

# **Cyclic Scheduling Problem of a Flexible Robotic Cell with a Self-Buffered Robot**

**Fatma Mohamed Ali Al-Hindwan**

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

Master of Science  
in  
Industrial Engineering

Eastern Mediterranean University  
July 2018  
Gazimağusa, North Cyprus

Approval of the Institute of Graduate Studies and Research

---

Assoc. Prof. Dr. Ali Hakan Ulusoy  
Acting Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science in Industrial Engineering.

---

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

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

---

Asst. Prof. Dr. Hüseyin Güden  
Supervisor

---

Examining Committee

1. Asst. Prof. Dr. Elif Binboğa

---

2. Asst. Prof. Dr. Sahand Daneshvar

---

3. Asst. Prof. Dr. Hüseyin Güden

---

## ABSTRACT

Extensive usage of automatic processing in industries has created Flexible Manufacturing Systems (FMSs). In a robotic FMS there are some Computer Numerical Control (CNC) machines for processing the parts, there is an input buffer for keeping unprocessed parts and an output buffer for finished parts, and at least one robot for transporting the parts in the system and loading/unloading the machines, and a central computer controlling the system. Such systems provide advantage in flexibility and standardization in production systems and they have been employed in recent years in order to keep up with the market competition. In order to use the system efficiently the system should be scheduled carefully. In this content, the order of the robot actions such as robot movements and loading/unloading activities should be determined for maximizing the system's efficiency. When the system repeats a cycle in its run maximizing the efficiency is equivalent to minimizing the cycle time.

This thesis considers a robotic FMS in which there is a single self-buffered robot which has the ability to carry more than one part at a time in an inline robotic cell where parts produced are identical. The system repeats a cycle in its long run. The problem is to determine the schedule of the system for minimizing the cycle time. A Mixed Integer Programming (MIP) model of the problem is developed to find the optimal solutions. Since the developed MIP model could not solve the large size problems a Simulated Annealing meta-heuristic algorithm is developed to solve those problems. Performances of the proposed methods and considered robotic FMS cells are evaluated on several numerical instances. Numerically, it has been shown that the performances of the proposed methods are satisfactory and the performance

of the robotic FMS increases significantly by using a self-buffered robot up to some robot buffer capacity. After a point more robot buffer capacity becomes useless.

**Keywords:** Flexible Manufacturing Systems, Cyclic Robotic FMS Scheduling, Self-Buffered Robot.

## ÖZ

Sanayide otomasyona dayalı üretimin geniş şekilde kullanımı Esnek Üretim Sistemleri'ni (EÜSleri'ni) ortaya çıkarmıştır. Robotlu bir EÜS'de parçaları işlemek için CNC makineler, işlenmemiş parçaları tutmak için bir stok alanı, bitmiş parçalar için bir stok alanı, parçaları sistemde taşımak ve makineleri yüklemek/boşaltmak için en az bir robot ve sistemi kontrol eden bir merkezi bilgisayar vardır. Böyle sistemler esneklik ve standartlaşma konularında avantaj sağlamaktadır ve son yıllarda rekabet edebilmek için tercih edilmektedirler. Sistemin verimli bir şekilde kullanılabilmesi için dikkatlice çizelgelenmesi gerekmektedir. Bu kapsamda, robot hareketi ve yükleme/boşaltma gibi robot faaliyetleri sistem verimliliğini en büyüleyecek şekilde belirlenmelidir. Sistem çalışırken bir döngüyü tekrarlıyor ise döngü süresinin en küçüklenmesi sistem verimliliğinin en büyüklenmesiyle aynıdır.

Bu tezde, birden fazla parçayı aynı anda taşıyabilecek kendi stok alanına sahip bir robotun bulunduğu ve parçaların özdeş olduğu bir robotlu EÜS ele alınmıştır. Sistem uzun süreli çalışmasında bir döngüyü tekrarlar. Problem, döngü süresini en küçükleyecek sistem çizelgesinin bulunmasıdır. En iyi çözümleri bulmak üzere problemin Karışık Tamsayı Programlama (KTP) modeli geliştirilmiştir. Geliştirilen KTP modeli büyük boyutlu problemleri çözemediği için bu problemleri çözmek amacı ile Tavlama Benzetimi modern-sezgisel algoritması geliştirilmiştir. Geliştirilen yöntemlerin ve ele alınan EÜS hücresinin performansları çeşitli sayısal problemler üzerinden değerlendirilmiştir. Önerilen yöntemlerin memnun edici bir performansa sahip oldukları ve ele alınan robotlu EÜS'nin performansının kendi stok alanı olan robot kullanılarak belli bir stok kapasitesine kadar önemli derecede arttığı sayısal

olarak gösterilmiştir. Belli bir noktadan sonra daha fazla robot stok alanının olması faydasızdır.

**Anahtar Kelimeler:** Esnek Üretim Sistemleri, Döngülü Robotlu EÜS Çizelgeleme, Kendinden Stok Alanlı Robot.

## **DEDICATION**

I dedicate this thesis to my late father who had a dream one day that I was on top of the mountains and he told me to him that meant that someday I will reach great heights to achieve my countless dreams. He was the reason I never gave up to become at the top of everything I ever work on.

## ACKNOWLEDGMENT

I would like to extend my gratefulness and indebtedness to my Supervisor Assist. Prof. Dr. Hüseyin Güden who has continuously shared his wisdom and knowledge which has indefinite description and for guiding, advising and directing me through what has been only what I can describe as an interesting journey through my thesis. He saw the ability in my performance from the beginning of my Master journey and he gave me the motivation to put in as much effort as required to make an outcome that I can be proud of. Most of all, I would like to thank him for always encouraging me by making me believe that someday I would become a successful academic.

I would also like to thank all my Professors who have taught me during my Master Degree. Assoc. Prof. Dr. Gökhan İzbirak for teaching us to think outside the box so that we can be productive engineers. Assoc. Prof. Dr. Adham Mackieh for showing us the importance of how hypothesizing any factors can lead to a definite conclusion. Prof. Dr. Bela Vizvari for pointing out that even though we are mathematicians we should remember simplification is always better. And Assist. Prof. Dr. Sahand Daneshvar for making our exams hard enough so that we learn to work as hard as we can.

My special gratitude also goes to two of my great friends, Tareq Babaqi and Katriye Dalcı for being always supportive and helpful during the period of my Masters by providing me with encouraging environment. Lastly, I would like to thank my family because without them I would have never had the pleasure of writing this acknowledgment to being with.



# TABLE OF CONTENTS

ABSTRACT.....	iii
ÖZ.....	v
DEDICATION.....	vii
ACKNOWLEDGMENT.....	viii
LIST OF TABLES.....	xii
LIST OF FIGURES.....	xiv
1 INTRODUCTION.....	1
1.1 Motivation.....	6
2 LITERATURE REVIEW.....	7
2.1 Robots with Single Gripper.....	9
2.2 Robots with Dual Gripper.....	16
2.3 Robots with either Single or Dual Gripper.....	18
2.4 Robots with Dual Arm.....	19
2.5 Robots with Swap Ability.....	19
2.6 Robots with Output Buffer at Each Machine.....	20
2.8 Robots with Input and Output Buffer at Each Machine.....	21
2.9 Robot with Buffer Capacity.....	22
3 PROBLEM DEFINITON.....	23
3.1 Notations and Definitions.....	23
3.2 Distance Matrix Derivation.....	25
3.3 Process Time.....	33
4 METHODOLOGY.....	35
4.1 Mixed Integer Programming Model.....	35

4.2 Software Used to Solve the Model .....	45
4.3 Cycle and Variables Representation .....	46
4.4 Simulated Annealing Algorithm .....	48
4.4.1 Creating the Initial Current Order .....	49
4.4.2 Defining x Variables and Setting Best Order.....	51
4.4.3 Starting the Iteration Loop .....	52
4.4.4 Strategy Used for New Solution .....	52
4.4.5 Condition for Finding a Better Solution .....	56
5 RESULTS AND DISCUSSION .....	58
5.1 Cycle Time Calculation .....	58
5.2 Cycle Time for Robot Buffer Capacity $> m$ and $\leq 2m$ .....	64
5.3 Comparison between MIP Model and SA Algorithm Results.....	67
5.3.1 Case 1: $\delta = 2, \epsilon = 1, P = 0$ .....	68
5.3.2 Case 2: $\delta = 2, \epsilon = 1, P = 22$ .....	76
5.3.3 Case 3: $\delta = 2, \epsilon = 1, P = 40$ .....	85
5.3.4 Case 4: $\delta = 2, \epsilon = 1, P = 50$ .....	90
5.3.5 Case 5: $\delta = 2, \epsilon = 1, P = 100$ .....	92
5.3.6 Case 6: $\delta = 2, \epsilon = 1, P = 5000$ .....	97
5.4 Effect of Robot Buffer Capacity on Cycle Time .....	101
5.5 Cycle Time is same for Various Process Times .....	108
5.6 Cycle Time Increase with Increase in Process Time .....	108
5.7 Cycle Time after Robot Buffer Capacity $> 1$ is Constant for Large Process Times.....	112
5.8 Waiting Time is considered for One Machine .....	113
6 CONCLUSION AND FUTURE RECOMMENDATIONS .....	118

6.1 Conclusions.....	118
6.2 Future Recommendations .....	120
REFERENCES .....	121
APPENDICES .....	128
Appendix A: Simulated Annealing Solution Time Convergence Graphs for Case when $\delta = 2$ , $\varepsilon = 1$ , $P = 0$ , $P = 22$ , $P = 40$ , $P = 50$ and $P = 100$ .....	129
Appendix B: Simulated Annealing Solution Time Convergence Graphs for Case when $\delta = 2$ , $\varepsilon = 1$ , $P = 5000$ .....	143

## LIST OF TABLES

Table 2.1: Literature Survey Table .....	8
Table 4.1: Representation of Constraint (4.2) and (4.3) .....	38
Table 5.1: Case when $m = 2, \delta = 2, \varepsilon = 1, K = 1, P = 22$ .....	58
Table 5.2: Describing Completion Time Calculation for Case when $m = 2, \delta = 2, \varepsilon = 1, K = 1, P = 22$ .....	60
Table 5.3: Case when $m = 2, \delta = 2, \varepsilon = 1, K = 2, P = 22$ .....	63
Table 5.4: Case when $m = 2, \delta = 2, \varepsilon = 1, K = 3, P = 22$ .....	65
Table 5.5: Case when $m = 2, \delta = 2, \varepsilon = 1, K = 4, P = 22$ .....	66
Table 5.6: Case when $\delta = 2, \varepsilon = 1, P = 0$ .....	68
Table 5.7: Case when $m = 4, \delta = 2, \varepsilon = 1, K = 2, P = 0$ .....	71
Table 5.8: Case when $m = 4, \delta = 2, \varepsilon = 1, K = 3, P = 0$ .....	71
Table 5.9: Case when $m = 4, \delta = 2, \varepsilon = 1, K = 4, P = 0$ .....	73
Table 5.10: Case when $m = 5, \delta = 2, \varepsilon = 1, K = 1, P = 0$ .....	73
Table 5.11: Case when $m = 5, \delta = 2, \varepsilon = 1, K = 2, P = 0$ .....	74
Table 5.12: Case when $m = 5, \delta = 2, \varepsilon = 1, K = 3, P = 0$ .....	74
Table 5.13: Case when $m = 5, \delta = 2, \varepsilon = 1, K = 4, P = 0$ .....	75
Table 5.14: Case when $m = 5, \delta = 2, \varepsilon = 1, K = 5, P = 0$ .....	75
Table 5.15: Case when $\delta = 2, \varepsilon = 1, P = 22$ .....	76
Table 5.16: Case when $m = 4, \delta = 2, \varepsilon = 1, K = 2, P = 22$ .....	79
Table 5.17: Case when $m = 4, \delta = 2, \varepsilon = 1, K = 4, P = 22$ .....	82
Table 5.18: Case when $m = 5, \delta = 2, \varepsilon = 1, K = 3, P = 22$ .....	83
Table 5.19: Case when $m = 5, \delta = 2, \varepsilon = 1, K = 5, P = 22$ .....	84
Table 5.20: Case when $\delta = 2, \varepsilon = 1, P = 40$ .....	86

Table 5.21: Case when $m = 5, \delta = 2, \varepsilon = 1, K = 6, P = 40$ .....	88
Table 5.22: Case when $\delta = 2, \varepsilon = 1, P = 50$ .....	91
Table 5.23: Case when $\delta = 2, \varepsilon = 1, K = 2, P = 100$ .....	94
Table 5.24: Case when $m = 5, \delta = 2, \varepsilon = 1, K = 1, P = 100$ .....	95
Table 5.25: Case when $m = 5, \delta = 2, \varepsilon = 1, K = 2, P = 100$ .....	96
Table 5.26: Case when $\delta = 2, \varepsilon = 1, P = 5000$ .....	99
Table 5.27: Case when $m = 5, \delta = 2, \varepsilon = 1, K = 1, P = 5000$ .....	100
Table 5.28: Case when $m = 3, \delta = 2, \varepsilon = 1, K = 1, P = 0$ .....	109
Table 5.29: Case when $m = 3, \delta = 2, \varepsilon = 1, K = 1, P = 22$ .....	109
Table 5.30: Case when $m = 3, \delta = 2, \varepsilon = 1, K = 1, P = 40$ .....	110
Table 5.31: Case when $m = 3, \delta = 2, \varepsilon = 1, K = 1, P = 50$ .....	110
Table 5.32: Case when $m = 3, \delta = 2, \varepsilon = 1, K = 1, P = 100$ .....	111
Table 5.33: Case when $m = 3, \delta = 2, \varepsilon = 1, K = 1, P = 5000$ .....	111
Table 5.34: Relation of Process Times, Number of Machines and Robot Buffer Capacity .....	113
Table 5.35: Case when $m = 2, \delta = 2, \varepsilon = 1, K = 2, P = 50$ .....	114
Table 5.36: Case when $m = 2, \delta = 2, \varepsilon = 1, K = 2, P = 100$ .....	114
Table 5.37: Case when $m = 3, \delta = 2, \varepsilon = 1, K = 2, P = 50$ .....	115
Table 5.38: Case when $m = 3, \delta = 2, \varepsilon = 1, K = 2, P = 100$ .....	115

