $S_{12}S_{21}$:

$$C_t = 12\epsilon + 14\delta + P + \frac{w_{11} + w_{12} + w_{21} + w_{22} + \ldots + w_{(2k-1)1} + w_{(2k-1)2}}{2k-1} \geq 12\epsilon + 14\delta + P$$

In conclusion, we say that, two-allocation type $S_{12}S_{21}$ gives the optimal result for 2-unit robot move cycles. □

Now we can present the main results of this chapter, but first note that, in Theorem 3.1 we prove that the optimal number of allocation types for $S2$ is two and the optimal allocation is reached by letting $P_{21} = P - P_{22}$ and $P - P_{21} = P_{22}$.

Theorem 3.3 *For $IRC2|k = 1, \delta_i = \delta, \epsilon_i = \epsilon|C_t$ with the assumption of operational flexibility, one of the robot move cycles $S1$, $S2$ or $S_{12}S_{21}$ gives the minimum cycle time among the robot move cycles that are reported in the literature and the following holds:*

1- If $0 \leq P \leq \delta$ then $S1$ gives the minimum cycle time,

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

3- If $P \geq 2\epsilon + 6\delta$ *then*

3.1- If $(P_{21} \geq 2\epsilon + 4\delta) \wedge ((P - P_{21}) \leq 2\epsilon + 4\delta)$ *then*

3.1.1- If $P \leq P_{21} + 2\delta$ *then* $S_{12}S_{21}$ *gives the minimum cycle time,*

3.1.2- Else $S2$ *gives the minimum cycle time,*

3.2- Else $S2$ *give the minimum cycle time.*

Proof. Lemma 3.2 and Lemma 3.3 jointly prove that for IRC2$|k = 1, \delta_i = \delta, \epsilon_i = \epsilon|C_t$ with the assumption of operational flexibility, either $S1$ or $S2$ or $S_{12}S_{21}$ gives the minimum cycle time. Now let us consider each region given in the theorem separately. For $S2$ and $S_{12}S_{21}$, it is proved in Theorems 3.1 and 3.2 respectively that using two allocation types is optimal.

1- If $0 \leq P \leq \delta$, the cycle times of $S1$, $S2$ and $S_{12}S_{21}$(to produce one part) are as follows:
$C_t^{S1} = 6\epsilon + 6\delta + P$
$C_t^{S2} = 6\epsilon + 8\delta$
$C_t^{S_{12}S_{21}} = 6\epsilon + 7\delta.$
Since in this region $P \leq \delta$ then $C_t^{S1} \leq 6\epsilon + 7\delta$. Thus we conclude that in this region $S1$ gives the minimum cycle time.

2- If $\delta \leq P \leq 2\epsilon + 6\delta$ then the cycle times are:
$C_t^{S1} = 6\epsilon + 6\delta + P$

$$C_t^{S2} = \begin{cases} 6\epsilon + 8\delta, & \text{if } (P_{21} \leq 2\epsilon + 4\delta) \wedge ((P - P_{21}) \leq 2\epsilon + 4\delta) \\ 6\epsilon + 8\delta + P_{21} - (2\epsilon + 4\delta), & \text{if } P_{21} > 2\epsilon + 4\delta \end{cases}$$

$$C_t^{S_{12}S_{21}} = \begin{cases} 6\epsilon + 7\delta, & \text{if } P \leq 2\epsilon + 4\delta \\ 5\epsilon + 5\delta + \frac{P}{2}, & \text{if } P \geq 2\epsilon + 4\delta \end{cases}$$

A simple comparison reveals that $C_t^{S_{12}S_{21}}$ is the minimum for this range.

3- If $P \geq 2\epsilon + 6\delta$ then the cycle times are:
$C_t^{S1} = 6\epsilon + 6\delta + P$

$$C_t^{S2} = \begin{cases} 6\epsilon + 8\delta, & \text{if } (P_{21} \leq 2\epsilon + 4\delta) \wedge ((P - P_{21}) \leq 2\epsilon + 4\delta) \\ 4\epsilon + 4\delta + \frac{P}{2}, & \text{if } (P_{21} \geq 2\epsilon + 4\delta) \wedge ((P - P_{21}) \geq 2\epsilon + 4\delta) \\ 5\epsilon + 6\delta + \frac{P_{21}}{2}, & \text{if } (P_{21} > 2\epsilon + 4\delta) \wedge ((P - P_{21}) < 2\epsilon + 4\delta) \end{cases}$$

$C_t^{S_{12}S_{21}} = 5\epsilon + 5\delta + \frac{P}{2}$

Thus, in this region, if $(P_{21} > 2\epsilon + 4\delta) \wedge ((P - P_{21}) < 2\epsilon + 4\delta)$, $S_{12}S_{21}$ is better than $S2$ if and only if $P \leq P_{21} + 2\delta$. For all other cases, S2 is better than $S_{12}S_{21}$.

This completes the proof. □

The following numerical example is intended to depict the results obtained thus far.

Example 3.3 Let us take $\epsilon = 10$ and $\delta = 10$. Since there is no allocation problem for $S1$ and $S_{12}S_{21}$, their cycle times can be represented graphically as shown in Figure 3.4. On the other hand, the best allocation for $S2$ was to let $P_{21} = (P - P_{22})$. If $P \leq 2\epsilon + 4\delta$, then $(P_{21} \leq 2\epsilon + 4\delta) \wedge ((P - P_{21}) \leq 2\epsilon + 4\delta)$, thus the cycle time is $6\epsilon + 8\delta$. If $P \geq 2\epsilon + 4\delta$, the cycle time of $S2$ depends on the allocation, but we can still find upper and lower bounds for it. The upper bound is obtained by letting $P_{21} = (P - P_{22}) = P$ and thus $P_{22} = (P - P_{21}) = 0$, so the cycle time is $5\epsilon + 5\delta + \frac{P}{2}$. The lower bound is obtained by letting $P_{21} \leq 2\epsilon + 4\delta$, and $(P - P_{22}) \leq 2\epsilon + 4\delta$, so the cycle time is $6\epsilon + 8\delta$. However, this condition can only hold for $P \leq 4\epsilon + 8\delta$. After this point the lower bound is obtained by letting $P_{21} \geq 2\epsilon + 4\delta$, and $(P - P_{22}) \geq 2\epsilon + 4\delta$, and the cycle time is $4\epsilon + 4\delta + \frac{P}{2}$.

In summary, Figure 3.4 shows how the cycle times of the three robot move cycles change with respect to total processing time. The boldly dashed lower envelope shows the optimal solution to the problem. As we can see, for $P \geq 2\epsilon + 4\delta$ the cycle time of $S2$ may take any value between the upper and lower bounds shown as the dashed region depending on the allocation of tasks. Thus, the solution changes according to the allocation of tasks after the point where the cycle times of $S_{12}S_{21}$ and $S2$ intersect, that is, for $P > 2\epsilon + 6\delta$ and the solution can be anywhere between the two boldly dashed lines in the graph depending on the allocation of the tasks. For $P \leq 2\epsilon + 6\delta$, we know the exact solution and there is no allocation problem in this region.

In the following section we will propose a new robot move cycle which is a

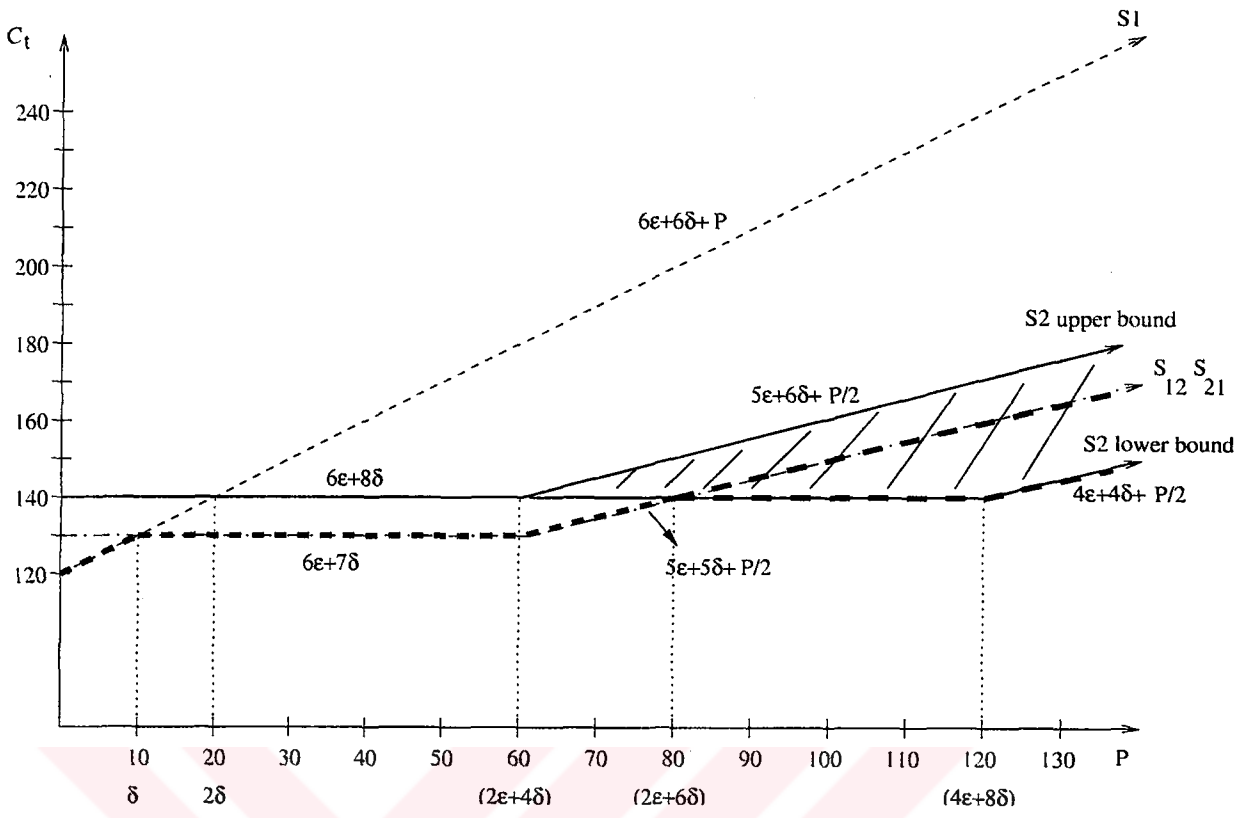

Figure 3.4: Change of cycle times according to P

direct result of assuming operational flexibility and we will prove that this new cycle is better than all of the robot move cycles reported in the literature.

3.3 A new robot move cycle

Our assumption in the problem definition was that, all of the tasks can be allocated to only one machine and each machine has the capacity of performing all of the tasks. This problem can also be viewed as a parallel machine scheduling problem with a common server as described in Hall et al.[13]. The operational flexibility of CNC machines allows the possibility of a new cycle. With our new cycle the robot does not have to transfer parts between two consecutive machines but may transfer from any machine labelled i to any other labelled j such that $i \leq j$. We require this forward behavior since the

parts cannot turn back from a machine to input buffer or from output buffer to a machine or input buffer etc. This new behavior necessitates definition of new robot activities. Let,

A_{ij} = The robot activity in which the robot unloads machine i, transfers the part from machine i to machine j, and loads machine j.

By using these activities, we have a new cycle.

Definition 3.2 *$A_{01}A_{02}A_{13}A_{23}$: The robot first loads machine 1 and later loads machine 2, all of the tasks will be processed by only one machine so, next activity for the robot is to turn back to machine 1, if necessary wait in front of this machine and unload the machine, transfer the part to output buffer, then turn back to machine 2, if necessary wait for the machine to finish the processing of the part, unload machine 2 and transfer the part to output buffer.*

Note that no additional assumptions are made in the definition of this new cycle. The cycle time is given by: $C_t = 8\epsilon + 12\delta + w_1 + w_2$ where w_1 is the waiting time of the robot in front of the first machine and w_2 is the waiting time of the robot in front of the second machine. In other words, $w_1 = max\{0, P - (2\epsilon + 4\delta)\}$ and $w_2 = max\{0, P - (2\epsilon + 4\delta + w_1)\}$ and the cycle time becomes: $C_t = 8\epsilon + 12\delta + max\{0, P - (2\epsilon + 4\delta)\}$. Since two parts are produced at the end of this cycle, our new cycle is a 2-unit cycle.

The next theorem will establish another important contribution of this thesis.

Theorem 3.4 *For $IRC2|k = 1, \delta_i = \delta, \epsilon_i = \epsilon|C_t$, the new robot move cycle, $A_{01}A_{02}A_{13}A_{23}$, gives the minimum cycle time.*

Proof. In order to prove this, let us compare the new robot move cycle with the three robot move cycles of Theorem 3.3, which are proved to give the minimum cycle time among the robot move cycles that are reported in the literature. First let us compare $S1$ with the new cycle. Since $S1$ is a one unit

cycle and our new cycle is a two unit cycle, in order to compare these two, let us multiply the cycle time of $S1$ with 2. Thus, we will compare $12\epsilon + 14\delta + 2P$ with $8\epsilon + 12\delta + max\{0, P - (2\epsilon + 4\delta)\}$. We easily conclude that the new cycle is always better than $S1$.

While analyzing $S2$ we showed that the optimal partition, if possible, is the equal one and we also proved that two-allocation type $S2$ is better than any other number of allocation types. Thus, let us use two-allocation type $S2$, and let us assume we divided the tasks equally (even if this is not attainable according to the given task times, we get a lower bound for the cycle time of $S2$). The cycle time for two-allocation $S2$ was:

$$\begin{aligned} C_t = 12\epsilon + 16\delta \quad & + max\{0, P_{21} - (2\epsilon + 4\delta), (P - P_{22}) - (2\epsilon + 4\delta)\} \\ & + max\{0, (P - P_{21}) - (2\epsilon + 4\delta), P_{22} - (2\epsilon + 4\delta)\} \end{aligned}$$

and to get the lower bound we use $P/2$ for P_{21}, $(P - P_{21})$, P_{22}, and $(P - P_{22})$. In particular,

1 - If $P \leq 2\epsilon + 4\delta$, for $S2$ the cycle time is $12\epsilon + 16\delta$ and for the new cycle the cycle time is $8\epsilon + 12\delta$. Thus, for this case new cycle is better than $S2$.

2 - If $P \geq 2\epsilon + 4\delta$ but $P/2 \leq 2\epsilon + 4\delta$ then for $S2$, $C_t = 12\epsilon + 16\delta$ and for the new cycle, $C_t = 6\epsilon + 8\delta + P$. Thus, the new cycle is better than $S2$ for this case as well.

3 - If $P/2 \geq 2\epsilon + 4\delta$, then the cycle time for $S2$ is $C_t = 8\epsilon + 8\delta + P$ and the cycle time for the new cycle is $C_t = 6\epsilon + 8\delta + P$. Once more, the new cycle is better than $S2$.

Lastly, let us compare the new cycle with $S_{12}S_{21}$. For two-allocation type $S_{12}S_{21}$ the cycle time was: $C_t = 12\epsilon + 14\delta + P_{12} + P_{21} + max\{0, P_{22} - (2\epsilon + 4\delta), P_{11} - (2\epsilon + 4\delta)\}$.
We also proved that the ideal is achieved by letting $P_{12} = P_{21} = 0$ and $P_{11} = P_{22} = P$. Thus, the cycle time is $C_t = 12\epsilon + 14\delta + max\{0, P - (2\epsilon + 4\delta)\}$.

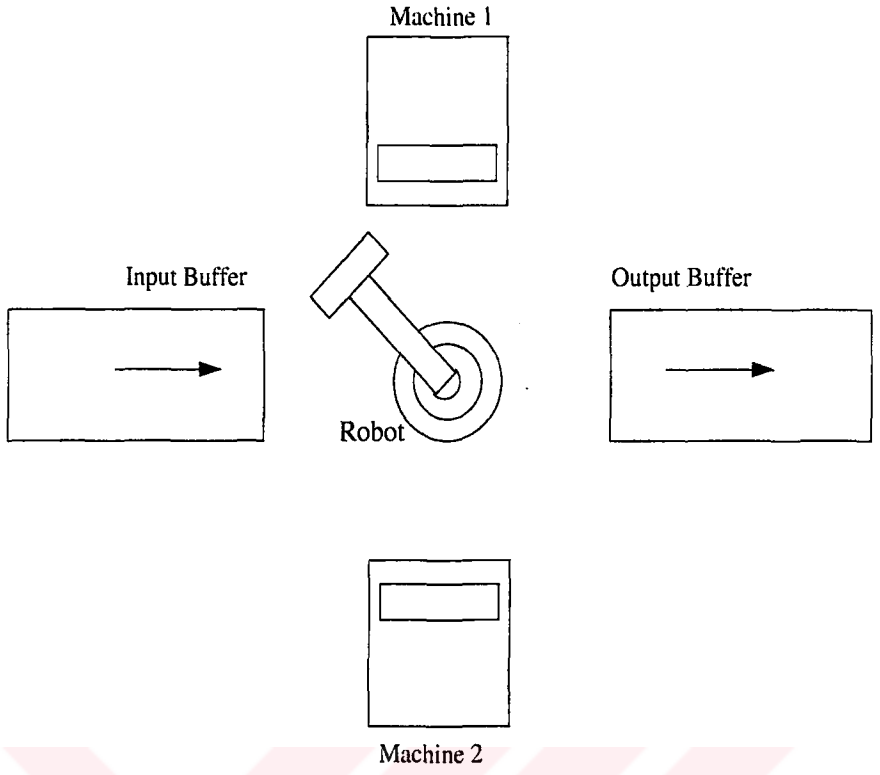

Figure 3.5: Robot Centered Cell Layout

A simple analysis shows that the new cycle is also better than $S_{12}S_{21}$. This completes the proof. □

In the beginning of this chapter, we assumed an in-line robotic cell. For this layout we proved that if we assume operational flexibility a new cycle, $A_{01}A_{02}A_{13}A_{23}$, gives better results than all of the common cycles reported in the literature. Now we will prove that if the layout of the cell is changed to a robot-centered cell, we reduce the cycle time even further. The new layout can be seen in Figure 3.5.

Theorem 3.5 *Changing the layout from in-line robotic cell to robot-centered cell reduces the cycle time of the new cycle even further while the cycle times of the other robot move cycles remain the same.*

Proof. We first show that changing the layout reduces the cycle time for the new robot move cycle. If we use the in-line robotic cell, the cycle time for

the new cycle is: $C_t = 8\epsilon + 12\delta + max\{0, P - (2\epsilon + 4\delta)\}$. If we use a robot centered cell, the new cycle time becomes $C_t = 8\epsilon + 12\delta + w_{11} + w_{12}$ where $w_{11} = max\{0, P - (2\epsilon + 4\delta)\}$ and $w_{12} = \{0, P - (2\epsilon + 4\delta + w_{11})\}$. More compactly, $C_t = 8\epsilon + 10\delta + max\{0, P - (2\epsilon + 4\delta)\}$. We conclude that changing the layout proves to be favorable.