## LIST OF FIGURES

Figure 1.1: Inline Robotic Cell with a Self-Buffered Robot.....	5
Figure 3.1: Case when Activity $a \in \{L_i, U_i\}$ and Activity $b \in \{L_j, U_j\}$ .....	25
Figure 3.2: Example when Activity $a \in \{L_1\}$ and Activity $b \in \{L_2\}$ .....	26
Figure 3.3 a: Case when Activity $a \in \{L_i, U_i\}$ and Activity $b \in \{I\}$ .....	27
Figure 3.3 b: Case when Activity $a \in \{I\}$ and Activity $b \in \{L_i, U_i\}$ .....	27
Figure 3.4: Example when Activity $a \in \{L_2\}$ and Activity $b \in \{I\}$ .....	28
Figure 3.5 a: Case when Activity $a \in \{L_i, U_i\}$ and Activity $b \in \{O\}$ .....	29
Figure 3.5 b: Case when Activity $a \in \{O\}$ and Activity $b \in \{L_i, U_i\}$ .....	29
Figure 3.6: Example when Activity $a \in \{L_i, U_i\}$ and Activity $b \in \{O\}$ .....	30
Figure 3.7 a: Case when Activity $a \in \{I\}$ and Activity $b \in \{O\}$ .....	31
Figure 3.7 b: Case when Activity $a \in \{O\}$ and Activity $b \in \{I\}$ .....	31
Figure 3.8: Example when Activity $a \in \{I\}$ and Activity $b \in \{O\}$ .....	32
Figure 3.9: Case when Activity $a, b \in \{I\}$ .....	32
Figure 3.10: Case when Activity $a, b \in \{O\}$ .....	33
Figure 4.1: Representation of Constraint (4.4) by an Example of Case 1 .....	39
Figure 4.2: Representation of Constraint (4.6) by an Example of Case 1 .....	41
Figure 4.3: Representation of Constraint (4.7) .....	41
Figure 4.4: Representation of Constraint (4.8) .....	42
Figure 4.5: Solution Page for 2 Machine Case with $K = 1, P = 22$ .....	47
Figure 4.6: Representation of a Cycle for a 2 Machine Case .....	48
Figure 4.7: Creating Initial Current Order .....	49
Figure 4.8: Initial Current Order Created for 2 Machine Case .....	50

Figure 4.9: Initial Current Order for 2 Machine Case Represented by Activities and Machines .....	50
Figure 4.10: General Initial Current Order .....	51
Figure 4.11: General Initial Current Order Represented by Activities L, U, I and O.51	
Figure 4.12: Setting x Variables to Lower Bound 1 .....	51
Figure 4.13: Swapping Strategy for New Solution to be generated.....	53
Figure 4.14: Swapping Strategy to Generate New Solution .....	54
Figure 4.15: Swapping Strategy for Robot Buffer Capacity 1.....	54
Figure 4.16: New Orders generated for 2 Machines with $K = 1$ .....	55
Figure 5.1: Gantt Chart for Case when $m = 2, \delta = 2, \varepsilon = 1, K = 1, P = 22$ .....	62
Figure 5.2: Comparison between MIP and SA Cycle Time for 2 Machines with $P = 0$ .....	69
Figure 5.3: Comparison between MIP and SA Cycle Time for 3 Machines with $P = 0$ .....	69
Figure 5.4: Comparison between MIP and SA Cycle Time for 2 Machines with $P = 22$ .....	77
Figure 5.5: Comparison between MIP and SA Cycle Time for 3 Machines with $P = 22$ .....	77
Figure 5.6: Gantt Chart for Case when $m = 4, \delta = 2, \varepsilon = 1, K = 2, P = 22$ .....	81
Figure 5.7: Comparison between MIP and SA Cycle Time for 2 Machines with $P = 40$ .....	85
Figure 5.8: Comparison between MIP and SA Cycle Time for 3 Machines with $P = 40$ .....	85
Figure 5.9: Gantt Chart for Case when $m = 5, \delta = 2, \varepsilon = 1, K = 6, P = 40$ .....	89

Figure 5.10: Comparison between MIP and SA Cycle Time for 2 Machines with P = 50.....	90
Figure 5.11: Comparison between MIP and SA Cycle Time for 3 Machines with P = 50.....	90
Figure 5.12: Comparison between MIP and SA Cycle Time for 2 Machines with P = 100.....	92
Figure 5.13: Comparison between MIP and SA Cycle Time for 3 Machines with P = 100.....	93
Figure 5.14: Comparison between MIP and SA Cycle Time for 2 Machines with P = 5000.....	97
Figure 5.15: Comparison between MIP and SA Cycle Time for 3 Machines with P = 5000.....	98
Figure 5.16: Robot Buffer Capacity Effect on Cycle Time for 2 Machines.....	101
Figure 5.17: Robot Buffer Capacity Effect on Cycle Time for 2 Machines and P = 5000.....	101
Figure 5.18: Robot Buffer Capacity Effect on Cycle Time for 3 Machines.....	102
Figure 5.19: Robot Buffer Capacity Effect on Cycle Time for 3 Machines and P = 5000.....	102
Figure 5.20: Robot Buffer Capacity Effect on Cycle Time for 4 Machines.....	104
Figure 5.21: Robot Buffer Capacity Effect on Cycle Time for 4 Machines and P = 5000.....	105
Figure 5.22: Robot Buffer Capacity Effect on Cycle Time for 5 Machines.....	106
Figure 5.23: Robot Buffer Capacity Effect on Cycle Time for 5 Machines and P = 5000.....	107
Figure 5.24: Gantt Chart for Case when $m = 5, \delta = 2, \epsilon = 1, K = 6, P = 40$ .....	117



# Chapter 1

## INTRODUCTION

A flexible manufacturing system has been widely used over the years because it has the ability to adapt to changes in the process of production without causing any delays and thus increasing the efficiency while processing products at a faster pace. Machine and routing flexibility are the two types offered by a flexible manufacturing system for the production process to be improved. When a new product type is created the system changes to adjust itself and this refers to machine flexibility. While the performance of the same set of operations by all the machines are referred to as routing flexibility.

Extensive scheduling and designing of the system is required which makes it complicated and thus only skilled workers can run such systems in industries which lead to high cost and this poses as a disadvantage of a flexible manufacturing system. However, defective products are prevented because of the ability of the system to adapt to changes in the product and thus the cost is reduced in the long run.

The increase in flexibility which in turn increases the profit of an industrialized company that operates a production line automatically by implementing modern technology such as flexible manufacturing systems that are made up of robots are known as flexible manufacturing robotic cells.

Since most industries strive in the market competition, they have chosen to use flexible robotic cells for higher production and greater efficiency and some of these industries include: aerospace, automotive, metal conductors and machinery. Some real examples of where FMS are applied include the Ingersoll-Rand Corporation for the hoist division in Virginia, USA. They built a parallel track cell with drill machines and machining centers on either sides of the track and a roller conveyor to transport processed parts. Aluminum and cast iron castings are the unfinished parts that enter the production line and finally motor cases are the finished products. Other industries include the Vought Aerospace in Dallas, USA where the FMS is made up of CNC machines that produce different components of aircrafts and Allen-Bradley Company that has 26 workstation cells to produce motor starters and many more.

Generally, a robotic cell consists of input and output buffers at the beginning and end of a production line so that unfinished parts can be stored at the input buffer and finished products are stored at the output buffer. Then between those buffers there are  $m$  machines that process the parts and these are CNC machines which are computer controlled. A single robot or in some cases multiple robots are responsible for handling the parts and transporting them between the machines and from or to the input and output buffer and also for loading and unloading operations.

There are two types of commonly used robotic cell environments which is either flowshop or jobshop. A flowshop robotic cell means that a part has to go through all the machines for processing in the same sequence. However, a jobshop robotic cell has machines that perform the same set of operations so a job can be processed by any of the machines. Other characteristics of a robotic cell include the number of robots used which is either single or multiple. Usually, when multiple robots are used

there should more than one track for them to move because a single track means that there will be collision between robots. Therefore studies proposing collision free robotic cells with multiple robots have been considered.

During the start of robotic cell studies the most common robot type was a single gripper robot which means the robot only had the ability to handle one part at a time. Later on, studies proposed a dual gripper which can handle two parts at once. So for example, a robot will pick up an unfinished part from the input buffer and then move to a machine that finished processing a part rotates the wrist to pick up the finished part and then rotates again to load the machine with the unfinished part. Another type that has not been studied extensively is the dual arm robot which differs from a dual gripper robot since it can place the arms at two successive machines consequently rather than working on the same machine at a time. Other types of robots include swap able robots and multifunction robots with hybrid grippers.

Layout of a production line is either inline or circular which from its name indicates that the machines are placed in series in an inline layout. Circular layout is usually seen to be more efficient because the robot has the ability to move clockwise or anticlockwise thus decreasing the travel time. When the travel time is  $|i - j|\delta$  between two consecutive machines then it is known as additive travel time for  $0 \leq i, j \leq m + 1$ . However, most studies consider a constant travel time  $\delta$  between any pair of machines. And finally there is Euclidean travel time  $\delta_{ij}$  which must satisfy three properties and one of them is the triangle inequality  $\delta_{ij} + \delta_{jk} \geq \delta_{ik}$ .

When the robot travels between machines it can either pick up a part as soon as it finishes processing which defines the no wait constraint or it can leave the part on the

machine for an undefined amount of time which is referred to as the free pickup criteria. Sometimes the part can stay on the machine for an interval and that is known as the interval pickup criteria.

The processing of parts is repeated in a similar sequence and the movement of the robot is fixed to repeat the sequence which represents a cycle. Parts processed may either be identical or non-identical. In one cycle when one part is processed the strategy of production is called 1-unit. If two parts are processed in a cycle then it is a 2 unit cycle and so on. In a cycle a set of operations are performed which is repeated in the same sequence in the next cycle. Cycle time is the duration of one cycle and the aim of most studies is to schedule the order of robot moves in a manner that reduces cycle time which in turn meets the common objective of maximizing throughput. Such a problem is an optimization problem.

In this study the main contribution is to study self-buffered robots which means the robot has a buffer that can store any amount of parts. For example, if the robot buffer capacity is 2 it means the robot can store one part and handle one part. Figure (1.1) shows an example of a self-buffered robot with the ability to handle 2 parts. The main problem is minimizing cycle time by scheduling robot moves for a robotic cell with a self-buffered robot.

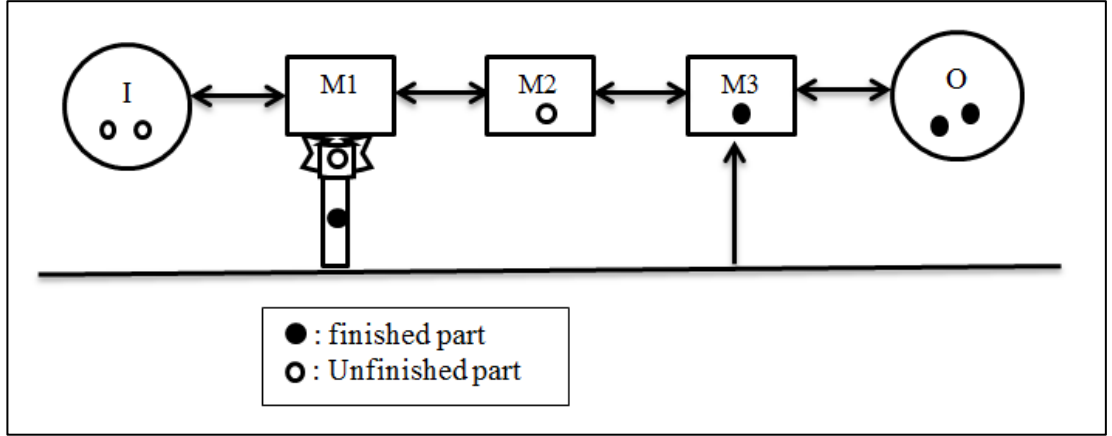


Figure 1.1: Inline Robotic Cell with a Self-Buffered Robot

The study is considered for 2, 3, 4 and 5 machines case where the CNC machines are identical in a parallel machine flowshop environment with one machine in each stage and an inline layout with a single robot that has a single gripper and the parts produced are identical and the robot travel time is additive which means the travel time between any two machines is  $|i - j|\delta$ . In one cycle only one part is processed by each machine indicating it is a 1 unit cycle and the criterion of pick up is free.

Any scheduling problem is classified in terms of  $\alpha|\beta|\gamma$  where  $\alpha$  refers to machine environment  $\alpha = R_{m,r}^{g,l}(m_1, \dots, m_m)$  where R defines the environment, g defines robot type, l defines layout, m defines number of stages, r defines number of robots and  $(m_1, \dots, m_m)$  indicates how many machines are there in one stage. The processing characteristics  $\beta = (\text{pickup criteria, travel time metric, part type, production strategy})$  and the objective function is  $\gamma$ . Our scheduling problem will be classified as follows:

$$RF_{m,1}^{1,|}|(free, additive, Cyclic - 1)|Minimize cycle time$$

Initially, an MIP model for the problem is created and used to find exact solutions for 2 and 3 machines case. However, since solution time is extremely long for robotic cells with 4 machines or more, a heuristic approach is used by applying the simulated annealing algorithm and then comparison between exact and heuristic approach is analyzed.

This study will be classified into the following: Chapter 2 will include a literature review of all common studies, Chapter 3 will include problem definition, Chapter 4 comprises methodology, Chapter 5 includes the results and discusses the relative meaning of the results found and finally Chapter 6 concludes the study.

## **1.1 Motivation**

Flexible Manufacturing Cells as seen are employed by several industries and the efficiency of the FMC system depends entirely on the schedule of the system. Finding the most efficient schedule for such an FMC system is an important factor to most industries. Because of its importance there is a huge literature studying this optimization. However, studies mainly focus on robots of different types and a major gap was realized concerning a self-buffered robot. Currently only one study has considered a self-buffered robot in such systems which may increase the performance of the FMC significantly.

## **Chapter 2**

### **LITERATURE REVIEW**

In this study, we consider a flexible robotic cell with  $m$  machines that are served by a single robot with a single gripper that has a buffer capacity and thus the ability to store finished or unfinished parts along with handling one part when the capacity of the buffer is more than one. The layout of the cell is inline and has one input buffer that stores unfinished products and one output buffer that stores finished products. The travel time of the robot between the machines or between the input buffer and a machine or between the machine and an output buffer is additive and a product can stay on the machine for any amount of time corresponding to the free pickup criteria. Parts produced are identical and only one unit is processed in a cycle. The objective is to minimize cycle time by scheduling robot moves and the main focus of the study is to show how the robot buffer capacity can reduce cycle time compared to single gripper robots with no buffer capacity. An MIP model of the problem was created and was used to solve the problem for an exact solution. However, large problems were solved by applying the simulated annealing heuristic algorithm.

Since this study considered a production environment that is flowshop and when there is only a single robot, all the literature that will be reviewed in this following chapter will contain articles that considered the same environment and number of robots. Differences will be seen among the robot type and whether it is single gripper, dual gripper, dual arm, and robot with swap ability and whether there are

machine or robot buffers considered since the main focus of this study has to do with the type of robot and whether applying a buffer capacity will have an impact on the cycle time. Table (2.1) shows all the articles that were surveyed.