We now proceed to show that the new layout does not change the cycle times of the original three cycles.

i - For $S1$, the cycle time for in-line robotic cell layout was $C_t = 6\epsilon + 6\delta + P$ and with the robot centered cell layout the cycle time becomes $C_t = 6\epsilon + 6\delta + a + b$ where a and b are the processing times of the parts on the first and the second machines respectively. Hence, $C_t = 6\epsilon + 6\delta + P$ so the cycle time does not change with change of the layout.

ii - For $S2$ we proved that the optimal is using two different types of allocation. Clearly, this result does not change with layout. The cycle time with in-line robotic cell was:

$$\begin{aligned} C_t = 12\epsilon + 16\delta \quad & + max\{0, P_{21} - (2\epsilon + 4\delta), (P - P_{22}) - (2\epsilon + 4\delta)\} \\ & + max\{0, (P - P_{21}) - (2\epsilon + 4\delta), P_{22} - (2\epsilon + 4\delta)\} \end{aligned}$$

The cycle time with the robot centered cell layout is: $C_t = 12\epsilon + 16\delta + w_{11} + w_{12} + w_{22} + w_{21}$ where $w_{11} + w_{22} = max\{0, P_{21} - (2\epsilon + 4\delta), (P - P_{22}) - (2\epsilon + 4\delta)\}$ and $w_{12} + w_{21} = max\{0, (P - P_{21}) - (2\epsilon + 4\delta), P_{22} - (2\epsilon + 4\delta)\}$. Thus the cycle time for $S2$ does not change with the change of layout.

iii - For $S_{12}S_{21}$ using two different allocation types (optimal) with the in-line robotic cell layout the cycle time is:
$C_t = 12\epsilon + 14\delta + Q_{22} + (P - Q_{21}) + max\{0, Q_{21} - (2\epsilon + 4\delta), (P - Q_{22}) - (2\epsilon + 4\delta)\}$.
The cycle time with the robot centered cell layout is: $C_t = 12\epsilon + 14\delta + Q_{22} + (P - Q_{21}) + w_{11} + w_{22}$ where, $w_{11} = max\{0, Q_{21} - (2\epsilon + 4\delta + w_{22})\}$ and $w_{22} = max\{0, (P - Q_{22}) - (2\epsilon + 4\delta)\}$. Since $w_{11} + w_{22} = max\{0, Q_{21} - (2\epsilon + 4\delta), (P - Q_{22}) - (2\epsilon + 4\delta)\}$ we conclude that the change of layout does not change the cycle time of $S_{12}S_{21}$. □

3.4 Conclusion

In this chapter, we considered the two machine identical parts robotic cell scheduling problem with operational flexibility. Since the machines in a robotic cell are predominantly CNC machines, the assumption of operational flexibility is a realistic one. With this definition of the problem, each part has a number of tasks to be processed and the problem is to allocate these tasks to the machines and is to find the corresponding robot move cycle that jointly minimize the cycle time.

This problem could also be solved by modeling and solving ILP models, however, since there may be infinite number of models to be developed and solved this procedure may not be tractable and it is very burdensome. Therefore, we presented a new solution procedure. First of all in lemmas 3.1, 3.2 and 3.3, we proved that the number of robot move cycles to be considered is three. In theorem 3.1 and 3.2 we proved that the optimal number of allocation types to be used with $S2$ and $S_{12}S_{21}$ is two. The main results of this chapter are given in Theorems 3.3, 3.4 and 3.5. In Theorem 3.3, we proved that the optimal robot move cycle is not necessarily a 1-unit cycle as Sethi et al. [34] prescribes, but a 2-unit robot move cycle may also be optimal for some parameter inputs and we provided the regions of optimality for each robot move cycle. In Theorem 3.4, we proposed a new robot move cycle and proved that this cycle is not only simple and practical, but also gives the best cycle time among all of the robot move cycles reported in the literature. In Theorem 3.5, we proved that changing the layout from an in-line robotic cell to a robot-centered cell reduces the cycle time of the new cycle even further while the rest remains the same.

In the next chapter we will consider a variation of the problem considered in this chapter. CNC machines possess an operational flexibility, that is, these machines are capable of performing different tasks. However, these tasks are performed by different tools and these tools are placed in the tool magazines of the machines. The tool magazines have a limited capacity. Thus assuming that the machines are capable of performing all of the tasks may not be a realistic

assumption in certain cases. Thus, in the next chapter we will assume that a given set of tasks can only be processed in the first machine while certain tasks can only be processed in the second machine, and the remaining tasks can be processed on both machines. The problem to be considered is to find the allocation of the tasks that can be assigned to either one of the machines and to find the corresponding robot move cycle that jointly minimize the cycle time.

Chapter 4

Scheduling with Tooling Constraints

4.1 Introduction and Problem Definition

In this chapter, we will consider a variation of the problem of Chapter 3. It is well known that a robotic cell contains a robot system and programmable computer numerically controlled (CNC) machines (Noh and Herring 1988). These CNC machines have the capability of performing different tasks. However, they use different tools to perform these different tasks. The tools are placed and stored in the tool magazines of these machines. In the previous chapter we assumed that these machines have the capability to perform all of the tasks to produce a given part. In this chapter we assume that the CNC machines are equipped with a limited tool magazine capacity, and thus they are not capable of performing all of the required tasks but rather we assume that we have three sets of tasks. The first set, T_1, is the set of tasks that can only be processed on machine 1, T_2 is the set of tasks that can only be processed on the second machine, and T is the set of tasks that can be processed on both machines. So the new definition of the problem is to find the optimal robot move sequence with the corresponding allocation of tasks that are in set T that

jointly minimize the cycle time.

For this variation of the problem we proved that Lemmas 3.1, 3.2 and 3.3 of the previous chapter are still valid, which reduced the number of robot move cycles to be considered to three. We proved in Theorem 4.1 that for the robot move cycle $S2$, using two-allocation types is not optimal in some regions and we proved that in these regions using one-allocation type $S2$ is optimal. In Theorem 4.2, we proved that for robot move cycle $S_{12}S_{21}$ using two-allocation types is still optimal. The final solution of the considered problem is presented in Theorem 4.3 where we presented regions of optimality for each robot move cycle.

4.2 Solution Procedure

For this definition of the problem, we can also find the optimum solution by solving similar ILP models presented in Section 3.1. However, because of the reasons mentioned then, this is not an easy and efficient way to solve the problem. Thus, we will proceed in the same way and try to solve the problem algebraically. First of all we have to prove or disprove that the theorems and lemmas of the first part are still valid for this variation of the problem too. In this part we only changed the cycle times of the three robot move cycles but not the basic assumptions and the feasibility constraints. Since Lemmas 3.1, 3.2 and 3.3 do not depend on the cycle times, any change on the cycle times will not effect the validity of these lemmas. However, Theorems 3.1 and 3.2 depend on cycle times of the robot move cycles and any change in cycle times will affect these theorems. Thus, more detailed analysis is required for the validity of these theorems.

Lemma 3.1 states that in any feasible robot move cycle, the number of S_{12} cycles is equivalent to S_{21} cycles. This is still valid because the constraints about which cycles can be followed by which ones did not change in this variation of the problem and the proof of this lemma depends on the definition

of robot move cycle and these constraints.

Lemma 3.2 is still valid, again for the same reasoning. This lemma states that the cycle time of any n-unit robot move cycle is the summation of the cycle times of three robot move cycles in corresponding numbers that form the n-unit robot move cycle.

Lastly, Lemma 3.3 states that at least one of the cycles $S1$, $S2$ or $S_{12}S_{21}$ have a cycle time that is less than or equal to the cycle time of any n-unit robot move cycle. The proof of this lemma also remains the same with this definition of the problem.

Thus, Lemmas 3.1, 3.2 and 3.3 together reduced the number of robot move cycles to be considered to three. So, after proving the validity of these three lemmas, we conclude that the optimal solution to the considered problem is one of the robot move cycles, $S1$, $S2$ or $S_{12}S_{21}$. Before checking the validity of Theorems 3.1 and 3.2, we first define the following parameters:

P^1 : Total processing time of the tasks that are in set T_1,

P^2 : Total processing time of the tasks that are in set T_2,

P : Total processing time of the tasks that are in set T, that is, total processing time of the tasks that can be allocated to either one of the machines.

Theorem 3.1 states that under the robot move cycle $S2$, using two allocation types, is better than using any number of allocation types. First, let us show why this theorem is not valid any longer with an example:

Example 4.1 Let us assume $P^1 = 60$, $P^2 = 10$ and we have three tasks to be allocated to the machines with $t_i = 10$, $i = 1, 2, 3$. In the first part, the cycle time for a k-allocation type $S2$ was found and we can adjust this cycle time according to our new problem definition as follows:

$$
\begin{aligned}
C_t = \; & 6(k)\epsilon + 8(k)\delta \\
& + max\{0, P^1 + P_{k1} - (2\epsilon + 4\delta), P^2 + (P - P_{kk}) - (2\epsilon + 4\delta)\} \\
& + max\{0, P^1 + P_{k2} - (2\epsilon + 4\delta), P^2 + (P - P_{k1}) - (2\epsilon + 4\delta)\} \\
& + max\{0, P^1 + P_{k3} - (2\epsilon + 4\delta), P^2 + (P - P_{k3}) - (2\epsilon + 4\delta)\} \\
& + \ldots \\
& + max\{0, P^1 + P_{kk} - (2\epsilon + 4\delta), P^2 + (P - P_{k(k-1)}) - (2\epsilon + 4\delta)\}
\end{aligned}
$$

Putting the values of P^1 and P^2, we get:

$$
\begin{aligned}
C_t = \; & 6(k)\epsilon + 8(k)\delta \\
& + max\{0, 60 + P_{k1} - (2\epsilon + 4\delta), 10 + (P - P_{kk}) - (2\epsilon + 4\delta)\} \\
& + max\{0, 60 + P_{k2} - (2\epsilon + 4\delta), 10 + (P - P_{k1}) - (2\epsilon + 4\delta)\} \\
& + max\{0, 60 + P_{k3} - (2\epsilon + 4\delta), 10 + (P - P_{k3}) - (2\epsilon + 4\delta)\} \\
& + \ldots \\
& + max\{0, 60 + P_{kk} - (2\epsilon + 4\delta), 10 + (P - P_{k(k-1)}) - (2\epsilon + 4\delta)\}
\end{aligned}
$$

The optimal allocation for this problem is letting $P_{ki} = 0$ and $P - P_{ki} = 30$. So the result is the following:

$$
\begin{aligned}
C_t = 6(k)\epsilon + 8(k)\delta \quad & + max\{0, 60 - (2\epsilon + 4\delta), 40 - (2\epsilon + 4\delta)\} \\
& + max\{0, 60 - (2\epsilon + 4\delta), 40 - (2\epsilon + 4\delta)\} \\
& + \ldots \\
& + max\{0, 60 - (2\epsilon + 4\delta), 40 - (2\epsilon + 4\delta)\}
\end{aligned}
$$

Thus we conclude that for the given example one-allocation type is optimal. Following theorem shows us the optimal number of allocation types to be used with $S2$:

Theorem 4.1 *1. If either P^1 or $P^2 > \frac{P+P^1+P^2}{2}$ then for $S2$ using one-allocation is optimal.*

2. If both P^1 and $P^2 < \frac{P^1+P^2+P}{2}$, then using two-allocation type $S2$ is better than using any other number of allocation types.

Proof.

1. Assume w.l.o.g. $P^1 > \frac{P+P^1+P^2}{2}$ or in other words $P^1 > P+P^2$ then when we allocate P such that $P_{ki} = 0$ and $P - P_{ki} = P$, we have:

$$\begin{aligned} C_t = 6(k)\epsilon + 8(k)\delta \quad & +max\{0, P^1 - (2\epsilon + 4\delta), P + P^2 - (2\epsilon + 4\delta)\} \\ & +max\{0, P^1 - (2\epsilon + 4\delta), P + P^2 - (2\epsilon + 4\delta)\} \\ & +\ldots \\ & +max\{0, P^1 - (2\epsilon + 4\delta), P + P^2 - (2\epsilon + 4\delta)\} \end{aligned}$$

   and since $P^1 > P + P^2$ we have:

$$\begin{aligned} C_t = 6(k)\epsilon + 8(k)\delta \quad & +max\{0, P^1 - (2\epsilon + 4\delta)\} \\ & +max\{0, P^1 - (2\epsilon + 4\delta)\} \\ & +\ldots \\ & +max\{0, P^1 - (2\epsilon + 4\delta)\} \end{aligned}$$

   So any other allocation does not improve the cycle time.

2. Assume that in the optimal allocation for a k-allocation type $S2$, $P_{k1} = X_1$, $P_{k2} = X_2$, $P_{k3} = X_3$, ..., $P_{kk} = X_k$. So the cycle time for this cycle is as follows:

$$\begin{aligned} C_t = 6(k)\epsilon + 8(k)\delta \quad & +max\{0, P^1 + X_1 - (2\epsilon + 4\delta), P^2 + (P - X_k) - (2\epsilon + 4\delta)\} \\ & +max\{0, P^1 + X_2 - (2\epsilon + 4\delta), P^2 + (P - X_1) - (2\epsilon + 4\delta)\} \\ & +max\{0, P^1 + X_3 - (2\epsilon + 4\delta), P^2 + (P - X_2) - (2\epsilon + 4\delta)\} \\ & +\ldots \\ & +max\{0, P^1 + X_k - (2\epsilon + 4\delta), P^2 + (P - X_{k-1}) - (2\epsilon + 4\delta)\} \end{aligned}$$

   From the first part we know that since X_1, X_2, ..., X_k are optimal allocations, they must satisfy the following equalities. If the equalities can not be satisfied with the given parameters than the difference between both sides must be minimum.
   $P^1 + X_2 = P^2 + P - X_1$ and also
   $P^1 + X_3 = P^2 + P - X_2 \Rightarrow P^1 + X_2 = P^2 + P - X_3$

If w.l.o.g. the difference between both sides of the equality is smaller when we use X_1 than the difference when we use X_3, then using another allocation type increases the cycle time. Thus we conclude that $X_3 = X_1$ and this is actually not a k-allocation type $S2$ but a two-allocation type $S2$. □

Now let us prove that the optimal number of allocation types to be used with $S_{12}S_{21}$ is two:

Theorem 4.2 *For 2-unit robot move cycle, $S_{12}S_{21}$, using two allocation types is better than using any number of allocation types.*

Proof. Let us consider the cycle time of i^{th}, (since there are at most $\frac{k}{2}$ repetitions for a k-allocation type $S_{12}S_{21}$ $i < \frac{k}{2}$) repetition of a k-allocation type $S_{12}S_{21}$, where k is even:

$$\begin{aligned} C_t = \; & 12\epsilon + 14\delta + P^1 + P^2 + Q_{(k)(2i)} + (P - Q_{(k)(2i-1)}) \\ & + max\{0, P^1 + Q_{(k)(2i-1)} - (2\epsilon + 4\delta), P^2 + (P - Q_{(k)(2i)}) - (2\epsilon + 4\delta)\} \end{aligned}$$

and the optimal solution can be found by letting, $Q_{(k)(2i)} = P - Q_{(k)(2i-1)} = 0$ which implies that

$$\Rightarrow C_t = 12\epsilon + 14\delta + P^1 + P^2 + max\{0, (P^1 + P) - (2\epsilon + 4\delta), (P^2 + P) - (2\epsilon + 4\delta)\}$$

Since we assumed that $P^1 > P^2$ we have the following:

$$C_t = 12\epsilon + 14\delta + P^1 + P^2 + max\{0, (P^1 + P) - (2\epsilon + 4\delta)\}$$

Thus we conclude that if k is even using two-allocations is optimal.

Now let us consider using an odd number of allocation types, say k where k is odd. The following allocation types will be used in each repetition of $S_{12}S_{21}$:

$$\overbrace{1 \text{ and } 2}^{1}, \overbrace{3 \text{ and } 4}^{2}, \ldots, \overbrace{k \text{ and } 1}^{(k+1)/2}, \overbrace{2 \text{ and } 3}^{(k+3)/2}, \ldots, \overbrace{(k-1) \text{ and } (k)}^{k}$$

The cycle time for this allocation type is as follows:

$$\begin{aligned}
C_t = \; & 12\epsilon + 14\delta + P^1 + P^2 + Q_{(k)(1)} + (P - Q_{(k)(2)}) \\
& + max\{0, P^1 + Q_{(k)(2)} - (2\epsilon + 4\delta), P^2 + (P - Q_{(k)(1)}) - (2\epsilon + 4\delta)\} \\
& 12\epsilon + 14\delta + P^1 + P^2 + Q_{(k)(3)} + (P - Q_{(k)(4)}) \\
& + max\{0, P^1 + Q_{(k)(4)} - (2\epsilon + 4\delta), P^2 + (P - Q_{(k)(3)}) - (2\epsilon + 4\delta)\} \\
& \vdots \\
& 12\epsilon + 14\delta + P^1 + P^2 + Q_{(k)(k)} + (P - Q_{(k)(1)}) \\
& + max\{0, P^1 + Q_{(k)(1)} - (2\epsilon + 4\delta), P^2 + (P - Q_{(k)(k)}) - (2\epsilon + 4\delta)\} \\
& \vdots \\
& 12\epsilon + 14\delta + P^1 + P^2 + Q_{(k)(k-1)} + (P - Q_{(k)(k)}) \\
& + max\{0, P^1 + Q_{(k)(k)} - (2\epsilon + 4\delta), P^2 + (P - Q_{(k)(k-1)}) - (2\epsilon + 4\delta)\}
\end{aligned}$$

When we arrange the cycle time we get the following:

$$\begin{aligned}
C_t = \; & 12(k)\epsilon + 14(k)\delta + (k)P + (k)P^1 + (k)P^2 \\
& + max\{0, P^1 + Q_{(k)(2)} - (2\epsilon + 4\delta), P^2 + (P - Q_{(k)(1)}) - (2\epsilon + 4\delta)\} \\
& + max\{0, P^1 + Q_{(k)(4)} - (2\epsilon + 4\delta), P^2 + (P - Q_{(k)(3)}) - (2\epsilon + 4\delta)\} \\
& \vdots \\
& + max\{0, P^1 + Q_{(k)(1)} - (2\epsilon + 4\delta), P^2 + (P - Q_{(k)(k)}) - (2\epsilon + 4\delta)\} \\
& \vdots \\
& + max\{0, P^1 + Q_{(k)(k)} - (2\epsilon + 4\delta), P^2 + (P - Q_{(k)(k-1)}) - (2\epsilon + 4\delta)\}
\end{aligned}$$

This is the cycle time to repeat $S_{12}S_{21}$ k times and in order to compare this time with the cycle time of two-allocation $S_{12}S_{21}$, we have to divide this by k.

For two-allocation $S_{12}S_{21}$, we have the following cycle time:

$$\Rightarrow C_t = 12\epsilon + 14\delta + P^1 + P^2 + max\{0, (P^1 + P) - (2\epsilon + 4\delta)\}$$

So we have to analyse the following cases:

1- If $P + P^1 < 2\epsilon + 4\delta$, then for two-allocation $S_{12}S_{21}$ we have:

$$C_t = 12\epsilon + 14\delta + P^1 + P^2$$

and for k-allocation $S_{12}S_{21}$, where k is odd, we have:

$$C_t = 12\epsilon + 14\delta + P + P^1 + P^2 + X$$

where X represents the max terms and we know that $X \geq 0$. So for this case using two-allocations is always better than using k-allocations where k is an odd number.

2- If $P + P^1 < 2\epsilon + 4\delta$, then for two-allocation $S_{12}S_{21}$ we have:

$$C_t = 12\epsilon + 14\delta + P^1 + P^2 + P + P^1 - (2\epsilon + 4\delta)$$

and for k-allocation $S_{12}S_{21}$ we have:

$$C_t = 12\epsilon + 14\delta + P + P^1 + P^2 + X$$

where X is identical with the previous case. Then using two-allocations is better than using k-allocations for k is an odd number if the following condition holds:

$$12\epsilon + 14\delta + P^1 + P^2 + P + P^1 - (2\epsilon + 4\delta) \leq 12\epsilon + 14\delta + P + P^1 + P^2 + X$$

After some elimination we get the following:

$$P^1 - (2\epsilon + 4\delta) \leq X$$

In this condition we know that $X \geq 0$. Then we have the following two cases:

i- If $P^1 \leq (2\epsilon + 4\delta)$, then the condition is satisfied.

ii- If $P^1 \geq (2\epsilon + 4\delta)$, then when we observe the max terms we see that, $X \geq P^1 - (2\epsilon + 4\delta)$ and the condition is satisfied for this case also.