Table 2.1: Literature Survey Table

Article No.	No. of Machines	Type of Robot	Layout	Pickup Criteria	Travel Time Metric	Part Types	Prod. Strategy	Buffer space	Solution Method
1	M machines	Single	Circular, Inline	Blocking	Constant	Identical	1 unit		TSP, Polynomial
2	M machines	Single	Circular	Blocking	Constant	Identical	1 unit		NP complete proof
3	2, 3 machines	Single	Mobile		Constant	Multiple, Identical	MPS		Algorithm solved by GAMS
4	M machines	Single		No wait	Additive	Identical	1 unit		Polynomial
5	3 machines	Single	Mobile		Constant	Multiple	MPS		Unary NP complete proof
6	M machines	Single	Mobile	Free	Additive	Multiple	MPS		Polynomial
7	3 machines	Single	Inline	Blocking	Constant	Identical	1 unit		Proof of Sethi et al. conjecture
8	3 machines	Single	Inline		Additive	Multiple	MPS		Heuristic approach
9	3 machines	Single	Circular	No wait	Additive	Multiple	1 unit		Polynomial
10	2, 3 machines	Single	Circular	No wait	Additive	Multiple, Identical	1 unit, 2 unit		Polynomial
11	2, 3 machines	Single	Circular	No wait	Additive	Identical	1 unit, 2 unit		Pyramidal permutation
12	M machines	Dual	Circular		Additive	Identical	1 unit		Permutation
13	M machines	Single	Circular	No wait	Symmetric, Additive, General	Identical	1 unit		TSP, NP hardness
14	5 and M machines	Single	Inline	No wait	Additive	Identical	K unit		Lower bound method
15	2 machines	Dual	Circular		Constant	Multiple	MPS		Heuristic approach
16	2 and M machines	Dual	Circular	No wait	Additive	Multiple	MPS		Gilmore Gomory, Polynomial
17	M machines	Single	Circular	Blocking	Additive, Constant, Euclidean	Identical	K unit		Polynomial
18	2 machines	Single	Inline	No wait	Constant	Identical	1 unit, 2 unit		Sensitivity analysis
19	M machines	Dual	Circular	Free	Additive	Identical	1 unit	Each machine has an O	Comparative Study
20	2 machines	Single	Inline	No wait	Additive	Identical	1 unit, 2 unit		Sensitivity analysis
21	M machines	Single	Circular	Free	Additive, Constant, Euclidean	Identical			Optimal schedules
22	3 machines	Single	Inline	No wait	Additive	Identical	1 unit, 2 unit		Dominant cycles
23	M machines	Single	Circular	Free	Additive	Identical	1 unit		Polynomial
24	M machines	Single	Inline	No wait	Euclidean	Identical	1 unit		Branch and bound algorithm



Article No.	No. of Machines	Type of Robot	Layout	Pickup Criteria	Travel Time Metric	Part Types	Prod. Strategy	Buffer space	Solution Method
25	M machines	Single, Dual	Circular	Free	Constant	Identical	K unit		Polynomial
26	M machines	Dual	Parallel machine, Circular	Free	Constant	Identical	K unit		Lower bound method
27	M machines	Single	Inline	Free	Additive	Multiple	MPS	Each machine has an O	Dominant cycles
28	M machines	Dual		Interval, No wait, Free	Constant	Identical	1 unit		Polynomial
29	M machines	Single	Circular	No wait	Euclidean	Identical	1 unit		Branch & Bound
30	M machines	Single	Circular	Free	Additive	Identical	1 unit	Each machine has I/O	Structural Analysis
31	M machines	Single	Inline, Circular	Free	Additive	Identical	1 unit		Polynomial
32	M machines	Dual	Circular	Free	Additive	Identical	K unit	Each machine has I/O	Lower bound method
33	3 and M machines	Single	Circular	Interval	Additive	Identical	K unit		Lower bound method
34	2 machine	Single with swap able	Circular vs inline	Free, No wait	Additive	Identical	K unit		Pure cycle performance
35	2 machines	Single with swap able	Inline		Additive	Identical	1 unit		Sensitivity analysis
36	M machines	Single	Inline	No wait	Euclidean	Identical	K unit		MIP model, solved by CPLEX
37	2, 3 and M machines	Dual arm	Circular	Free	Additive	Identical, Multiple	1 unit, K unit		Dominant cycles
38	2 machines	Single	Circular	Free	Additive	Multiple	1 unit		MILP, Branch & bound, Simulated Annealing
39	M machines	Dual	Inline	Free	Additive	Identical	2 unit		Lower bound method
40	M machines	Single	Inline		Euclidean	Multiple	K unit		Branch and bound
41	M machines	Single	Parallel machine, Inline	Free	Additive	Multiple			MILP, Simulated Annealing
42	M machines	Single, Dual	Circular	Free	Additive	Identical	K unit		Polynomial
43	2 machines	Single, Dual	Inline	Interval	Additive	Identical	1 unit	Robot with buffer	Dominant cycles
<b>This study</b>	M machines	Single	Parallel machine, Inline	Free	Additive	Identical	1 unit	Robot with buffer	MIP, Simulated Annealing

## 2.1 Robots with Single Gripper

In the current literature some of the studies that considered **single gripper** robots include [1 – 11], [13 – 14], [17 – 18], [20 – 24], [29], [31], [33], [36], [38], [40 – 41] and out of these studies the articles that considered m machines case are [1 – 2], [4],

[6], [13 – 14], [17], [21], [23 – 24], [29], [31], [33], [36] and [40 – 41]. Robotic cell with 2 machines were studied in [3], [10 – 11], [18], [20], [38] and those with 3 machines were considered in [3], [5], [7 – 11], [22] and [33]. A 5 machine robotic cell was also studied in [14]. In terms of the layout of the cell, [1 – 2], [9 – 11], [13], [17], [21], [23], [29], [31], [33] and [38] implemented a circular layout. Whereas [1], [7 – 8], [14], [18], [20], [22], [24], [31], [36], [40] and [41] considered an inline layout. A mobile layout which is a generalization of the other two layouts was studied earlier in [3] and [5 – 6]. The pickup criteria studied by [4], [9 – 11], [13 – 14], [18], [20], [22], [24], [29] and [36] was the no wait pick up criteria. On the other hand, [6], [21], [23], [31], [38] and [41] implemented the free pickup criteria and [33] studied interval pickup criteria. The blocking constraint states that when a part finishes processing on a machine, the machine does not have the ability to process another part until the finished part has been unloaded by the robot and this was seen in [1 – 2], [7] and [17]. Additive travel time metric was considered in [4], [6], [8 – 11], [13 – 14], [17], [20 - 23], [31], [33] and [41]. Constant travel time metric was implemented in [1 – 3], [5], [7], [17 – 18] and [21] and the Euclidean travel time was studied by [17], [21], [24], [29], [36] and [38]. Identical parts were produced in [1 – 4], [7], [10 – 11], [13 – 14], [17 – 18], [20 – 24], [29], [31], [33] and [36]. Whereas multiple parts were produced in [3], [5 – 6], [8 – 10], [38] and [40 – 41]. In a cycle when one unit is produced it is called a 1-unit cycle and this was studied in [1 – 2], [4], [7], [9 – 11], [13], [18], [20], [22 – 24], [29], [31] and [38]. A 2-unit cycle was considered in [10 – 11], [18], [20] and [22] and a k-unit cycle was studied by [14], [17], [33], [36] and [40]. And finally MPS cycles is a cycle strategy that can be implemented only by multiple part types and its partitions sets with identical parts

and produces each set in one cycle and thus the name Minimal Part Set and this was studied in [3], [5 – 6] and [8].

Even though some similarities and differences can be seen among these papers in terms of the characteristics of the robotic cell. The main difference between them lies in the contribution of each article.

The problem in [1] is finding the robot cyclic schedule that is the shortest for the cyclic scheduling of identical parts. In contrast to previous papers, the problem in this paper is a case where they consider  $m$  arbitrary number of machines, but all parts are the same.  $O(m^3)$  polynomial time is solved by an approach of dynamic programming. Pyramidal permutations are a concept relied on heavily in the paper's analysis which is connected to the travelling salesman problem. In [2] proof that the problem is strongly NP complete was undertaken. In [3] for a robotic cell with two machines and part types are multiple, an efficient algorithm is provided that was used to optimize the problem of part sequencing and move cycles of robots. A computational program known as General Algebraic Modelling System (GAMS) was used to test the algorithm. For a robotic cell with 3 machines and part types are identical, the repetition of 1 unit cycles optimality has a conjecture that was addressed and it was shown that the cycles that are more complicated is dominated by such a procedure when two units are produced.

The Cyclic Robotic Flowshop Problem (CRFP) in the version that is solved in polynomial time is studied in [4]. Processing times are numerical in the problem and the no wait constraint is considered. When triangle inequality is satisfied by the operation times, it is shown that  $O(m^5)$  can be improved. The problem is solved by

an algorithm that is derived in  $O(m^3 \log m)$  time. [5] proved that the problem of part sequencing is unary NP complete when the version is recognition and that when one unit is produced; the robot move cycles that are potentially optimal are 2 out of 6. Part sequencing problems that are solved efficiently are defined by the remaining four cycles.

Considering the classification of the cycles of robots moves associated with the problem of part sequencing was proved in [6] that out of the  $m!$  available cycles of robot moves exactly  $2m-2$  are solved in polynomial time. While the cycles that are remaining associated with the problem of part sequencing are unary NP-hard. For a robotic flowshop with 3 machines, it was conjectured in [7] that optimal production is yielded when the unit cycle is 1. The conjecture validity was established. In [8] move cycles of robots and the sequence of parts that together lead to cycle time minimization which is required to produce a set of minimal parts are determined. Previous algorithms that were provided and intractability proofs for different cell configurations are used for a heuristic procedure to be developed for the problem of part sequencing for different move cycles of robots in a robotic cell with 3 machines. They described how the heuristic methodology can be extended for a robotic cell with 4 machines and they tested it.

The complexity of sequencing of parts problem is analysed for a 3-machine robotic cell when moves of robots are of different periodical patterns. Complexity is investigated for six possible 1 unit cycles in the problem of part sequencing in [9]. The optimization and feasibility problem for each of them is considered and it was seen that out of the six cases, the problem of feasibility is polynomial for 4 of them and NP complete for the other two. In [10] two cases were studied: Case of 2

machines and parts processed are multiple: after being reduced to classical flowshop no wait problem of 2 machines, it was seen that the problem is solved by  $O(n \log n)$  polynomial algorithm and case of 3 machines and parts processed are identical: move cycles of robots are considered when parts visit the machine either once or twice. In [11] there is a threefold contribution:

1. Active schedules which are the so-called notion on cycle times are discussed in more detail. In this case, the no wait criterion is applied in which operations are executed by robots as early as possible.
2. Conjecture of one cycle is presented in a new approach.
3. Conjecture of one cycle is settled completely. Counterexamples are constructed by the new approach which proves that for  $m \geq 4$  machines the conjecture is not valid any longer. Two cases were distinguished: 1. equidistant machines when cells are regular. 2. The non-regular cells. And then the dominance of the cycle was demonstrated to be different for two configurations.

[13] studies the computational complexity of finding robot moves shortest route between one machine and the next. Even though this complex problem was discussed in previous literature, previous studies took into consideration some assumptions which were dropped in this paper and NP hardness is proved in the strong sense when there are symmetric travel times between the robotic cell machines and triangle inequality is satisfied. The robotic cell scheduling problem considered in [14] has processing windows which are unbounded. A conjecture was presented which provides production cycle optimality with structure. Lower bound method was used to prove optimality of the conjecture. Results confirm Agnetis conjecture that claims

dominant cycles degree in a robotic cell that is in a no wait condition can be bounded by  $m-1$  machines. Agnetis proves conjecture for  $m = 2$  and 3. [14] studies for  $m$  and  $m = 5$ .

In [17] a polynomial algorithm was presented that produce solutions for multiunit cycles for classes of robotic cells which are most commonly known: constant, additive and Euclidean travel time. The optimal solutions are within constant factor per unit cycle time. In [18] the objective was to find on 2 machines the process times by operations being allocated to the machines as well as finding the move cycles of robot that will jointly lead to cycle time minimization. Rather than the previous proof that 1-unit robot move cycles are optimal, it was proved that either 1 or 2-unit robot move cycles are optimal depending on given parameters.

Generally, it is assumed that since a tool magazine is stored with all tools that are required, the operations can be performed by the CNC machines. But, the capacity of the tool magazine is limited and the numbers of tools which are usually required exceed the capacity. Thus, [20] considers the following assumption: due to constraints in tooling, operations can be performed on the first machine while others are performed on the second machine. While the operations that are remaining can be performed on either one of the two machines. In [21] the knowledge concerning cyclic schedules with respect to the robotic cells of different classes that are the three travel times: additive, constant and Euclidean was discussed. [22] considers part processing time as a decision variable for robot move cycles of 1 unit and 2 unit and a new lower bound was proposed. And also, a new robot move cycle was proposed which possesses the flexibility of operation. A cyclic solution is produced by an algorithm presented in [23]. The cycle time is a factor  $10/7$  of the optimal solution

per unit.  $O(m)$  time runs the algorithm where  $m$  corresponds to number of machines in the cell. Compared to the  $3/2$  best known guarantee, this result proved to be an improvement. [24] proposed an exact algorithm which is the branch and bound algorithm for a cyclic schedule to be optimal when processing times are flexible. Based on machine and robot capacity constraints, the cycle time prohibited intervals are used to formulate the problem. After the developed model is analysed, transformation of the problem is conducted for the nonprohibited intervals of cycle time to be enumerated.

[29] initially used method of prohibited intervals to formulate the problem and then interval bounds were linearly expressed, and subsets were used to divide the intervals and nonprohibited intervals were enumerated in each subset. In [31] NP hardness proof was conducted when 1-unit cycle optimality was obtained for a circular layout robotic cell with pickup criteria that is free and when the travel time metric is additive, and the throughput increase was assessed. [33] considered a special case of  $m = 3$  and they analysed the case when processing times are controllable and manufacturing cost associated with processing time was considered. Results proved that at least one of two pure cycles reach optimality and proved that pure cycles are dominant compared to classical cycles.

[36] used binary variables to define machine availability constraints in the scheduling problem of multicyclic robotic flowshop cell when formulating the MIP model, the input sequence is fixed which is not the case for multicyclic production. MIP model was solved by CPLEX software and generated instances that are random proved that the MIP approach proposed can solve scheduling problems in real life efficiently. [38] proved NP hardness for the two-machine robotic cell scheduling problem with

sequence dependent setup times. They also developed a time complex lower bound of the problem using the algorithm of Gilmore and Gomory. And finally, an MILP model was developed to address determination of best robot moves and parts sequencing.

[40] developed analytical properties and branch and bound scheme that are specific and efficient based on the problem characteristics, which allow the solution search process to eliminate infeasible or dominated solutions. [41] introduced additional constraints such as machine eligibility and parallel machines with different processing speeds at each stage. They developed an MILP model and minimized makespan for hybrid flowshop scheduling problem and a simulated annealing algorithm which used a neighbourhood structure with block properties was employed.

## **2.2 Robots with Dual Gripper**

The articles that studied **dual gripper** robots include [12], [15 – 16], [26], [28] and [39] and out of these studies the articles that considered m machines case are [12], [16], [26], [28] and [39]. Robotic cell with 2 machines were studied in [15 - 16]. In terms of the layout of the cell, [12], [15 – 16] and [26] implemented a circular layout. Whereas [39] considered an inline layout. The pickup criteria studied by [16] and [28] was the no wait pick up criteria. On the other hand, [26], [28] and [39] implemented the free pickup criteria and [28] studied interval pickup criteria. Additive travel time metric was considered in [12], [16] and [39]. Constant travel time metric was implemented in [15], [26] and [28]. Identical parts were produced in [12], [26], [28] and [39]. Whereas multiple parts were produced in [15] and [16]. 1-unit cycle was studied in [12] and [28]. A 2-unit cycle was considered in [39] and a



k-unit cycle was studied by [26]. And finally MPS cycles was studied in [15] and [16].

In [12] the analytical framework that exists in the literature was extended for all 1-unit cycles to be systematically developed for 2 machine robotic cells with a dual gripper robot and then the difference between dual gripper and single gripper is investigated in terms of cycle time advantage.

Finding part sequence optimality is known to be strongly NP hard even when they provide the sequence of robot moves. A framework which is modelled and notated is provided in [15] for the NP hard family of problems to be studied which are associated with robot move sequences which are optimal. An algorithm which is approximate is developed with the guarantee ratio of worst case performance of  $3/2$  which is estimated using a linear program without lower bound being calculated. The system operation at steady state under numerous options of cyclic scheduling was the focus of [16]. The problem of 2 machines was solved by the Gilmore Gomory heuristic approach on problem instances that are randomly generated. Testing procedures indicate that less than 10% of relative errors are realized when cycle time lower bound at optimality is compared. A comparison between single and dual gripper robots was carried out by conducting productivity gain estimation. There was between 18% and 36% realization in relative improvement. [26] provides insights to managers on how a dual gripper robot is beneficial and how a parallel machine cell along with a dual gripper robot is more beneficial. Throughput improvement are realised when such improvements are considered. [28] considers scheduling a robotic cell with a dual gripper robot. The cases considered initially are the no wait and free pick up cells. For the case when the pickup criteria is no wait, polynomial time

algorithm was used to find the optimal solution and when it was free, the algorithm was used to find the asymptotically optimal solution. For an interval robotic cell the problem was proved to be NP hard. Also results showed that throughput improved significantly when dual gripper was used rather than single gripper. In [39] a methodology for optimizing a robotic cell with a hub re-entrant machine was newly introduced. For all robot move cycles the cycle time is determined to find the lower bound of the cycle time for the dual gripper robot. The optimal sequence of robot tasks is determined which was a 2-unit cycle. The cycle time lower bound was also obtained for the dominant cycle and the optimal solution found for this cycle was demonstrated and they proved that for the robotic cell with hub machine that is re-entrant this is the most appropriate option.

### **2.3 Robots with either Single or Dual Gripper**

Studies that compared **single and dual gripper** robots include [25] and [42]. And in both those studies the robotic cell was assumed to have  $m$  machines and the layout of the cell was circular. The parts produced were identical with  $K$ -unit being produced per cycle with free pickup criteria. The only difference was that [25] implemented a constant travel time while [42] considered an additive travel time metric. [25] provides valuable insights to production managers regarding how productivity is maximized for both single and dual gripper cells for any combination of requirements for processing and physical parameters. [42] provides an insight into schedules for productivity maximization of either dual or single gripper robotic cells. And the performance of dual gripper robotic cells under relevant conditions was studied.

## 2.4 Robots with Dual Arm

A **dual arm** robot was studied by [37] in a cell with either 2, 3 or  $m$  machines and a circular layout. The travel time metric was additive and parts produced were either identical or multiple and thus either 1 unit or  $k$  unit per cycle were produced. The pickup criteria were free. [37] identified optimal sequence when identical parts are processed for 2 and 3 machine cases. Also cells with  $m$  machines were studied and they derived structural results for the case. For cells with two machines, they also analysed the case when parts processed are of multiple types. They proved that productivity was higher in dual arm robots compared to robot with single arm or single gripper and the gains realized were quantified.

## 2.5 Robots with Swap Ability

A special type of robot was proposed by [34] and [35] and that is a robot with **swap ability**. A robot with swap ability can handle only one part at once but the constraint of blocking is eliminated since an occupied machine can be simultaneously loaded or unloaded. In these studies the cell was made up of 2 machines and an additive travel time metric when the parts produced were identical and 1-unit cycle was considered in [35] while a  $k$ -unit cycle was studied in [34]. The layout considered was inline in [35] and both circular and inline in [34] and the pickup criteria studied were free and no wait.

[34] concentrate on a class of pure cycles which are newly introduced in which less than  $m$  parts are processed in a cycle compared to previous studies were pure cycles were  $m$  unit cycles meaning  $m$  parts are produced in a cycle. [35] studies robotic cells which are reentrant in which the centered robot can swap and part types processed are identical. In the beginning, the optimality regions are determined when

a part enters the first machine for the second time. The same is done when the part enters the second machine for the second time. And optimality regions are determined when a part enters the two machines for the second time. They then perform a sensitivity analysis for the parameters related to cycle time objective function and results indicate that gain in productivity is realized in a swap able robot.

## **2.6 Robots with Output Buffer at Each Machine**

In all the previous studies that were discussed the cell had only input and output buffers at the beginning and end of the robotic production line. However, [19] and [27] proposed having an **output buffer for each machine** in the line and discussed how this would affect reduction in cycle time. Both studies considered an m machine case with an additive travel time metric and a pickup criterion that is free. However, [19] considered a dual gripper robot in a circular layout robotic cell when parts produced are multiple and the strategy of production is MPS cycles. On the other hand, [27] considered a single gripper robot in an inline layout robotic cell when parts produced are identical and 1 unit is produced in a cycle.

Practically it has been studied that the advantage of a dual gripper is that there is increase in cell productivity compared to a single gripper. [19] provided an extended insight and conceptual framework to the scheduling problem with a dual gripper robot. For the robotic cell with a dual gripper in which the production is cyclic, a modelling framework is provided. Active cycles were the so-called cycles they focused on and the feasibility and combinatorial issues of the problem were studied. Complete family of active cycles are described by an algorithm approach that was provided. Moreover, in an m machine case with gripper switching time that is small, a polynomial time algorithm was devised for a 1-unit cycle optimal solution to be

found. With presence of an output buffer at each machine with capacity of 1 unit, a comparative study was employed between single and dual gripper. Results showed that more productivity was realized from dual grippers when compared to single grippers. Two models that are different are considered in [27]. The first model is a robotic cell with a single gripper robot and at each machine there is an output buffer with unit capacity. The second model is a robotic cell with a dual gripper robot and is bufferless. Concentrated Robot Move sequence (CRM) cycles are the focus of this study. Under common conditions in practice, the equivalence in the throughput of these two models is this paper's main outcome. Discussions indicated that the model with an output buffer had total cost that was 20% less than the model with a dual gripper. And this result argues the fact that there is equivalence between the two models.

## **2.8 Robots with Input and Output Buffer at Each Machine**

In the previous two studies only an output buffer at each machine is considered. However, [30] and [32] proposed having an **input and output buffer for each machine** in the production line and discussed how this would affect reduction in cycle time. Both studies considered an  $m$  machine case in a circular layout robotic cell with an additive travel time metric, a pickup criterion that is free and parts produced are identical. However, [30] considered a single gripper robot and 1 unit is produced in a cycle. On the other hand, [32] considered a dual gripper robot and  $k$  unit is produced in a cycle.

Literature has extensively studied robotic cells that are bufferless. Few studies have considered each machine with an output buffer and their results showed that such a configuration can improve the throughput. [30] considered a robotic cell where each

machine has an output and input buffer with one unit capacity and their results showed that there was no throughput improved when compared to the output buffer model. [32] is the first that considers scheduling of a robotic cell with input and buffer at each machine and unit capacity and a dual gripper robot. An optimal throughput upper bound that is tight is first obtained and an asymptotically optimal sequence is then obtained using this bound under common condition in practice. Then, the realized productivity improvement was quantified when using input and output buffers with unit capacity at each machine. Production managers can use these results to measure gain in productivity when installation of unit capacity buffers at each stage of processing of a cluster tool with a dual gripper is conducted.

## **2.9 Robot with Buffer Capacity**

The only study that considered a robot with a buffer capacity that is infinite is [43] and the robotic cell studied in the article was a 2 machine cell with an inline layout, single and dual gripper robot and parts produced are identical and 1 unit produced in each cycle. The travel time metric was additive and the pickup criterion was interval. They considered the single gripper robot with buffer capacity and compared it to a dual gripper robot with no buffer capacity to see whether it further improves the cycle time over that for classical robotic cells where robots have no buffer space. They derived the dominant cycles for both the cases and the results indicate that self-buffered robot leads to reduction in cycle time and performs more efficiently when compared to dual gripper robots and robots with swap ability.

The main contribution in this study is that  $m$  machines were considered rather than 2 and an MIP model was created and the results were compared to that of the simulated annealing algorithm rather than deriving the dominant cycles.

## Chapter 3

### PROBLEM DEFINITION

The cyclic scheduling problem considered is defined by many notations. Other than the commonly known notations such as process time, number of machines, etc., four activities that are repeated  $m$  times in every cycle by the robot such as loading, unloading and moving the parts within the robotic cell are considered. The problem is to determine the order of these activities that are performed by the robot  $4m$  times in total with the objective of minimizing cycle time for the FMC system described. These set of activities are separately defined below.

#### 3.1 Notations and Definitions

**$m$** : number of machines that makeup the robotic cell considered.

**$p$** : part process time by a machine.

**$K$** : robot buffer capacity which is the number of parts that can be held by the robot.

**$I_i$** : activity which involves taking the  $i^{th}$  unfinished part from the input buffer after moving to it where  $I = \{I_1, I_2, \dots, I_m\}$

**$L_i$** : activity which involves loading an unfinished part to the  $i^{th}$  machine after moving to it where  $L = \{L_1, L_2, \dots, L_m\}$

**$U_i$** : activity which involves unloading a finished part from  $i^{th}$  machine after moving to it where  $U = \{U_1, U_2, \dots, U_m\}$

**$O_i$** : activity which involves putting  $i^{th}$  finished part to the output buffer after moving to it where  $O = \{O_1, O_2, \dots, O_m\}$

**$A$** : union of the activities  $I, L, U, O$

When activity  $I_i$  is completed the robot stays at the input buffer and similarly it stays at the output buffer when activity  $O_i$  is completed. When activity  $U_i$  or  $L_i$  is completed the robot stays at the  $i^{th}$  machine. It must be noted that  $i$  refers to the machine number when it is considered for activities  $L$  and  $U$ . However,  $i$  refers to the  $i^{th}$  finished or unfinished part when it is considered from activities  $O$  and  $I$ .

During each cycle these set of activities are carried out by the robot and the same set of operations are repeated in each cycle. The time taken for a cycle to be completed also known as cycle time is the duration spanning from starting the first activity and completing all the other activities and then at the end coming back to the same activity we started with. A setup is needed in the beginning of each cycle which can either mean all machines are emptied or loaded. The machine is only loaded and unloaded once per cycle. The cycle time depends on the activities of the robot that include traveling from one machine to another and loading and unloading. Thus, for the cycle time to be calculated we need to consider travel time, loading and unloading time between any two activities and a distance matrix is formulated for this reason.

$\epsilon$ : time spent picking up/putting a part from/to a machine or input/output buffer.

$\delta$ : time taken by the robot to travel between two successive machines or between input/output buffer and a machine.

$d_{ab}$ : time that is required for the operation executed by the robot between for completing activity  $b$  after  $a$ .



It must be noted that activity  $a$  must be followed by activity  $b$  and machine process times are not included in the  $d_{ab}$  formula because it is not an activity carried out by the robot.

### 3.2 Distance Matrix Derivation

Below the different cases will be discussed to show how the distance matrix was found along with an example for each case.

**Case 1:**  $a \in \{L_i, U_i\}$  and  $b \in \{L_j, U_j\}$

This case is divided into 4 subdivisions where  $i \neq j$ :

1. activity  $a$  is loading machine  $i$  ( $L_i$ ) and activity  $b$  is loading machine  $j$  ( $L_j$ )
2. activity  $a$  is loading machine  $i$  ( $L_i$ ) and activity  $b$  is unloading machine  $j$  ( $U_j$ )
3. activity  $a$  is unloading machine  $i$  ( $U_i$ ) and activity  $b$  is loading machine  $j$  ( $L_j$ )
4. activity  $a$  is unloading machine  $i$  ( $U_i$ ) and activity  $b$  is unloading machine  $j$  ( $U_j$ )

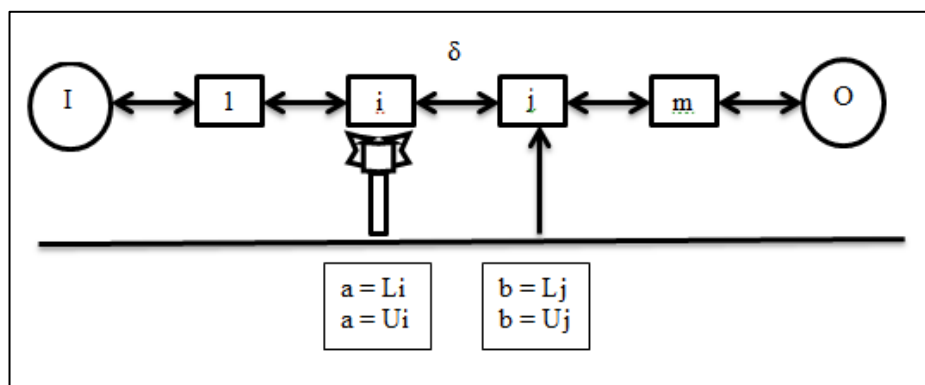


Figure 3.1: Case when Activity  $a \in \{L_i, U_i\}$  and Activity  $b \in \{L_j, U_j\}$

So, at the completion of  $L_i$  activity the robot will move to machine  $j$  at travel time  $\delta$  and the distance between those two machines is  $i - j$ . Since machine  $i$  might not

always be before machine  $j$ , the absolute of the distance must be taken as  $|i - j|$  and then once the robot reaches machine  $j$  it will execute the operation of putting the unfinished part at the end of the loading activity with time  $\varepsilon$ . So,  $d_{ab} = \varepsilon + |i - j|\delta$ .

It must be noted that  $\varepsilon$  is only considered for the second activity because  $d_{ab}$  is the time between completion time of  $a$  and completion time of  $b$ . The completion time of  $a$  indicates that the activity has already been executed at the beginning of considering this formulation.

By considering an example of this case it might be clearer. For activity  $a$  being  $L_1$  that means it will load machine 1 with an unfinished part and activity  $b$  will be  $L_2$  which means it will load machine 2 with an unfinished part.