The analysis of these cases proved that the cycle time of using two-allocation $S_{12}S_{21}$ is always better than using k-allocation $S_{12}S_{21}$ where k is odd. □

Assuming w.l.o.g. that $P^1 > P^2$ we can give the cycle times of $S1$, $S2$ and $S_{12}S_{21}$ respectively:

i- $C_t^{S1} = 6\epsilon + 6\delta + P + P^1 + P^2$

ii- If $P^1 > \frac{P+P^1+P^2}{2}$ then $C_t^{S2} = 6\epsilon + 8\delta + max\{0, P^1 - (2\epsilon + 4\delta)\}$
otherwise if $P^1 < \frac{P+P^1+P^2}{2}$ then
$C_t^{S2} = 12\epsilon + 16\delta + max\{0, (P^1 + P_{21}) - (2\epsilon + 4\delta), (P^2 + P - P_{22}) - (2\epsilon + 4\delta)\}$
$+ max\{0, (P^1 + P_{22}) - (2\epsilon + 4\delta), (P^2 + P - P_{21}) - (2\epsilon + 4\delta)\}$

iii- $C_t^{S_{12}S_{21}} = 12\epsilon + 14\delta + P^1 + P^2 + max\{0, (P^1 + P) - (2\epsilon + 4\delta), (P^2 + P) - (2\epsilon + 4\delta)\}$

Now we can give the main result of this chapter in the following theorem:

Theorem 4.3 *Assuming w.l.o.g. that $P^1 > P^2$, the following statements hold:*

1- If $2P + P^1 + P^2 < 2\delta$, then $S1$ gives the minimum cycle time,

2- Otherwise if $P^1 + P^2 < 2\delta$ and $2P^1 + P^2 + P < 2\epsilon + 6\delta$, then $S_{12}S_{21}$ gives the minimum cycle time,

3- Otherwise if $P^1 + P^2 < 2\delta$ and $2P^1 + P^2 + P > 2\epsilon + 6\delta$, then depending on the allocation of the tasks for $S2$ either $S2$ or $S_{12}S_{21}$ gives the minimum cycle time,

4- Otherwise $S2$ gives the minimum cycle time.

Proof. In Theorem 4.1 we proved that if both P^1 and $P^2 < \frac{P^1+P^2+P}{2}$ then for $S2$ using 2 allocation types was optimal and if either P^1 or $P^2 > \frac{P^1+P^2+P}{2}$ then using 1 allocation types was optimal. Thus, in order to prove this theorem let us consider these two regions:

1. If either P^1 or $P^2 > \frac{P^1+P^2+P}{2}$: assume w.l.o.g. that $P^1 > \frac{P^1+P^2+P}{2} \Rightarrow P^1 > P + P^2$ then in this region the cycle times are as follows:

i- $C_t^{S1} = 6\epsilon + 6\delta + P + P^1 + P^2$

ii- If $P^1 > 2\epsilon + 4\delta$ then $C_t^{S2} = 4\epsilon + 4\delta + P^1$
otherwise if $P^1 < 2\epsilon + 4\delta$ then $C_t^{S2} = 6\epsilon + 8\delta$

iii- If $P^1 + P > 2\epsilon + 4\delta$ then $C_t^{S_{12}S_{21}} = 5\epsilon + 5\delta + P^1 + \frac{P^2+P}{2}$
otherwise if $P^1 + P < 2\epsilon + 4\delta$ then $C_t^{S_{12}S_{21}} = 6\epsilon + 7\delta + \frac{P^1+P^2}{2}$

Now let us consider the following cases:

1. If $P^1 > 2\epsilon + 4\delta$ then this implies $P^1 + P > 2\epsilon + 4\delta$
   The cycle times are:
   $C_t^{S1} = 6\epsilon + 6\delta + P + P^1 + P^2$
   $C_t^{S2} = 4\epsilon + 4\delta + P^1$ and
   $C_t^{S_{12}S_{21}} = 5\epsilon + 5\delta + P^1 + \frac{P^2+P}{2}$
   Looking at the cycle times we conclude that for this case $S2$ gives the minimum cycle time.

2. If $P^1 < 2\epsilon + 4\delta$ and $P^1 + P^2 > 2\epsilon + 4\delta$
   The cycle times are:
   $C_t^{S1} = 6\epsilon + 6\delta + P + P^1 + P^2$
   $C_t^{S2} = 6\epsilon + 8\delta$ and
   $C_t^{S_{12}S_{21}} = 5\epsilon + 5\delta + P^1 + \frac{P^2+P}{2}$
   Since in this region $P^1 + P^2 > 2\epsilon + 4\delta$, $C_t^{S1} > 8\epsilon + 10\delta + P$, then $S2$ is always better than $S1$.
   When we compare $S2$ with $S_{12}S_{21}$, we see that if $2P^1 + P^2 + P < 2\epsilon + 6\delta$ then $S_{12}S_{21}$ is better than $S2$, otherwise $S2$ is better than $S_{12}S_{21}$.
   However, we know that in this region $P^1 + P > 2\epsilon + 4\delta$, then $2P^1 + P^2 + P > P^1 + P^2 + 2\epsilon + 4\delta$. From here, we conclude that if $S_{12}S_{21}$ is better than $S2$ then $2\epsilon + 6\delta > P^1 + P^2 + 2\epsilon + 4\delta \Rightarrow P^1 + P^2 < 2\delta \Rightarrow P^1 < 2\delta$.
   We also know that in this region, $P^1 > P^2 + P \Rightarrow P^1 > P \Rightarrow P < 2\delta$ then, $P^1 + P < 4\delta$, which contradicts $P^1 + P > 2\epsilon + 4\delta$. Thus we conclude that in this region $S2$ is always better than both $S1$ and $S_{12}S_{21}$.

3. If $P^1 + P < 2\epsilon + 4\delta$. This implies that $P^1 < 2\epsilon + 4\delta$
   The cycle times are:
   $C_t^{S1} = 6\epsilon + 6\delta + P + P^1 + P^2$
   $C_t^{S2} = 6\epsilon + 8\delta$ and
   $C_t^{S_{12}S_{21}} = 6\epsilon + 7\delta + \frac{P^1+P^2}{2}$

When we compare $S1$ with $S2$, we observe that if $P + P^1 + P^2 < 2\delta$ then $S1$ is better than $S2$, otherwise $S2$ is better than $S1$. One more observation is that, if $P + P^1 + P^2 < 2\delta$, then this implies that $P^1 + P < 2\epsilon + 4\delta$ and $P^1 < 2\epsilon + 4\delta$. So these observations help us to generalize the following statement:
If $P + P^1 + P^2 < 2\delta$ then, not only for this case but in general, $S1$ is better than $S2$, otherwise $S2$ is better than $S1$.
Now let us compare $S1$ with $S_{12}S_{21}$. For this case we observe that, if $2P + P^1 + P^2 < 2\delta$, $S1$ is better than $S_{12}S_{21}$, otherwise $S_{12}S_{21}$ is better than $S1$. In the same manner if we can generalize this statement, if $2P + P^1 + P^2 < 2\delta$ then $P^1 + P < 2\epsilon + 4\delta$. Thus we conclude that not only for this case but in general if $2P + P^1 + P^2 < 2\delta$ than $S1$ is better than $S_{12}S_{21}$, otherwise $S_{12}S_{21}$ is better than $S1$.
Lastly, we will compare $S2$ with $S_{12}S_{21}$. We can easily observe that if $P^1 + P^2 < 2\delta$ than $S_{12}S_{21}$ is better than $S2$ otherwise $S2$ is better than $S_{12}S_{21}$. We can also generalize this statement. If $P^1 + P^2 < 2\delta$ then $P^1 < 2\delta$. Since we consider the region for which $P^1 > P^2 + P$, then $P^1 > P$ and thus $P < 2\delta$. So we conclude that $P + P^1 < 4\delta < 2\epsilon + 4\delta$.

So the above analysis of the cases leads us to the conclusion that if $P^1 > \frac{P^1+P^2+P}{2}$, then the following holds:

i- If $2P + P^1 + P^2 < 2\delta$, then $S1$ gives the minimum cycle time,

ii- Otherwise if $P^1 + P^2 < 2\delta$, then $S_{12}S_{21}$ gives the minimum cycle time,

iii- Otherwise $S2$ gives the minimum cycle time.

2. If both P^1 and $P^2 < \frac{P^1+P^2+P}{2}$, in another representation $P^1 < P + P^2$ and $P^2 < P + P^1$, and assume w.l.o.g. that $P^1 > P^2$ similar to the above analysis:
For this case let us first compare the cycle times of $S1$ and $S_{12}S_{21}$:
$S1$ is better than $S_{12}S_{21}$ if

$$6\epsilon + 6\delta + P + P^1 + P^2 < 6\epsilon + 7\delta + \frac{P^1 + P^2 + max\{0, P^1 + P - (2\epsilon + 4\delta)\}}{2}$$

We have two subcases for this case:

i- If $P + P^1 < 2\epsilon + 4\delta$, then we conclude that if $2P + P^1 + P^2 < 2\delta$, then $S1$ is better than $S_{12}S_{21}$, otherwise $S_{12}S_{21}$ is better than $S1$. We also observe that if $2P + P^1 + P^2 < 2\delta$ then $P + P^1 < 2\epsilon + 4\delta$.

ii- Otherwise if $P + P^1 > 2\epsilon + 4\delta$, we conclude that if $P + P^2 < -(2\epsilon + 2\delta)$ then $S1$ is better than $S_{12}S_{21}$. However, since $P+P^2 > 0$ and $2\epsilon+2\delta > 0$, this case is infeasible. So for this case $S_{12}S_{21}$ is always better than $S1$.

In summary: If $2P + P^1 + P^2 < 2\delta$, then $S1$ is better than $S_{12}S_{21}$ otherwise $S_{12}S_{21}$ is better than $S1$.

In order to compare $S2$ with $S1$ and $S_{12}S_{21}$, let us find a lower and an upper bound for $S2$. Thus for this purpose, partitioning P such that $P + P_{2j} = \frac{P+P^1+P^2}{2}$ $j \in [1, 2]$ gives a lower bound. In order to find an upper bound we will consider the case that P cannot be partitioned and must be allocated to a single machine. The cycle times of the lower and upper bounds of $S2$ are as follows respectively:
Lower bound:

$$6\epsilon + 8\delta + max\{0, \frac{P + P^1 + P^2}{2} - (2\epsilon + 4\delta)\}$$

Upper bound:

$$6\epsilon + 8\delta + \frac{max\{0, P + P^1 - (2\epsilon + 4\delta), P + P^2 - (2\epsilon + 4\delta)\}}{2} + \frac{max\{0, P^1 - (2\epsilon + 4\delta), P^2 - (2\epsilon + 4\delta)\}}{2}$$

Since we assumed that $P^1 > P^2$ the upper bound becomes:

$$6\epsilon + 8\delta + \frac{max\{0, P + P^1 - (2\epsilon + 4\delta)\} + max\{0, P^1 - (2\epsilon + 4\delta)\}}{2}$$

Thus we have a total of four comparisons: Comparisons of both the lower and upper bounds of $S2$ with $S1$ and $S_{12}S_{21}$ separately:

1. Comparison of $S1$ with the lower bound of $S2$:

   $S1$ is better than the lower bound of $S2$ if:

$$6\epsilon + 6\delta + P + P^1 + P^2 < 6\epsilon + 8\delta + max\{0, \frac{P + P^1 + P^2}{2} - (2\epsilon + 4\delta)\}$$

Thus we have two cases:

i- If $P + P^1 + P^2 < 4\epsilon + 8\delta$, then $max\{0, \frac{P+P^1+P^2}{2} - (2\epsilon + 4\delta)\} = 0$, and $S1$ is better than the lower bound of $S2$ if $P + P^1 + P^2 < 2\delta$

ii- Otherwise if $P + P^1 + P^2 > 4\epsilon + 8\delta$, then

$$max\{0, \frac{P + P^1 + P^2}{2} - (2\epsilon + 4\delta)\} = \frac{P + P^1 + P^2}{2} - (2\epsilon + 4\delta)$$

So $S1$ is better than the lower bound of $S2$ if:

$$6\epsilon + 6\delta + P + P^1 + P^2 < 6\epsilon + 8\delta + \frac{P + P^1 + P^2}{2} - (2\epsilon + 4\delta)$$

which is same as:

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

Since $\frac{P+P^1+P^2}{2}$ and $2\epsilon + 2\delta$ are greater than 0, this case is not feasible.
Since $P + P^1 + P^2 < 2\delta$ also implies that $P + P^1 + P^2 < 4\epsilon + 8\delta$ we do not require this condition any more and conclude that if $P + P^1 + P^2 < 2\delta$ then $S1$ is better than the lower bound of $S2$, otherwise the lower bound of $S2$ is better than $S1$.

2. Comparison of $S1$ with the upper bound of $S2$:

$S1$ is better than $S2$ whenever,

$$6\epsilon+6\delta+P+P^1+P^2 < 6\epsilon+8\delta+\frac{max\{0, P+P^1-(2\epsilon+4\delta)\}+max\{0, P^1-(2\epsilon+4\delta)\}}{2}$$

$$\Rightarrow P+P^1+P^2 < 2\delta+\frac{max\{0, P+P^1-(2\epsilon+4\delta)\}+max\{0, P^1-(2\epsilon+4\delta)\}}{2}$$

Thus we have the following cases:

i- If $P^1 < 2\epsilon+4\delta$ then this implies $P+P^1 < 2\epsilon+4\delta$, then $S1$ is better than the upper bound of $S2$, if $P+P^1+P^2 < 2\delta$. Since $P+P^1+P^2 < 2\delta$ implies that $P^1 < 2\epsilon+4\delta$, we do not require this condition any more.

ii- If $P^1 < 2\epsilon+4\delta$ but $P+P^1 > 2\epsilon+4\delta$, then $S1$ is better than the upper bound of $S2$ if $P+P^1+P^2 < 2\delta+\frac{P+P^1-(\epsilon+2\delta)}{2}$
$\Rightarrow P+P^1+2P^2 < -2\epsilon$. Since all variables are greater than or equal to 0, then this condition is not attainable, which means $S2$ is always better than $S1$ for this case.

iii- If $P^1 > 2\epsilon+4\delta$, this also implies $P+P^1 > 2\epsilon+4\delta$, then $S1$ is better than $S2$ for $P+P^1+2P^2 < 2\delta+P^1+\frac{P}{2}-(2\epsilon+4\delta)$
$\Rightarrow \frac{P}{2}+P^2 < -(2\epsilon+2\delta)$, which is again not attainable. Thus for this case also $S2$ is always better than $S1$.

We conclude after comparing $S1$ with the upper bound of $S2$ that $S1$ is better than $S2$ if $P+P^1+P^2 < 2\delta$, otherwise $S2$ is better than $S1$. As we remember from the comparison of $S1$ with the lower bound of $S2$ that, this result is the same with the result of that comparison. So the final conclusion is that $S1$ is better than $S2$ if $P+P^1+P^2 < 2\delta$, otherwise $S2$ is better than $S1$.

3. Comparison of $S_{12}S_{21}$ with the lower bound of $S2$:

$S_{12}S_{21}$ is better than the lower bound of $S2$ if:

$$6\epsilon + 7\delta + \frac{P^1 + P^2 + max\{0, P + P^1 - (2\epsilon + 4\delta)\}}{2} <$$
$$6\epsilon + 8\delta + max\{0, \frac{P + P^1 + P^2}{2} - (2\epsilon + 4\delta)\}$$

Thus we should consider the following three cases:

i- If $P + P^1 < 2\epsilon + 4\delta$, this implies that $\frac{P+P^1+P^2}{2} < 2\epsilon + 4\delta$. This is because of the following reasoning:
Since we are considering the case that $P^2 < P + P^1$, then $\frac{P+P^1+P^2}{2} < \frac{2(P+P^1)}{2} = P + P^1 < 2\epsilon + 4\delta$.
For this case, $S_{12}S_{21}$ is better than the lower bound of $S2$ if

$$6\epsilon + 7\delta + \frac{P^1 + P^2}{2} < 6\epsilon + 8\delta$$

from here we get, if $P^1 + P^2 < 2\delta$ then $S_{12}S_{21}$ is better than $S2$, otherwise $S2$ is better than $S_{12}S_{21}$.

ii- If $\frac{P+P^1+P^2}{2} > 2\epsilon + 4\delta$, this implies that $P + P^1 > 2\epsilon + 4\delta$, which can be stated as follows:
In the previous case we proved that $\frac{P+P^1+P^2}{2} < P + P^1$ and if $\frac{P+P^1+P^2}{2} > 2\epsilon + 4\delta$ that means $P + P^1 > 2\epsilon + 4\delta$.
For this case, $S_{12}S_{21}$ is better than the lower bound of $S2$ if

$$6\epsilon + 7\delta + \frac{P + 2P^1 + P^2 - (2\epsilon + 4\delta)}{2} < 6\epsilon + 8\delta + \frac{P + P^1 + P^2}{2} - (2\epsilon + 4\delta)$$

then $S_{12}S_{21}$ is better than the lower bound of $S2$ if $P^1 < -(2\epsilon + 2\delta)$, and since all of the parameters must be greater than or equal to 0, this is infeasible. So for this case, $S2$ is always better than $S_{12}S_{21}$.

iii- If $\frac{P+P^1+P^2}{2} < 2\epsilon + 4\delta$ but $P + P^1 > 2\epsilon + 4\delta$ then for this case $S_{12}S_{21}$ is better than the lower bound of $S2$ if

$$6\epsilon + 7\delta + \frac{P + 2P^1 + P^2 - (2\epsilon + 4\delta)}{2} < 6\epsilon + 8\delta$$

then $S_{12}S_{21}$ is better than the lower bound of $S2$ if $P+2P^1+P^2 < 2\epsilon+6\delta$, and this condition implies that $\frac{P+P^1+P^2}{2} < 2\epsilon + 4\delta$.

We can further reduce the number of above cases and find a conclusion as follows:

If $P + P^1 > 2\epsilon + 4\delta$ and $P + 2P^1 + P^2 < 2\epsilon + 6\delta$, these two imply that $P^1 + P^2 < 2\delta$. In case (i) we considered the region $P + P^1 < 2\epsilon + 4\delta$ and found out that if $P^1 + P^2 < 2\delta$ then $S_{12}S_{21}$ is better than the lower bound of $S2$ and in case (iii) we considered the region $P + P^1 > 2\epsilon + 4\delta$ and found out that $S_{12}S_{21}$ is better than the lower bound of $S2$ if $P + 2P^1 + P^2 < 2\epsilon + 6\delta$. So we can combine these two conditions as follows:

If $P^1 + P^2 < 2\delta$ and $P + 2P^1 + P^2 < 2\epsilon + 6\delta$ then $S_{12}S_{21}$ is better than the lower bound of $S2$, otherwise the lower bound of $S2$ is better than $S_{12}S_{21}$.

4. Comparison $S_{12}S_{21}$ with the upper bound of $S2$:

$S_{12}S_{21}$ is better than the upper bound of $S2$ if

$$6\epsilon + 7\delta + \frac{P^1 + P^2 + max\{0, P + P^1 - (2\epsilon + 4\delta)\}}{2} <$$
$$6\epsilon + 8\delta + \frac{max\{0, P + P^1 - (2\epsilon + 4\delta)\} + max\{0, P^1 - (2\epsilon + 4\delta)\}}{2}$$

From here we get $S_{12}S_{21}$ is better than the upper bound of $S2$ if

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

Thus we have the following cases:

i- If $P^1 < 2\epsilon + 4\delta$, then if $P^1 + P^2 < 2\delta$ then $S_{12}S_{21}$ is better than $S2$, otherwise $S2$ is better than $S_{12}S_{21}$. If $P^1 + P^2 < 2\delta$ this implies that $P^1 < 2\epsilon + 4\delta$. So we do not require this condition any more.

ii- If $P^1 > 2\epsilon + 4\delta$, then if $P^1 + P^2 < 2\delta + P^1 - (2\epsilon + 4\delta) \Rightarrow P^2 < -(2\epsilon + 4\delta)$ then $S_{12}S_{21}$ is better than $S2$. Since all of the parameters are greater than or equal to 0, then this is impossible. So for this case the upper bound of $S2$ is always better than $S_{12}S_{21}$.

Now we can combine the cases of the comparisons of $S_{12}S_{21}$ with the lower and the upper bounds of $S2$ as follows:

1- If $P^1 + P^2 < 2\delta$ and $P + 2P^1 + P^2 < 2\epsilon + 6\delta$ then $S_{12}S_{21}$ is better than $S2$

2- Otherwise if $P^1 + P^2 < 2\delta$ and $P + 2P^1 + P^2 > 2\epsilon + 6\delta$ then according to the allocation of $S2$ either $S2$ or $S_{12}S_{21}$ gives the minimum cycle time,

3- Otherwise if $P^1 + P^2 > 2\delta$ then $S2$ gives the minimum cycle time.

So the above analysis of the cases leads us to the conclusion that if $P^1 < \frac{P^1+P^2+P}{2}$, then the following holds:

1- If $2P + P^1 + P^2 < 2\delta$, then $S1$ gives the minimum cycle time,

2- Otherwise if $P^1 + P^2 < 2\delta$ and $P + 2P^1 + P^2 < 2\epsilon + 6\delta$ then $S_{12}S_{21}$ gives the minimum cycle time,

3- Otherwise if $P^1 + P^2 < 2\delta$ and $P + 2P^1 + P^2 > 2\epsilon + 6\delta$ then according to the allocation of $S2$ either $S2$ or $S_{12}S_{21}$ gives the minimum cycle time,

4- Otherwise if $P^1 + P^2 > 2\delta$ then $S2$ gives the minimum cycle time.