**Example of Case 1:  $a \in \{L_1\}$  and  $b \in \{L_2\}$**

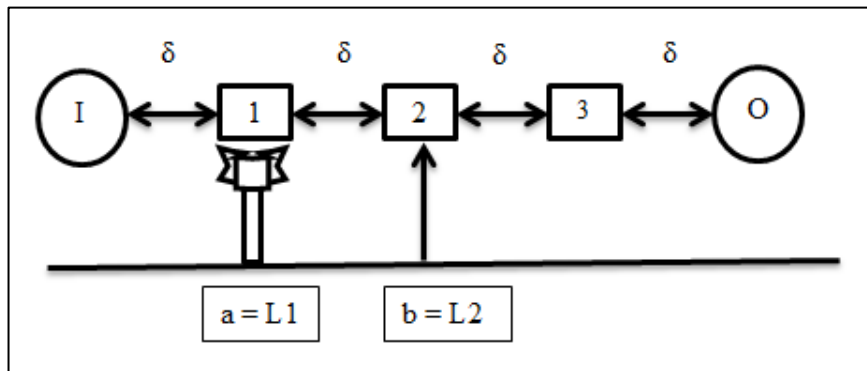


Figure 3.2: Example when Activity  $a \in \{L_1\}$  and Activity  $b \in \{L_2\}$

$$d_{L_1L_2} = |1 - 2|\delta + \varepsilon = \varepsilon + \delta$$

**Case 2:** ( $a \in \{L_i, U_i\}$  and  $b \in \{I\}$ ) or ( $a \in \{I\}$  and  $b \in \{L_i, U_i\}$ )

This case is divided into 4 subdivisions where  $i \neq j$ :

1. activity  $a$  is loading machine  $i$  ( $L_i$ ) and activity  $b$  is unloading input buffer ( $I$ )
2. activity  $a$  is unloading machine  $i$  ( $U_i$ ) and activity  $b$  is unloading input buffer ( $I$ )
3. activity  $a$  is unloading input buffer ( $I$ ) and activity  $b$  is loading machine  $i$  ( $L_i$ )
4. activity  $a$  is unloading input buffer ( $I$ ) and activity  $b$  is unloading machine  $i$  ( $U_i$ )

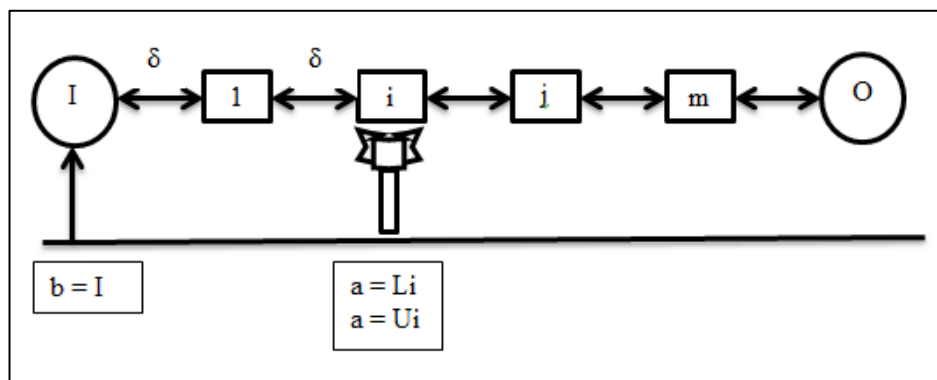


Figure 3.3 a: Case when Activity  $a \in \{L_i, U_i\}$  and Activity  $b \in \{I\}$

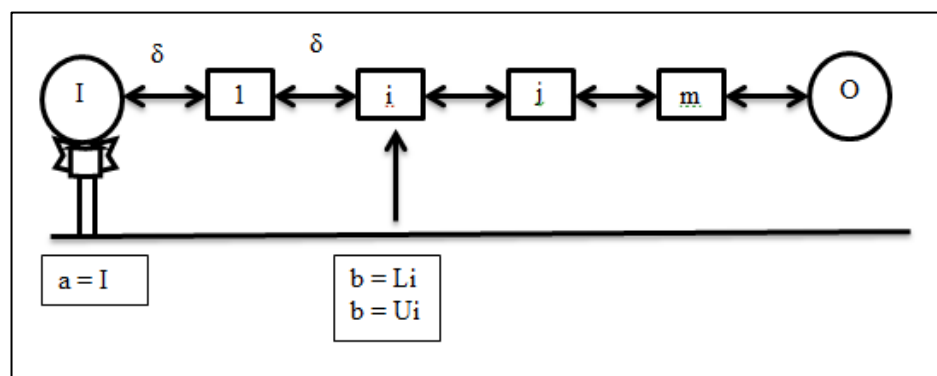


Figure 3.3 b: Case when Activity  $a \in \{I\}$  and Activity  $b \in \{L_i, U_i\}$

So at the completion of  $I$  activity the robot will move to machine 1 at travel time  $\delta$  and then from machine 1 to machine  $i$  at travel time  $\delta$  with the distance between those two machines being  $i - 1$  and then once the robot reaches machine  $i$  it will execute the operation of putting the unfinished part which takes time  $\varepsilon$  at the end of the loading activity. So,  $d_{ab} = \varepsilon + \delta + (i - 1)\delta = \varepsilon + i\delta$ .

For example, if activity  $a$  is  $L_2$  that means it will load machine 2 with an unfinished part and activity  $b$  will be  $I$  which means it will pick up an unfinished part from the input buffer.

**Example of Case 2:**  $a \in \{L_2\}$  and  $b \in \{I\}$

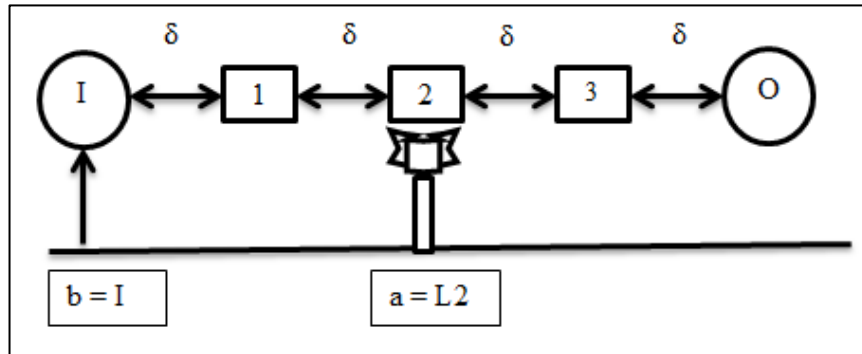


Figure 3.4: Example when Activity  $a \in \{L_2\}$  and Activity  $b \in \{I\}$

$$d_{L_2I} = (2-1)\delta + \delta + \varepsilon = \varepsilon + 2\delta$$

**Case 3:**  $(a \in \{L_i, U_i\}$  and  $b \in \{O\})$  or  $(a \in \{O\}$  and  $b \in \{L_i, U_i\})$

This case is divided into 4 subdivisions where  $i \neq j$ :

1. activity  $a$  is loading machine  $i$  ( $L_i$ ) and activity  $b$  is loading output buffer ( $O$ )

2. activity  $a$  is unloading machine  $i$  ( $U_i$ ) and activity  $b$  is loading output buffer ( $O$ )
3. activity  $a$  is loading output buffer ( $O$ ) and activity  $b$  is loading machine  $i$  ( $L_i$ )
4. activity  $a$  is loading output buffer ( $O$ ) and activity  $b$  is unloading machine  $i$  ( $U_i$ )

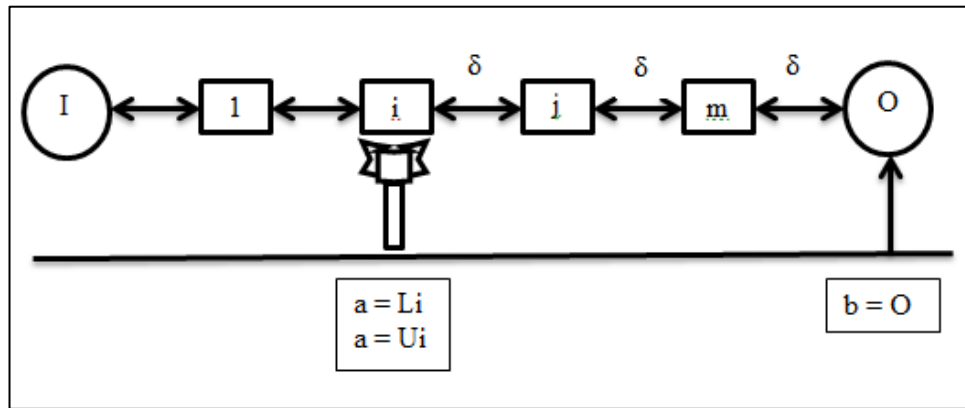


Figure 3.5 a: Case when Activity  $a \in \{L_i, U_i\}$  and Activity  $b \in \{O\}$

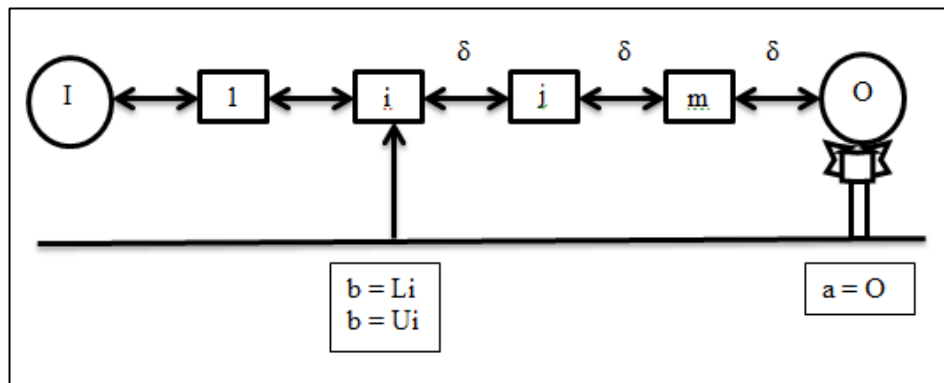


Figure 3.5 b: Case when Activity  $a \in \{O\}$  and Activity  $b \in \{L_i, U_i\}$

So at the completion of  $L_i$  activity the robot will move from machine  $i$  to machine  $m$  at travel time  $\delta$  with the distance between those two machines being  $m - i$  and then from machine  $m$  to the output buffer with travel time  $\delta$  and then once the robot reaches output buffer it will execute the operation of putting the finished part which

will take time  $\varepsilon$  at the end of the loading activity. So,  $d_{ab} = \varepsilon + \delta + (m - i)\delta = \varepsilon + (m - i + 1)\delta$ .

For example, if activity  $a$  is  $L_2$  that means it will load machine 2 with an unfinished part and activity  $b$  will be  $O$  which means it will put a finished part to the output buffer.

**Example of Case 3:**  $a \in \{L_2\}$  and  $b \in \{O\}$

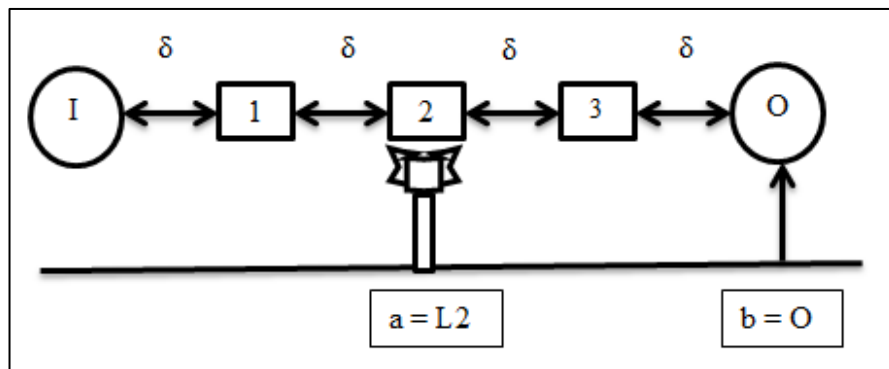


Figure 3.6: Example when Activity  $a \in \{L_i, U_i\}$  and Activity  $b \in \{O\}$

$$d_{L_2O} = (3-2)\delta + \delta + \varepsilon = \varepsilon + 2\delta$$

**Case 4:**  $(a \in \{I\} \text{ and } b \in \{O\})$  or  $(a \in \{O\} \text{ and } b \in \{I\})$

This case is divided into 2 subdivisions where:

1. activity  $a$  is unloading input buffer ( $I$ ) and activity  $b$  is loading output buffer ( $O$ )
2. activity  $a$  is loading output buffer ( $O$ ) and activity  $b$  is unloading input buffer ( $I$ )

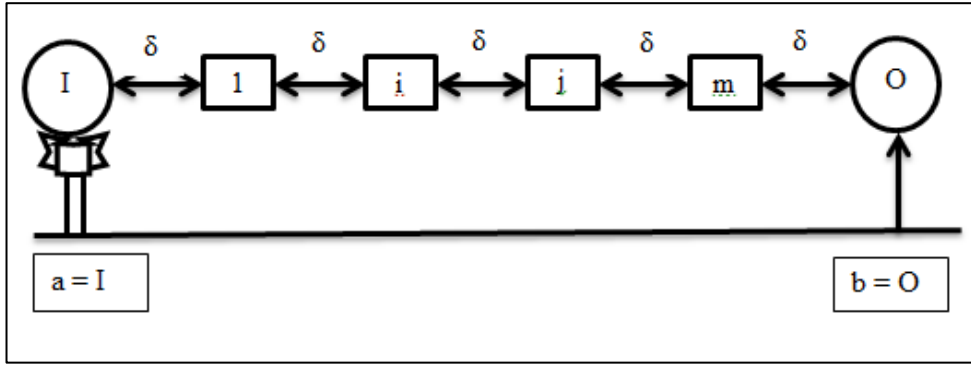


Figure 3.7 a: Case when Activity  $a \in \{I\}$  and Activity  $b \in \{O\}$

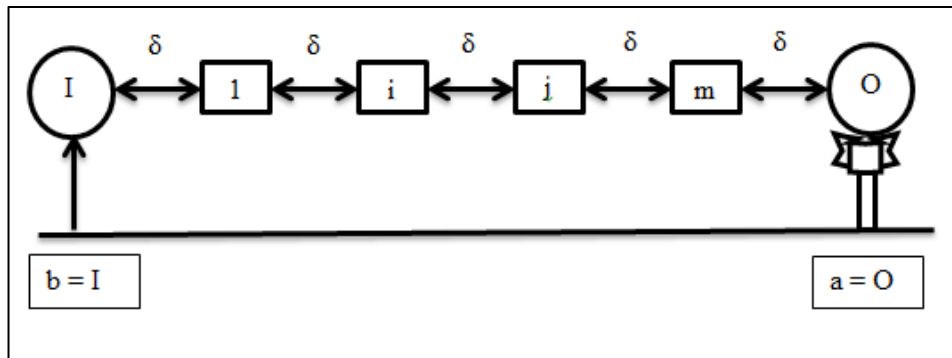


Figure 3.7 b: Case when Activity  $a \in \{O\}$  and Activity  $b \in \{I\}$

So at the completion of  $I$  activity the robot will move to machine 1 at travel time  $\delta$  and from machine 1 to machine  $m$  with the distance between those two machines being  $m - 1$  and then from machine  $m$  to the output buffer with travel time  $\delta$  and then once the robot reaches output buffer it will execute the operation of putting the finished part with time  $\varepsilon$  at the end of the activity. So,  $d_{ab} = \varepsilon + \delta + (m - 1)\delta + \delta = \varepsilon + (m + 1)\delta$ .

For example, if activity  $a$  is  $I$  that means it will pick up an unfinished part from input buffer and activity  $b$  will be  $O$  which means it will put a finished part to the output buffer.

**Example of Case 4:**  $a \in \{I\}$  and  $b \in \{O\}$

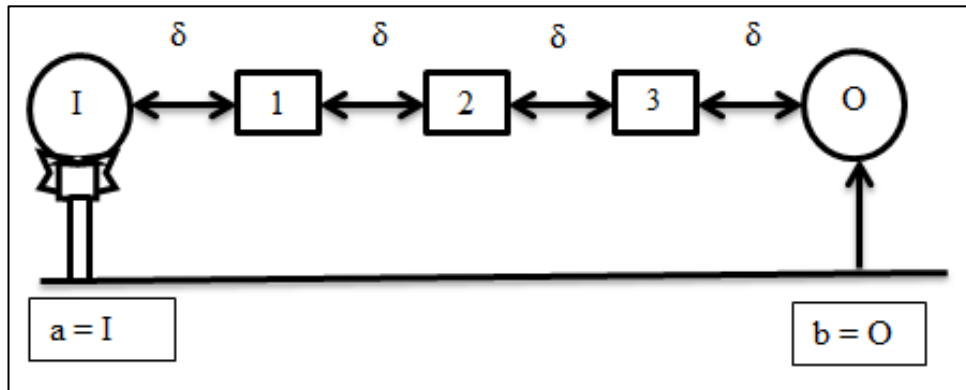


Figure 3.8: Example when Activity  $a \in \{I\}$  and Activity  $b \in \{O\}$

$$d_{IO} = \delta + (3 - 1)\delta + \delta + \varepsilon = \varepsilon + 4\delta$$

**Case 5:**  $(a, b \in \{I\})$  or  $(a, b \in \{O\})$

This case is divided into 2 subdivisions where:

1. activity  $a$  is unloading input buffer ( $I$ ) and activity  $b$  is unloading input buffer ( $I$ )
2. activity  $a$  is loading output buffer ( $O$ ) and activity  $b$  is loading output buffer ( $O$ )

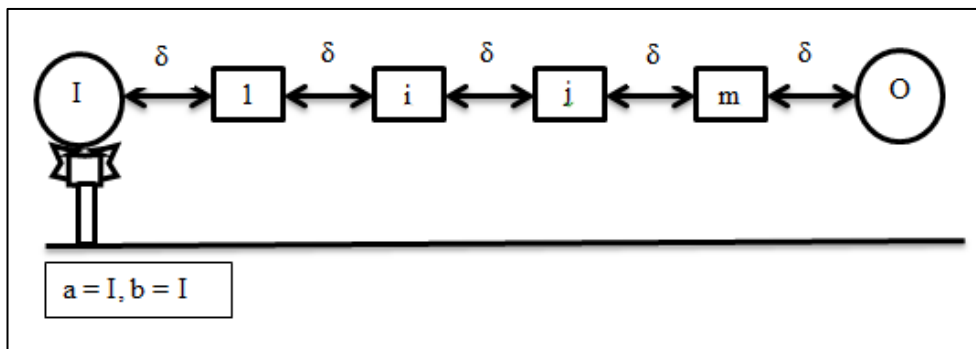


Figure 3.9: Case when Activity  $a, b \in \{I\}$



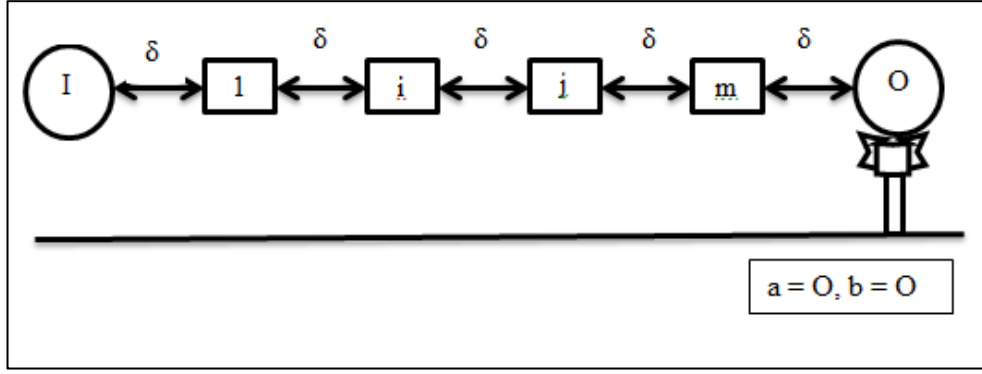


Figure 3.10: Case when Activity  $a, b \in \{O\}$

So at the completion of  $I$  activity the robot will not move since the second activity is also picking up an unfinished part from the input buffer so it will execute the operation of picking up the part and the time taken for this operation at the end of this activity is  $\varepsilon$ . And the same is true if the completion of  $O$  activity is followed by another activity of putting a finished part on the output buffer. So,  $d_{ab} = \varepsilon$ .

Thus, the distance matrix for all cases is summarized as follows:

$$d_{ab} = \begin{cases} \varepsilon + |i - j|\delta & \text{if } a \in \{L_i, U_i\} \text{ and } b \in \{L_j, U_j\} \\ \varepsilon + i\delta & \text{if } (a \in \{L_i, U_i\} \text{ and } b \in I) \text{ or } (a \in I \text{ and } b \in \{L_i, U_i\}) \\ \varepsilon + (m - i + 1)\delta & \text{if } (a \in \{L_i, U_i\} \text{ and } b \in O) \text{ or } (a \in O \text{ and } b \in \{L_i, U_i\}) \\ \varepsilon + (m + 1)\delta & \text{if } (a \in I \text{ and } b \in O) \text{ or } (a \in O \text{ and } b \in I) \\ \varepsilon & \text{if } (a, b \in I) \text{ or } (a, b \in O) \end{cases} \quad (3.1)$$

### 3.3 Process Time

The distance matrix does not contain process time since processing of a part is a function of the machine and is not involved with any kind of operation executed by the robot. However, in cases when process time is large, at some point in the cycle this may lead to the robot waiting for some amount of time.

So when the first activity is loading machine  $i$  ( $L_i$ ) with an unfinished part it will be processed for  $p$  time units, the robot will then travel between machines to carry out

other sets of activities while processing on machine  $i$  continues until it is time to unload machine  $i$  ( $U_i$ ) with the finished part and the operation of unloading will require unloading time of  $\varepsilon$  time units. Thus, regardless of the travelling time, the time between activities  $L_i$  and  $U_i$  is at least  $\varepsilon + p$  since  $U_i$  cannot be completed unless the processing of the part is completed. So if the activities between  $L_i$  and  $U_i$  take less than  $p$  time units, the robot will have to wait when it comes back to machine  $i$  for unloading. The order in which the activities are executed have an impact on whether there will be waiting time or not. It must be noted that this uncertain waiting time amount is not considered in the dab formula.

Since in every cycle the same set of activities will be repeated in order for a cycle to be fixed and for permutations to be avoided the first activity will always be fixed to loading machine 1 ( $L_1$ ) and thus the cycle ends when it comes back to the activity again and this duration will be the cycle time and the objective will be for the cycle time to be minimized by scheduling the order of robot activities.

## Chapter 4

### METHODOLOGY

Methods used to solve optimization problems are usually classified into two types, either exact solution methods that provide optimal solutions that are guaranteed or heuristic solution methods that do not guarantee optimality. They are also further classified as either constructive methods which means that we start from no solution and build up to a feasible or ultimately the optimal solution or improvement methods which means we initially start from a feasible solution and build up to a better solution. Examples of exact solution methods include: Branch and Bound algorithm, MIP (Mixed Integer Programming), IP (Integer Programming), LP (Linear Programming) or NMIP (Non-Linear Mixed Integer Programming) models, Polynomial algorithm, etc. On the other hand examples of heuristic or meta-heuristic approaches include: simulated annealing, genetic algorithm, tabu search, etc.

#### **4.1 Mixed Integer Programming Model**

In this study the optimization problem for scheduling the robot moves in order to minimize cycle time is modeled as a MIP Model which indicates that some of the decision variables are integer while others are non-integer. An MIP model is an exact approach that is discussed in detail below. The decision variables of the problem are as follows:

**Decision Variables:**

$$x_{ab} = \begin{cases} 1, & \text{if the robot performs activity a before activity b} \\ 0, & \text{otherwise} \end{cases}$$

$t_a$ : activity a completion time

$C$ : cycle time

$$z_i = \begin{cases} 1, & \text{if activity } U_i \text{ is performed after activity } L_i \\ 0, & \text{otherwise} \end{cases}$$

$Y_a^-$ : the number of parts on the robot that are unfinished at the end of activity a

$Y_a^+$ : the number of parts on the robot that are finished at the end of activity a

$M$ : is a big number that is defined

Definition of the notations  $m, p, K, I, L, U, O, A, \varepsilon$  and  $d_{ab}$  can be referred from Chapter 3 on pages 23 and 24.

$$\text{Min } C \tag{4.1}$$

s. t

$$\sum_{a \in A-b} x_{ab} = 1 \quad \forall b \in A \tag{4.2}$$

$$\sum_{b \in A-a} x_{ab} = 1 \quad \forall a \in A \tag{4.3}$$

$$t_b \geq t_a + d_{ab} - M(1 - x_{ab}) \quad \forall a \neq b \in A, b \neq L_1 \tag{4.4}$$

$$t_{U_i} - t_{L_i} \leq Mz_i \quad i = 1, \dots, m \tag{4.5}$$

$$t_{U_i} \geq t_{L_i} + (\varepsilon + p) - M(1 - z_i) \quad i = 1, \dots, m \tag{4.6}$$

$$t_{L_i} \leq t_{U_i} + C + (\varepsilon + p)(1 - z_i) \quad i = 1, \dots, m \tag{4.7}$$

$$C \geq t_a + d_{aL_1}x_{aL_1} \quad \forall a \in A - L_1 \tag{4.8}$$

$$x_{ab} \in \{0, 1\} \quad \forall a \neq b \in A \quad (4.9)$$

$$t_a \geq 0 \quad \forall a \in A \quad (4.10)$$

$$C \geq 0 \quad (4.11)$$

$$z_i \in \{0, 1\} \quad i = 1, \dots, m \quad (4.12)$$

$$Y_b^+ \geq Y_a^+ + 1 - (K + 1)(1 - x_{ab}) \quad \forall a \in A, b \in U \quad (4.13)$$

$$Y_b^+ \leq Y_a^+ + 1 + (K + 1)(1 - x_{ab}) \quad \forall a \in A, b \in U \quad (4.14)$$

$$Y_b^+ \geq Y_a^+ - 1 - (K + 1)(1 - x_{ab}) \quad \forall a \in A, b \in O \quad (4.15)$$

$$Y_b^+ \leq Y_a^+ - 1 + (K + 1)(1 - x_{ab}) \quad \forall a \in A, b \in O \quad (4.16)$$

$$Y_b^+ \geq Y_a^+ - (K + 1)(1 - x_{ab}) \quad \forall a \in A, b \in I \quad (4.17)$$

$$Y_b^+ \leq Y_a^+ + (K + 1)(1 - x_{ab}) \quad \forall a \in A, b \in I \quad (4.18)$$

$$Y_b^+ \geq Y_a^+ - (K + 1)(1 - x_{ab}) \quad \forall a \in A, b \in L \quad (4.19)$$

$$Y_b^+ \leq Y_a^+ + (K + 1)(1 - x_{ab}) \quad \forall a \in A, b \in L \quad (4.20)$$

$$Y_b^- \geq Y_a^- + 1 - (K + 1)(1 - x_{ab}) \quad \forall a \in A, b \in I \quad (4.21)$$

$$Y_b^- \leq Y_a^- + 1 + (K + 1)(1 - x_{ab}) \quad \forall a \in A, b \in I \quad (4.22)$$

$$Y_b^- \geq Y_a^- - 1 - (K + 1)(1 - x_{ab}) \quad \forall a \in A, b \in L \quad (4.23)$$

$$Y_b^- \leq Y_a^- - 1 + (K + 1)(1 - x_{ab}) \quad \forall a \in A, b \in L \quad (4.24)$$

$$Y_b^- \geq Y_a^- - (K + 1)(1 - x_{ab}) \quad \forall a \in A, b \in U \quad (4.25)$$

$$Y_b^- \leq Y_a^- + (K + 1)(1 - x_{ab}) \quad \forall a \in A, b \in U \quad (4.26)$$

$$Y_b^- \geq Y_a^- - (K + 1)(1 - x_{ab}) \quad \forall a \in A, b \in O \quad (4.27)$$

$$Y_b^- \leq Y_a^- + (K + 1)(1 - x_{ab}) \quad \forall a \in A, b \in O \quad (4.28)$$

$$Y_a^+ + Y_a^- \leq K \quad \forall a \in A \quad (4.29)$$

$$Y_a^+ \geq 0 \quad \forall a \in A \quad (4.30)$$

$$Y_a^- \geq 0 \quad \forall a \in A \quad (4.31)$$

(4.1) is the objective function which is cycle time minimization where cycle time is the amount of time that spans when a system starts at a specific state and comes back to the same state again.

(4.2) and (4.3) are like the constraints of the assignment problem which indicates that in (4.2) if  $a$  is  $I, O, U$  or  $L$  and we assume that only  $I$  is the activity that is active then only it should pass to another activity. There cannot be two activities passing to another activity at the same time and this constraint holds for all  $a$  except when  $a = b$ . Similarly in (4.3) for all  $b$  equal to either  $I, O, U, L$  if we assume that  $I$  is the activity that is performed after the activity that is active then it should be performed after only one of the activities and this constraint holds for all  $b$  except when  $a = b$ . For example if it is a 1 machine case, then there will be only 4 activities. If it is shown as an assignment problem then it will look like this:

Table 4.1: Representation of Constraint (4.2) and (4.3)

<b>a \ b</b>	<b>L<sub>1</sub></b>	<b>U<sub>1</sub></b>	<b>I<sub>1</sub></b>	<b>O<sub>1</sub></b>
<b>L<sub>1</sub></b>		$X_{U_1L_1}$	$X_{I_1L_1}$	$X_{O_1L_1}$
<b>U<sub>1</sub></b>	$X_{L_1U_1}$		$X_{I_1U_1}$	$X_{O_1U_1}$
<b>I<sub>1</sub></b>	$X_{L_1I_1}$	$X_{U_1I_1}$		$X_{I_1O_1}$
<b>O<sub>1</sub></b>	$X_{L_1O_1}$	$X_{U_1O_1}$	$X_{I_1O_1}$	

So, if  $x_{OL}$ ,  $x_{LU}$ ,  $x_{UI}$  and  $x_{IO}$  were equal to 1 then the order of the cycle would be  $O$ ,  $L$ ,  $U$ ,  $I$  and then again back to  $O$ . These two constraints guarantee that all activities have to be performed only once for each machine.

In (4.4) if  $x_{ab} = 1$  it means activity  $a$  is performed before  $b$  and if that is the case then the completion time of activity  $b$  will be equal or greater than completion time of activity  $a$  plus the distance matrix between activity  $a$  and  $b$  identified by  $d_{ab}$  thus:  $t_b \geq t_a + d_{ab}$ . However, if  $x_{ab} = 0$  indicating that activity  $a$  is not performed before  $b$  then  $t_b \geq t_a + d_{ab} - M$  and since  $M$  is a very big number that means  $t_b - t_a - d_{ab} \geq -M$  which is infeasible and thus this constraint ensures or guarantees feasibility of successive activities because when  $x_{ab} = 0$  there must be no relation between activity  $a$  and  $b$ . For Example, if  $a = L_1$ ,  $b = L_2$ ,  $\delta = 2$ ,  $\varepsilon = 1$ . From Case 1 of our distance matrix it was seen that for  $a \in \{L_i\}$  and  $b \in \{L_j\}$ ,  $d_{L_iL_j} = |i - j|\delta + \varepsilon$  and thus  $d_{L_1L_2} = |1 - 2|\delta + \varepsilon = \delta + \varepsilon = 2 + 1 = 3$  and let us assume  $t_a$  was 3. Since  $x_{L_1L_2}$  is 1 in this case because activity  $b$  is performed after  $a$ ,  $t_b = 3 + 3 = 6$ .