When we consider the two cases where $P^1 < P + P^2$ and $P^1 > P + P^2$, we observe that these two cases have very similar results, the only difference is for $P^1 < P + P^2$, we have extra conditions for $P^1 + P^2 = 2\delta$ and $P + 2P^1 + P^2 =$

$2\epsilon + 6\delta$ and we can easily verify that the intersection of $P^1 + P^2 = 2\delta$ and $P + 2P^1 + P^2 = 2\epsilon + 6\delta$ is for $P^1 < P$ which implies that $P^1 < P + P^2$. Thus we can combine the cases of $P^1 < P + P^2$ and $P^1 > P + P^2$, which completes the proof, as follows:

1- If $2P + P^1 + P^2 < 2\delta$, then $S1$ gives the minimum cycle time,

2- Otherwise if $P^1 + P^2 < 2\delta$ and $P + 2P^1 + P^2 < 2\epsilon + 6\delta$, then $S_{12}S_{21}$ gives the minimum cycle time,

3- Otherwise if $P^1 + P^2 < 2\delta$ and $P + 2P^1 + P^2 > 2\epsilon + 6\delta$, then depending on the allocation of the tasks for $S2$ either $S2$ or $S_{12}S_{21}$ gives the minimum cycle time,

4- Otherwise $S2$ gives the minimum cycle time.

□

Remember that for this theorem and the proof we assumed that $P^1 > P^2$. For the reverse case of this assumption, the results can be found easily in analogy with the above analysis.

In sake of well understanding an example will be helpful:

Example 4.2 Assume that we have $P^1 > P^2$ and $\epsilon = \delta = 10$. If we represent P^2 in the conditions above as αP^1, where $0 \leq \alpha \leq 1$, we have the following conditions:
If $2P + P^1 + \alpha P^1 < 2\delta$, then $S1$ gives the minimum cycle time,
Otherwise if $P^1 + \alpha P^1 < 2\delta$ and $2P^1 + \alpha P^1 + P < 2\epsilon + 6\delta$, then $S_{12}S_{21}$ gives the minimum cycle time,
Otherwise if $P^1 + \alpha P^1 < 2\delta$ and $2P^1 + \alpha P^1 + P > 2\epsilon + 6\delta$, then depending on the allocation of the tasks for $S2$ either $S2$ or $S_{12}S_{21}$ gives the minimum cycle time,
Otherwise $S2$ gives the minimum cycle time.
Thus by changing the values of α from 0 to 1 we can represent P^2 in terms of P^1

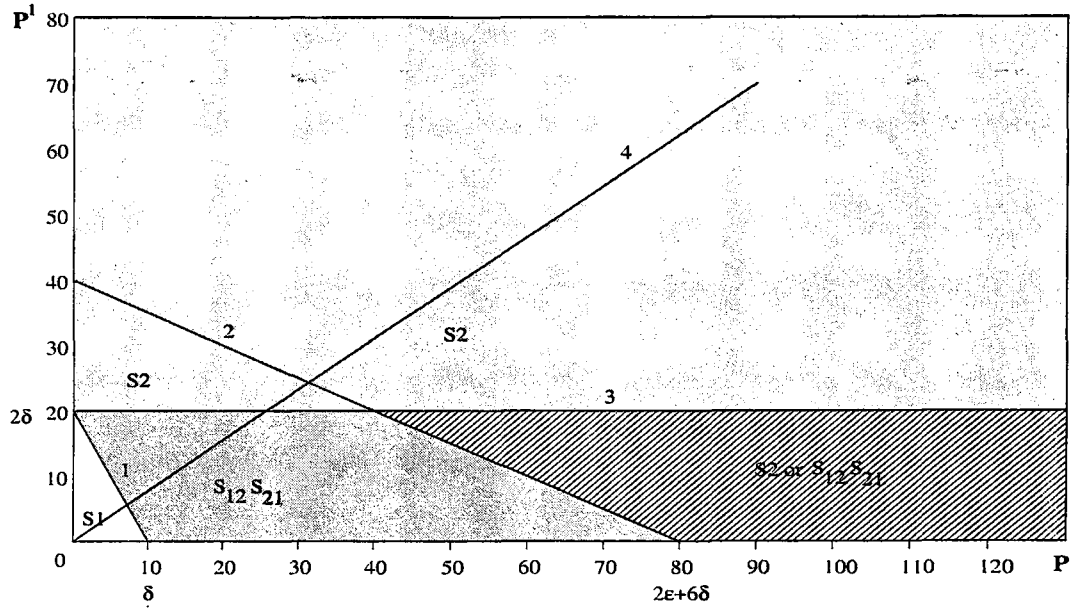

Figure 4.1: Optimal Regions for $S1$, $S2$ and $S_{12}S_{21}$

and this makes it easy to analyse the cases. Furthermore, by this representation we can analyse the results graphically. Let us assume that $\alpha = 0$, which means that $P^2 = 0$. So the above cases become:
If $2P + P^1 < 2\delta$, then $S1$ gives the minimum cycle time,
Otherwise if $P^1 < 2\delta$ and $2P^1 + P < 2\epsilon + 6\delta$, then $S_{12}S_{21}$ gives the minimum cycle time,
Otherwise if $P^1 < 2\delta$ and $2P^1 + P > 2\epsilon + 6\delta$, then depending on the allocation of the tasks for $S2$ either $S2$ or $S_{12}S_{21}$ gives the minimum cycle time,
Otherwise $S2$ gives the minimum cycle time.
We can show the regions for these cases as a graph of P vs. P^1 as in Figure 4.1.

Now let us analyse what happens to the above results when we perform a sensitivity analysis with respect to the values of α, ϵ and δ.

α: As we increase the value of α from 0 to 1, the slope of line 4 in the Figure 4.1 increases. This line is the result of the following inequality:
$P^1 < P + \alpha P^2$ and when $\alpha = 1$, then $P = 0$ and the slope of the line goes to

infinity. With this change of α, the intersection points of the lines with the P axes do not change. On the other side, when we increase α the constant term of P^1 in the inequalities increases. This increase results in a decrease in the intersection points of the lines with the P^1 axes. For example, line 1 intersects with P^1 axes at point $P^1 = 2\delta$ for $\alpha = 0$, whereas this point reduces to $P^1 = \delta$ for $\alpha = 1$, and for line 2, the intersection point reduces from $P^1 = \epsilon + 3\delta$ to $\frac{\epsilon+3\delta}{2}$.

Now let us consider the changes in the regions with the increase of α. The region for which $S1$ gives the minimum cycle time shrinks as well as the region for which $S_{12}S_{21}$ gives the minimum cycle time. Furthermore, the region in which the minimum depends on the allocation of tasks for $S2$ also shrinks. However, the region for which $P^1 > 2\delta$, that is, the region for which $S2$ gives the minimum cycle time enlarges. Thus, we can conclude that, increasing α from 0 to 1 is in favor of robot move cycle $S2$. In other words, increasing P^2 increases the number of cases for which $S2$ gives the minimum cycle time.

δ: If we reduce the value of δ to zero, that is, if we assume that the transportation time of the parts between the machines by the robot is 0, then line 3 comes down on to the P axes which brings the conclusion that for this problem definition $S2$ always gives the minimum cycle time. This remark leads us to the conclusion that, $S2$ is better than the other two cycles in terms of processing times but it is worse than them in terms of transportation times. That is, in cycle $S2$ the robot travels more distance than the other two cycles and if the transportation times are zero, this cycle always gives the minimum cycle time.

If we consider the opposite direction, that is, if we increase the transportation times, the region for which $S2$ gives the minimum cycle time shrinks and all other regions enlarge.

ϵ: When we observe the above inequalities, we see that the only equation in which ϵ is present is the following:

$2P^1 + P < 2\epsilon + 6\delta$, which determines line 2 in the graph. When we reduce the value of ϵ to 0 that is, when we assume that the loading and unloading times are zero, we see that the intersection points of this line with the P and P^1 axes decrease. This results in a shrinkage in the region for which $S_{12}S_{21}$ gives the minimum cycle time and an enlargement in the region for which the minimum cycle time is given by $S_{12}S_{21}$ or $S2$ depending on the allocation of tasks of $S2$. This means that when we have higher loading and unloading times, the number of cases for which $S_{12}S_{21}$ gives the minimum cycle time is also higher. Another conclusion is that, in cycle $S2$ we have more loading and unloading operations compared to the robot move cycle $S_{12}S_{21}$.

4.3 Conclusion

In this chapter, we assumed that some set of tasks can only be processed in the first machine and some set of tasks can only be processed in the second machine. The remaining tasks can be processed on both of the machines and the allocation of these tasks to the machines and the optimal robot move cycle corresponding to this allocation of tasks that jointly minimize the cycle time is the problem that is considered in this chapter.

For this variation of the problem we proved that the Lemmas 3.1, 3.2 and 3.3 of the previous chapter are still valid, which reduced the number of robot move cycles to be considered to three. We proved in Theorem 4.1 that for the robot move cycle $S2$, using two-allocation types is not optimal in some regions and we proved that in these regions using one-allocation type $S2$ is optimal. In theorem 4.2, we proved that for robot move cycle $S_{12}S_{21}$ using two-allocation types is still optimal. The solution of the considered problem is finally presented in Theorem 4.3 where we presented regions of optimality for each robot move cycle.

Chapter 5

Conclusion

In this study we considered the two-machines, identical parts robotic cell scheduling problem with operational flexibility. The workstations in a robotic cell are predominantly CNC machines, and these machines pose an operational flexibility by definition. Operational flexibility can be defined as the ability to interchange the ordering of several operations for each part type (Browne et al. [3]). The CNC machines are equipped with tool magazines in which the required tools are stored. Whenever an operation requires one of these tools, the tool change can be done within very small setup times and the operational flexibility is a result of this.

In chapter 3 we considered the problem assuming that the tool magazines of each machine are loaded with the required tools in order to complete the processing of a part. That is, we assumed that each machine is capable of performing all of the tasks required to finish the processing of a part. Assuming that the allocation of the tasks is fixed, Sethi et al. [34] proved that 1-unit cycles give the minimum cycle time. However, Lemmas 3.1, 3.2 and 3.3 jointly state that the optimal is one of the 1-unit robot move cycles or a 2-unit cycle. In Theorems 3.1 and 3.2 we proved that the optimal number of allocation types to be used with $S2$ and $S_{12}S_{21}$ is two. In Theorem 3.3 we gave the optimality regions for these robot move cycles.

In this chapter we also proposed a new robot move cycle which is a direct result of the assumption of operational flexibility. This robot move cycle is not only simple and practical but as proven in Theorem 3.4, it also gives the optimal solution for the problem considered in this chapter. Last analysis of the chapter considers the change of the layout from in-line robotic cell to robot-centered cell and we proved in Theorem 3.5 that this change further improves the cycle time of the new cycle while the cycle times of all of the other cycles remain the same.

In some cases, assuming that each machine is capable of performing all of the tasks may not be a realistic assumption. The tool magazines of the CNC machines have a limited capacity and thus it may not be possible to place all of the required tools to the tool magazines of these machines. As a result of this not all of the tasks can be processed on one machine but some tasks can only be processed on the first or the second machine and the remaining tasks are to be allocated to these two machines. In Chapter 4, we considered this problem and solved the problem for this new definition. We proved that Lemmas 3.1, 3.2 and 3.3 are still valid for this definition of the problem which means that the optimal solution to this problem also is one of the 1-unit robot move cycles or a 2-unit robot move cycle. The main result of the chapter is given in Theorem 4.3 which presents the optimality regions for these three robot move cycles.

Possible topics for future research include the extension of this study to three machine robotic cells. Proving the validity of the conjecture about 1-unit cycles being optimal in three machine case with operational flexibility is important. Of course that problem is much more complicated than this one since there are three 1-unit robot move cycles to be considered in a 3-machine robotic cell while for 2-machine cells we had two and the number of higher unit robot move cycles also increases drastically when we increase the number of machines from two to three.

Another topic is to consider the technological differences between the machines. For example one of the machines may be newer than the other one. Thus, this machine may have higher cutting speeds and feed rates than

the other one. The time to complete a specified task if it is allocated to the newer one may be t while if it is allocated to the older machine this time may be βt where $\beta \geq 1$. This time may also assumed to be $t + \beta$. It can also be considered that not one of the machines is better than the other but the tools loaded to these machines can be different in quality allowing the use of higher feed rates and cutting speeds. Thus one task may have a smaller processing time on the first machine while another task may have a smaller processing time on the second machine. For a further discussion on the determination of processing times with respect to the machining conditions, i.e. cutting speed and feed rate, we refer to Akturk [1]. As a result of that we have to deal with non-identical parallel machine scheduling problem, which is more complicated than the identical one. The difficulty with these problems is in finding the optimal number of allocation types for $S2$ and $S_{12}S_{21}$. Also, the validity of Lemmas 3.1, 3.2 and 3.3 is an open question.

Bibliography

[1] Akturk, M.S., (1999) An exact tool allocation approach for CNC machines. *International Journal of Computer Integrated Manufacturing* **12** *(2), 129–140.*

[2] Brauner, N., Finke, G., (1999a) On the conjecture in robotic cells: new simplified proof for the three-machine case. *INFOR* **37** *20–36.*

[3] Browne, J., Harhen, J., Shivnan, J., *Production Management Systems* , Addison-Wesley, New York, NY (1988).

[4] Crama, Y., Van de Klundert, J., (1997) Cyclic scheduling of identical parts in a robotic cell. *Operations Research* **45** *(6), 952–965.*

[5] Crama, Y., Van de Klundert, J., (1999) Cyclic scheduling in 3-machine robotic flow shops. *Journal of Scheduling* **4**, *35–54.*

[6] Crama, Y., Kats, V., Van de Klundert, J., Levner, E., (2000) Cyclic scheduling in robotic flowshops. *Annals of Operations Research* **96**, *97–124.*

[7] Finke, G., Gueguen, C., Brauner, N., Robotic cells with buffer space, in: *ECOCO IX Conference Proceedings, eds. R. O'Conner and P. Magee* (Dublin City University, 1996).

[8] Gilmore, P.C., Gomory, R.E., (1964) Sequencing in one state-variable machine: a solvable case of the travelling salesman problem. *Operations Research* **12** *655–679.*

[9] Gilmore, P.C., Lawler, E.L., Shmoys, D.B., *Well solved special cases in: The Travelling Salesman Problem: A Guided Tour of Combinatorial Optimization, Wiley-Interscience Series in Discrete Mathematics*, Wiley, Chichester, (1985.)

[10] Graham, R.L., Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G, (1979) Optimization adn approximation in deterministic sequencing and scheduling: a survey. *Annals Discrete Mathematics* **5** *287–326.*

[11] Hall, N.G., Kamoun, H., Sriskandarajah, C., (1997) Scheduling in robotic cells: classification, two and three machine cells. *Operations Research* **45** *(3) 421–439.*

[12] Hall, N.G., Kamoun, H., Sriskandarajah, C., (1998) Scheduling in robotic cells: complexity and steady state analysis. *European Journal of Operations Research* **109** *43–65.*

[13] Hall, N.G., Potts, C.N., Sriskandarajah, C., (2000) Parallel machine scheduling with a common server. *Discrete Applied Mathematics* **102,** *223–243.*

[14] Han, B.T., Cook, J.S., (1998) An efficient heuristic for robot acquisition and cell formation. *Annals of Operations Research* **77,** *229–252.*

[15] Hartley, J., *Robots at Work*, North-Holland Publishing Company, New York, (1983).

[16] Hitomi, K., Yoshimura, M., *Operations Scheduling for Work Transportation by Industrial Robots in Automated Manufacturing Systems, In Robotics and Material Flow*, Elsevier Science Publishers, New York, NY. pp. 131–139 (1986).

[17] Hurink, J., Knust, S., (2001) Makespan minimization for flow-shop problems with transportation times and a single robot. *Discrete Applied Mathematics* **112** *199–216.*

[18] Hurink, J., Knust, S., (2002) A tabu search algorithm for scheduling a single robot in a job-shop environment. *Discrete Applied Mathematics* **119**, *181–203.*

[19] Jeng, W., Lin, J., Wen, U., (1993) Algorithms for sequencing robot activities in a robot-centered parllel-processor workcell. *Computers and Operations Research* **20** *185–197.*

[20] Kamoun, H., Hall, N.G., Sriskandarajah, C., (1999) Scheduling in robotic cells: heuristics and cell design. *Operations Research* **47**, *(6) 821–835.*

[21] Karzanov, A.V., Livshits, E.M., (1978) Minimal quality of operators for serving a homogeneous linear technological process, Part 2. *Automation and Remote Control* **39** *(3) 445–450.*

[22] Kats, V., Levner, E., (1998) Cyclic scheduling of operations for a part type in an FMS handled by a single robot: a parametric critical path approach. *The International Journal of Flexible Manufacturing Systems* **10** *129–138.*

[23] Kats, V., Levner, E., (2002) Cyclic scheduling in a robotic production line. *Journal of Scheduling* **5**, *23–41.*

[24] King, R.E., Hodgson, T.J., Chafee, F.E., (1993) Robot task scheduling in a flexible manufacturing cell. *IIE Transactions* **25** *(2) 80–87.*

[25] Kise, H., (1991) On an automated two-machine flowshop secheduling problem with infinite buffer. *Journal of the Operations Research Society of Japan* **34** *(3) 354–361.*

[26] Kise, H., Shioyama, T., Ibaraki, T., (1993) Automated two machine flowshop scheduling: a solvable case. *IIE Transactions* **23** *80–87.*

[27] Kogan K., Levner, E., (1997) A polynomial algorithm for scheduling small-scale manufacturing cells served by multiple robots. *Computers and Operations Research* **25** *(1) 53–62.*

[28] Levner, E., Kogan, K., Levin, Y., (1995) Scheduling a two-machine robotic cell: a solvable case. *Annals of Operations Research* **57** *217–232.*

[29] Levner, E., Vlach, M., Single machine scheduling with fuzzy precedence constraints. *IS-RR-97-0031F* School of Information Science, Japan Institute of Science and Technology, Hokuriku, (1997).

[30] Lieberman, R.W., Turksen, I.B., (1981) Crane scheduling problems *AIIE Transactions* **13** *(4) 304–311.*

[31] Logendran, R., Sriskandarajah, C., (1996) Sequencing ofrobot activities and parts in two-machine robotic cells. *International Journal of Production Research* **34**, *3447–3463.*

[32] Miller, R.K., Walker, T.C., *FMS/CIM Systems Integration Handbook*, The Fairmont Press, Lilburn, GA, (1990).

[33] Papdimitriou, C.H., Kanellakis, P.C., (1980) Flowshop scheduling with limited temporary storage. *Journal of the ACM* **27** *(3) 533–549.*

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

[35] Song, W., Zabinksky, Z.B., Storch, R.L., (1993) An algorithm for scheduling a chemical processing tank line. *Production Planning and Control* **4** *(4) 323–332.*

[36] Sriskandarajah, C., Hall, N.G., Kamoun, H., (1998) Scheduling large robotic cells without buffers. *Annals of Operations Research* **76**, *287–321.*

[37] Stern, H.I., Vitner, G., (1990) Scheduling parts in a combined production-transportation work cell. *Journal of the Operational Reseach Society* **41** *625–632.*

[38] Tompkins, J.A., White, J.A., Bozer, Y.A., Frazelle, E.H., Tanchoco, J.M.A., Trevino, J., *Facilities Planning*, John Wiley and Sons, New York, (1996.)

VITA

Hakan Gültekin was born on May 5, 1978 in Nevşehir, Türkiye. He received his high school education at Bursa Ulubatlı Hasan Anadolu Lisesi, Türkiye. He has graduated from the Department of Industrial Engineering, Bilkent University in June 2000 and joined the Department of Industrial Engineering at Bilkent University as a research assistant in September 2000. From that time to present he worked with Assoc. Prof. Selim Aktürk for his graduate study at the same department. He is married with Feyza Gültekin and father of Ayşe Melek Gültekin.