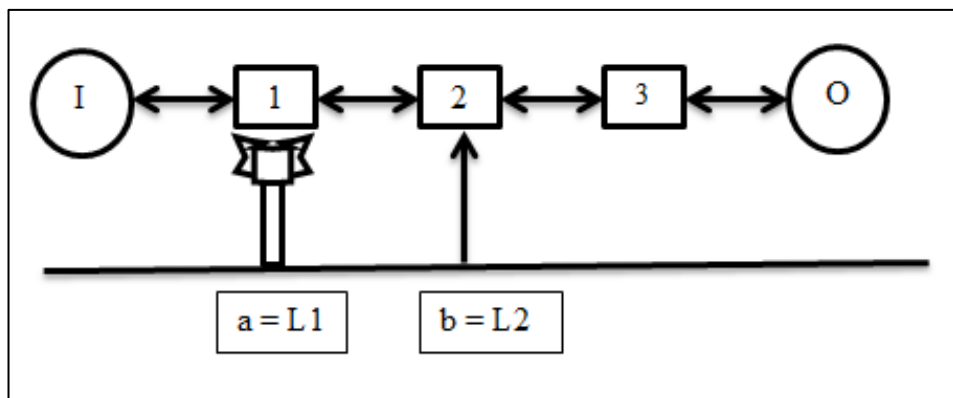


Figure 4.1: Representation of Constraint (4.4) by an Example of Case 1

Also, it should be noted that activity  $b$  cannot be  $L_1$  because in the problem definition chapter we discussed that  $L_1$  will always be fixed as first activity to avoid permutation.

In (4.5) activity of unloading a finished part from machine  $i$  must be performed after the loading of machine  $i$  with an unfinished part. Otherwise the constraint must be deemed infeasible. Thus, if  $z_i$  is 1 meaning activity  $L_i$  is before  $U_i$  then  $t_{U_i} - t_{L_i} \leq M$  which means  $t_{U_i} \leq t_{L_i} + M$ , since  $M$  is a very big number it basically means the completion time of  $U_i$  must be greater than completion time of  $L_i$ . However if  $z_i = 0$  indicating that activity  $U_i$  is not performed after  $L_i$  then  $t_{U_i} - t_{L_i} \leq 0$  which means  $t_{U_i} \leq t_{L_i}$  which is infeasible since the completion time of the  $U_i$  cannot be less than that of  $L_i$ . This constraint guarantees whether constraint (4.6) or (4.7) is active because only one of them will be active at the same time.

If  $z_i = 1$  meaning activity  $L_i$  is before  $U_i$  then (4.6) is active and that means  $t_{U_i} \geq t_{L_i} + (\varepsilon + p)$  which means completion time of  $U_i$  is atleast completion time of  $L_i$  plus the processing time of the part and the time taken to pick up the part from the machine. This means that after activity  $L_i$  is completed if the activity following it is  $U_i$  then assuming process time is 0 the least amount of completion time of activity  $U_i$  is the time taken to pick up the part because other times can include travel time and picking up/putting time for all activities between  $L_i$  and  $U_i$ . However if process time is too large and by the time the robot comes back to machine  $i$  and processing of the part was not completed, the robot will have to wait for a maximum amount of the process time itself. For Example, if  $a = L_1$ ,  $b = U_1$ ,  $\delta = 2$ ,  $\varepsilon = 1$ ,  $p = 100$  and let



us assume  $t_a$  was 20. Since  $U_i$  is 1 in this case because activity  $U_i$  is performed after  $L_i$ ,  $t_{U_i} \geq 20 + 1 + 100 = 121$ .

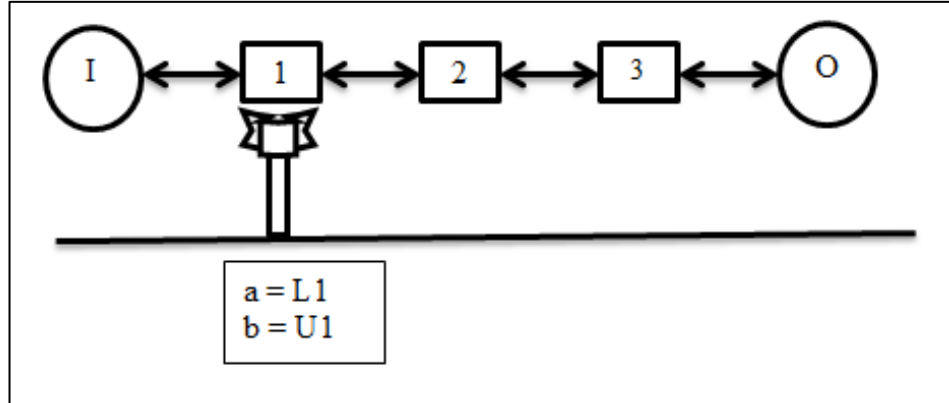


Figure 4.2: Representation of Constraint (4.6) by an Example of Case 1

However in (4.7) if  $z_i = 0$  then activity  $U_i$  is performed before  $L_i$  and hence, we need to guarantee feasibility.  $t_{L_i} \leq t_{U_i} + C + (\varepsilon + p)$  which becomes  $t_{U_i} \geq t_{L_i} - (\varepsilon + p) - C$  which means that the time span between the cycle time and completion time of activity  $L_i$  plus the processing time of a part and the time taken to pick up the part from the machine must be less than or equal to completion time of activity  $U_i$ .

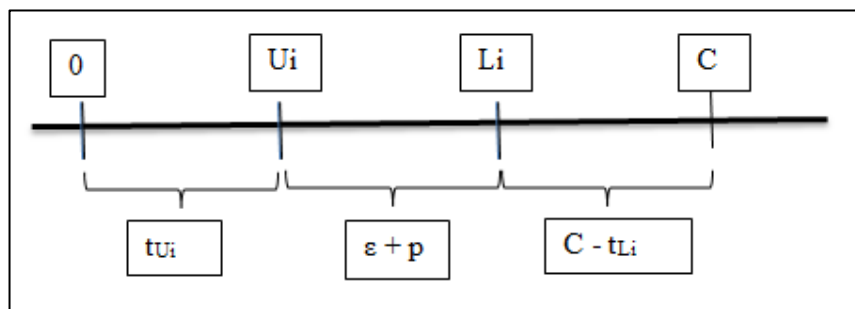


Figure 4.3: Representation of Constraint (4.7)

In (4.8) if activity  $a$  is the last activity before the cycle is repeated from  $L_1$  again and  $a$  is before  $b$  where  $b$  is the activity of loading machine 1 again meaning  $x_{aL_1} = 1$  then  $C \geq t_a + d_{aL_1}$  which means the cycle time is equal to completion of last

activity  $a$  plus the distance matrix from  $a$  back to first activity  $L_1$ . However, if  $x_{aL_1} = 0$  meaning activity  $a$  is not before  $L_1$  then  $C \geq t_a$  thus the constraint becomes redundant.

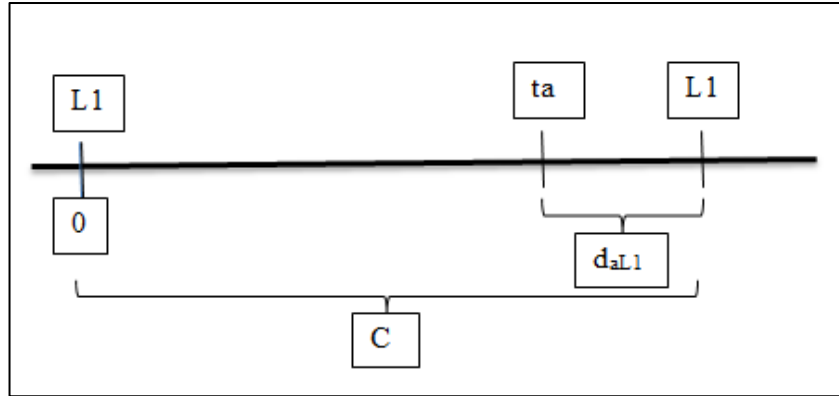


Figure 4.4: Representation of Constraint (4.8)

Since  $x_{ab}$  is either 1 or 0 it is a binary decision variable which is defined by (4.9) and the same goes for the decision variable  $z_i$  which is also a binary decision variable defined by (4.12).

The completion time of activity  $a$  ( $t_a$ ) and the cycle time are both time standards and thus they can be defined as linear decision variables that are non-negative and this is shown by (4.10) and (4.11).

Constraints (4.13) through (4.20) all deal with the number of finished parts on the robot at the end of an activity. When  $x_{ab} = 1$  it means that activity  $a$  is performed before  $b$  and thus (4.13) and (4.14) reduce to  $Y_b^+ \geq Y_a^+ + 1$  and  $Y_b^+ \leq Y_a^+ + 1$ . In optimization it is stated that a hyperplane is a set of points which satisfy one linear equation and it divides the space into half spaces determined by inequalities such as (4.13) and (4.14). Thus, the hyperplane in our case is reduced to the following one

linear equation  $Y_b^+ = Y_a^+ + 1$ . It can be seen that in (4.13) and (4.14) activity  $b$  is unloading  $a$  finished part which means that after completion of activity  $b$  one more finished part will be carried by the robot. On the contrary, when  $x_{ab} = 0$  which means activity  $a$  is not performed before  $b$  then (4.13) and (4.14) are reduced to  $Y_b^+ \geq Y_a^+ + 1 - (K + 1)$  and  $Y_b^+ \leq Y_a^+ + 1 + (K + 1)$  which is deemed infeasible since there is no intersection point between the two equations.

When  $x_{ab} = 1$  (4.15) and (4.16) reduces to  $Y_b^+ \geq Y_a^+ - 1$  and  $Y_b^+ \leq Y_a^+ - 1$ . Since it is two half spaces or two inequalities it is reduced to the following one linear equation  $Y_b^+ = Y_a^+ - 1$ . It can be seen that in (4.15) and (4.16) activity  $b$  is putting a finished part on the output buffer which means that after completion of activity  $b$  one more finished part will be removed from the robot. On the contrary, when  $x_{ab} = 0$  then (4.15) and (4.16) are reduced to  $Y_b^+ \geq Y_a^+ - 1 - (K + 1)$  and  $Y_b^+ \leq Y_a^+ - 1 + (K + 1)$  which is deemed infeasible since there is no intersection point between the two equations.

When  $x_{ab} = 1$  (4.17) and (4.18) reduces to  $Y_b^+ \geq Y_a^+$  and  $Y_b^+ \leq Y_a^+$ . Since it is two half spaces or two inequalities it is reduced to the following one linear equation  $Y_b^+ = Y_a^+$ . It can be seen that in (4.17) and (4.18) activity  $b$  is picking up an unfinished part to the input buffer which means that after completion of activity  $b$  there will be no finished parts added or removed to the robot buffer. On the contrary, when  $x_{ab} = 0$  then (4.17) and (4.18) are reduced to  $Y_b^+ \geq Y_a^+ - (K + 1)$  and  $Y_b^+ \leq Y_a^+ + (K + 1)$  which is deemed infeasible since there is no intersection point between the two equations. The same set of equations are seen in (4.19) and (4.20)

for the case when activity  $b$  is loading an unfinished part to the machine. Then again there are no finished parts added or removed to the robot buffer.

Constraints (4.21) through (4.28) all deal with the number of unfinished parts on the robot at the end of an activity. When  $x_{ab} = 1$  (4.21) and (4.22) reduces to  $Y_b^- \geq Y_a^- + 1$  and  $Y_b^- \leq Y_a^- + 1$ . Since it is two half spaces or two inequalities it is reduced to the following one linear equation  $Y_b^- = Y_a^- + 1$ . It can be seen that in (4.21) and (4.22) activity  $b$  is picking up an unfinished part from the input buffer which means that after completion of activity  $b$  one more unfinished part will be carried by the robot buffer. On the contrary, when  $x_{ab} = 0$  which means activity  $a$  is not performed before  $b$  then (4.21) and (4.22) are reduced to  $Y_b^- \geq Y_a^- + 1 - (K + 1)$  and  $Y_b^- \leq Y_a^- + 1 + (K + 1)$  which is deemed infeasible since there is no intersection point between the two equations.

When  $x_{ab} = 1$  (4.23) and (4.24) reduces to  $Y_b^- \geq Y_a^- - 1$  and  $Y_b^- \leq Y_a^- - 1$ . Since it is two half spaces or two inequalities it is reduced to the following one linear equation  $Y_b^- = Y_a^- - 1$ . It can be seen that in (4.23) and (4.24) activity  $b$  is loading an unfinished part on the machine which means that after completion of activity  $b$  one more unfinished part will be removed from the robot buffer. On the contrary, when  $x_{ab} = 0$  then (4.23) and (4.24) are reduced to  $Y_b^- \geq Y_a^- - 1 - (K + 1)$  and  $Y_b^- \leq Y_a^- - 1 + (K + 1)$  which is deemed infeasible since there is no intersection point between the two equations.

When  $x_{ab} = 1$  (4.25) and (4.26) reduces to  $Y_b^- \geq Y_a^-$  and  $Y_b^- \leq Y_a^-$ . Since it is two half spaces or two inequalities it is reduced to the following one linear equation

$Y_b^- = Y_a^-$ . It can be seen that in (4.25) and (4.26) activity  $b$  is unloading a finished part from the machine which means that after completion of activity  $b$  there will be no unfinished parts added or removed to the robot buffer. On the contrary, when  $x_{ab} = 0$  then (4.25) and (4.26) are reduced to  $Y_b^- \geq Y_a^- - (K + 1)$  and  $Y_b^- \leq Y_a^- + (K + 1)$  which is deemed infeasible since there is no intersection point between the two equations. The same set of equations are seen in (4.27) and (4.28) for the case when activity  $b$  is putting a finished part to the output buffer. Then again there are no unfinished parts added or removed to the robot buffer.

(4.29) indicates that the number of finished and unfinished parts on the robot at the end of the activity must not exceed the robot capacity. (4.30) and (4.31) are non-negativity constraints for the decision variables  $Y_a^-$  and  $Y_a^+$ . Since these variables are number of parts then they must be integer.

## **4.2 Software Used to Solve the Model**

This model can be solved by many software programs. However, the software used to solve this MIP model was a combination of Visual Studio 2017 and IBM ILOG CPLEX Optimization Studio 12.8.0. Visual Studio is used for computer program development which was founded by Microsoft as an IDE (Integrated Development Environment) and languages such as C++ are built into it. CPLEX on the other hand is a program used to solve models such as MIP, LP and so forth in order to provide an optimal solution. CPLEX concert technology is a library that has C++ language and thus the configuration between CPLEX and Visual Studio allows us to code in C++ language in Visual Studio with the possibility of getting an optimal solution due to the CPLEX configuration.

### 4.3 Cycle and Variables Representation

Since the distance matrix  $d_{ab}$  as well as variables  $x_{ab}$  are composed of two consecutive activities where  $a$  represents an activity and  $b$  represents another activity such as for example  $d_{U_1L_1}$  or  $x_{U_1L_1}$  it can be seen that  $a$  and  $b$  themselves are divided in two parts the activity itself for example unloading and on which machine the activity was carried out for example machine 1.

Thus, if we write the representation of  $x_{ab}$  as  $x[a][i][b][j]$  then  $a$  and  $b$  are an array of 4 elements, while  $i$  and  $j$  are an array having  $m$  elements. Also, since it is easier to deal with numbers rather than alphabets the activities  $L, U, I$  and  $O$  are represented as 0, 1, 2, and 3. However, when the output is printed in the solutions page any variables or activities represented as numbers are printed as their representations.

The same is true for the variables  $t_a, Y_a^+, Y_a^-, z_i$  which are represented as  $t[a][i]$ ,  $YP[a][i]$ ,  $YN[[a][i]$  and  $z[i]$ . An example of the solution page for a 2 machine case with process time 22 and  $K = 1$  is shown in Figure (4.5).

```

cycle time = 38

x_L1_U2 = 1
x_L2_U1 = 1
x_U1_O1 = 1
x_U2_O2 = 1
x_I1_L1 = 1
x_I2_L2 = 1
x_O1_I1 = 1
x_O2_I2 = 1
t_L1 = 0
t_L2 = 20
t_U1 = 23
t_U2 = 5
t_I1 = 35
t_I2 = 15
t_O1 = 28
t_O2 = 8
YP_L1 = 0
YP_L2 = 0
YP_U1 = 1
YP_U2 = 1
YP_I1 = 0
YP_I2 = 0
YP_O1 = 0
YP_O2 = 0
YN_L1 = 0
YN_L2 = 0
YN_U1 = 0
YN_U2 = 0
YN_I1 = 1
YN_I2 = 1
YN_O1 = 0
YN_O2 = 0
z1 = 1
z2 = 0

```

Figure 4.5: Solution Page for 2 Machine Case with K = 1, P = 22

The cycle on the other hand is represented by 2 arrays  $Corder[i][j]$  where  $i = 0$  represents array of activities and  $i = 1$  represents array of machine on which activity is conducted and  $j$  has  $4m$  elements since each activity is conducted once for each machine or in terms of output or input buffer for each part that is carried or dropped and there are 4 activities so if it is a 2 machine problem, there will be 8 activities in total, and if it is 3 machines there will be 12 activities in total and so forth. So generally for an  $m$  machine case there will be  $4m$  activities. For example, if

the following representation is printed as depicted in Figure (4.6) then the actual cycle that is being represented is  $L_1U_2O_2I_2L_2U_1O_1I_1$  for a 2 machine case.

	j = 0	j = 1	j = 2	j = 3	j = 4	j = 5	j = 6	j = 7
i = 0	0	1	3	2	0	1	3	2
i = 1	0	1	1	1	1	0	0	0

Figure 4.6: Representation of a Cycle for a 2 Machine Case

#### 4.4 Simulated Annealing Algorithm

The solver was used to run the MIP model for 2 machine and 3 machine cases, it could not be used to solve larger machine problems such as 4 machine and 5 machine case and that is due to the extensive solution time which is known to be one of the cons of using exact solution methods. Thus, a heuristic approach was proposed to be used for solving the 4 machine and 5 machine cases and that is the Simulated Annealing Algorithm. A heuristic approach will not guarantee the optimal solution at all times but the solution time will be reduced and this will be seen in the next chapter.

Compared to using other heuristic approaches such as genetic algorithm, gradient descent, etc. [44] stated the main advantages of using simulated annealing are:

1. Memory shortage problems are avoided because only one solution is used at a time for a run.
2. Neighboring solutions produced are feasible and a repair algorithm is not required leading to solutions that are highly diversified.



Generally the steps of the simulated annealing algorithm start with an initial solution being constructed. Then the iteration loop begins and a neighboring solution is found. If the neighboring solution is better than the current solution then it becomes the new current solution. Else, if it was worse than the bad solution might be accepted with some acceptance probability or otherwise rejected.

The Simulated Annealing algorithm coded in Visual-CPLEX solver for the model in this study followed the steps below which was an extension to the MIP model.

#### 4.4.1 Creating the Initial Current Order

```

for (i = 0; i < m; i++)
{
    Corder[0][2 * i] = 0;
    Corder[1][2 * i] = i;
    Corder[0][(2 * i) + (2 * m) - 1] = 1;
    Corder[1][(2 * i) + (2 * m) - 1] = i;
    Corder[0][(2 * i) + (2 * m)] = 3;
    Corder[1][(2 * i) + (2 * m)] = i;
}
for (i = 1; i < m; i++)
{
    Corder[0][(2 * i) - 1] = 2;
    Corder[1][(2 * i) - 1] = i;
}
Corder[0][(4 * m) - 1] = 2;
Corder[1][(4 * m) - 1] = 0;

```

Figure 4.7: Creating Initial Current Order

Figure (4.7) shows the algorithm that was coded to create a fixed initial current order of activities that make a cycle. Since we previously defined that  $i$  has  $m$  elements we fix the  $i$  in  $Corder[i][j]$  to 0 and 1 since there will always be 2 arrays: 0 for activities and 1 for machines.

The initial current order for a 2 machine case will look something like Figure (4.8) which is a cycle in the form of  $L_1I_2L_2U_1O_1U_2O_2I_1$  when the numbers that represented the activities are replaced by their actual representations and machines are replaced by  $i + 1$  is a cycle of the form as depicted in Figure (4.9).

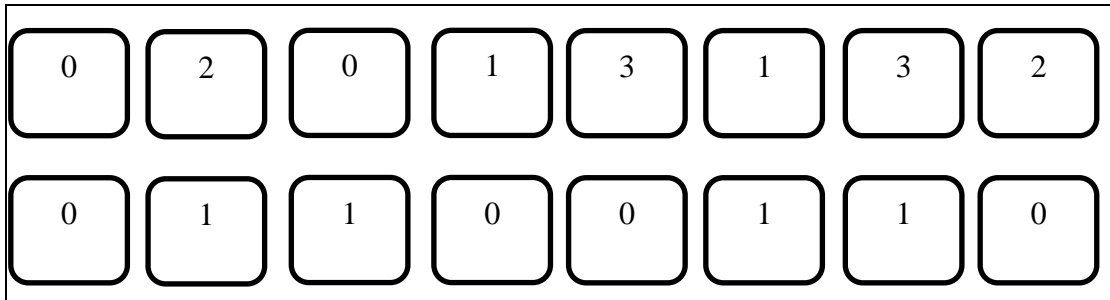


Figure 4.8: Initial Current Order Created for 2 Machine Case

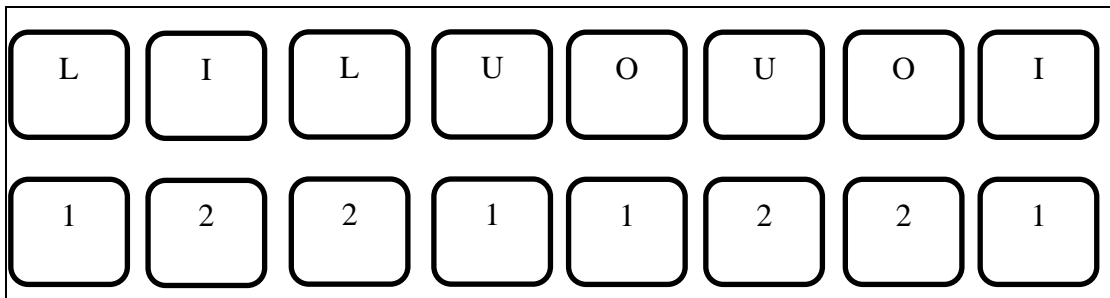


Figure 4.9: Initial Current Order for 2 Machine Case Represented by Activities and Machines

Generally, the idea of creating this initial order was because before loading a part on the machine, the robot needs to pick up a part from the input buffer and after the part is loaded it is processed and then unloaded after finishing processing and then finally the unloaded part is moved to the output buffer. This order was considered for a robot with buffer capacity 1 since an order created to consider robot buffer capacity 2 or higher will create an infeasible order for a robot with smaller buffer capacity. A generalized initial current order for  $m$  machine case is shown as Figure (4.10) and Figure (4.11) represents the order in terms of activities  $L, U, I$  and  $O$ .

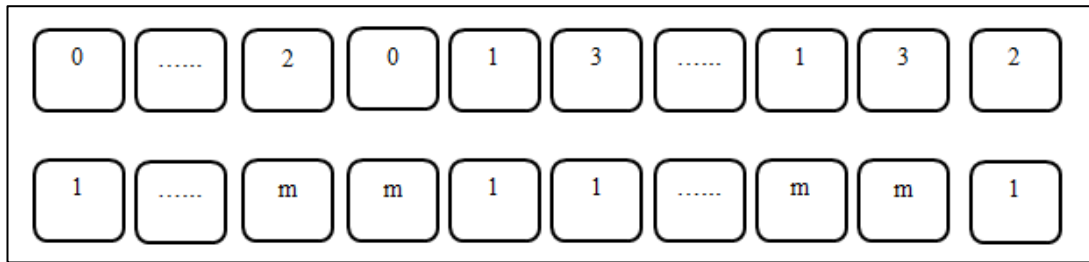


Figure 4.10: General Initial Current Order

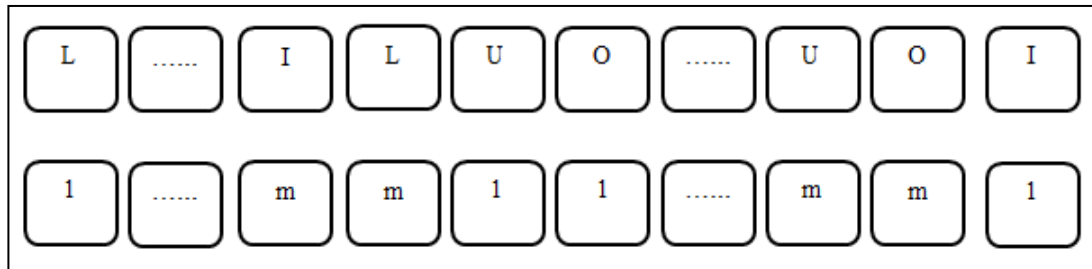


Figure 4.11: General Initial Current Order Represented by Activities L, U, I and O

The initial order was generated after testing a set of orders that in some cases seemed to be infeasible depending on the robot buffer capacity as stated before. It will be seen later in the results and discussion chapter that since this order represents a cycle for a robot with buffer capacity of size 1 then in some cases optimal solution will be reached from the first iteration.

#### 4.4.2 Defining x Variables and Setting Best Order

After defining all the constraints of the model, before calculating the current objective the variable  $x[a][i][b][j]$  is set to lower bound of 1 so that only the x variables that appear in the current order are set to 1 while others are 0. And this is coded by the algorithm shown in Figure (4.12).

```

for (i = 0; i < (4 * m - 1); i++)
    x[Corder[0][i]][Corder[1][i]][Corder[0][i + 1]][Corder[1][i + 1]].setLB(1);
x[Corder[0][4 * m - 1]][Corder[1][4 * m - 1]][Corder[0][0]][Corder[1][0]].setLB(1);

```

Figure 4.12: Setting x Variables to Lower Bound 1

So for example,  $x_{[Corder[0][0]][Corder[1][0]][Corder[0][1]] [Corder[1][1]]}$  is set to 1 for  $i = 0$ . This basically means that we just take two consecutive elements or activities from the array of  $Corder[i][j]$ .

After that current objective is found by CPLEX solver and then variables  $x_{[a][i][b][j]}$  for current order is set back to lower bound 0. The best order is set to be equal to the current order and the best objective is set to be equal to current objective.

#### **4.4.3 Starting the Iteration Loop**

In this algorithm the number of iterations is the stopping criteria. Since for cases when number of machines was 2 or 3 the optimal solution was found by the exact method it was easier to assume number of iterations required to reach optimal solution and that was around 1000 iterations. However, for 4 and 5 machine cases the number of iterations required trials until what can only seem as the minimum cycle time found and that was fixed as the number of iterations required. For a 4 machine case 2500 iterations were made and for 5 machine case the number of iterations ranged between 3000 to 5000.

When the runs were made by the SA algorithm it must be noted that each run was made 10 times which means if a case was run for 1000 iterations then those 1000 iterations were made 10 times. Out of those 10 times the result that gave the minimum cycle time was selected.

Then the first step was to set new order equal to the current order.

#### **4.4.4 Strategy Used for New Solution**

At this point a new solution is generated and a swapping method is used to generate the solution. The swap is conducted for 2 sets: one for activities and one for the

machine. So for example, if we have the initial current order for 2 machine case as the one in Figure (4.10) then a random number  $a$  and  $b$  is generated between 1 and  $4m - 1$ . Even though elements of  $j$  range between 0 and  $4m - 1$  we never swap with  $j = 0$  which is the loading activity of machine 1 and this to avoid permutations.

So if  $a = 4$  and  $b = 6$  there will be a swap between  $a$  and  $b$  and the new order will be as shown in Figure (4.13). It can be seen that the new order did not create any difference since a swap between two output buffer activities does not reduce distance matrix because the distance of travel will be the same. To avoid such a new order as well as to avoid swapping when  $a$  and  $b$  are equal conditions are used in the algorithm for the swap to be restricted as shown in Figure (4.14).

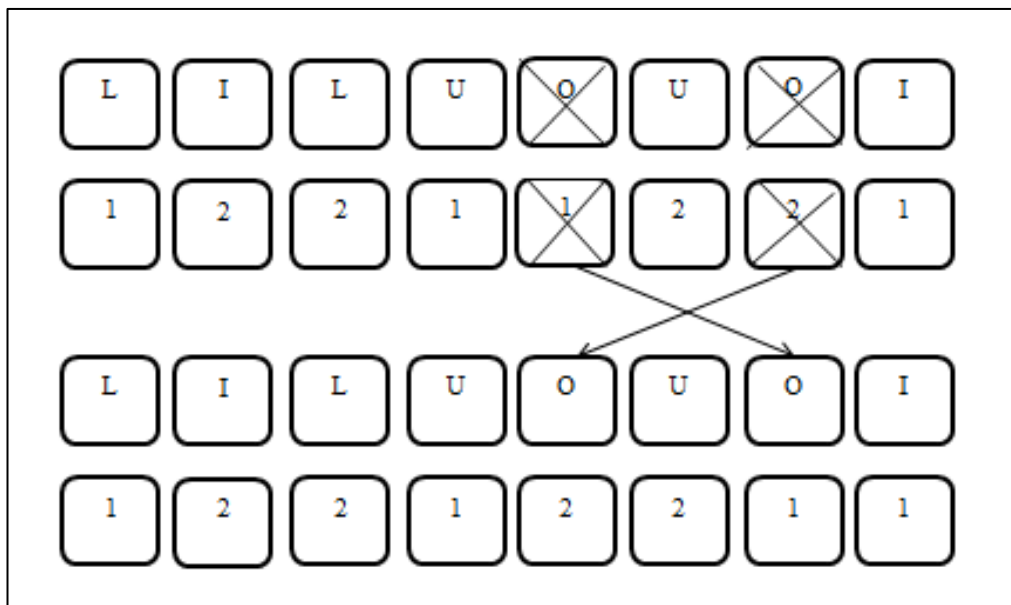


Figure 4.13: Swapping Strategy for New Solution to be generated

```

else {
    a = rand() % ((4 * m) - 1) + 1;
    b = rand() % ((4 * m) - 1) + 1;

    while ((a == b) || ((Norder[0][a] == 2) && (Norder[0][b] == 2)) || ((Norder[0][a] == 3) && (Norder[0][b] == 3)))
    {
        b = rand() % ((4 * m) - 1) + 1;
    }

    temp = Norder[0][a];
    Norder[0][a] = Norder[0][b];
    Norder[0][b] = temp;
    temp = Norder[1][a];
    Norder[1][a] = Norder[1][b];
    Norder[1][b] = temp;
}

```

Figure 4.14: Swapping Strategy to Generate New Solution

It can be seen that this swap is conducted for  $i = 0$  and  $i = 1$  which is activities and machines at the same time. It was also previously mentioned that since the order for a robot with buffer capacity of size 1 is very restricted since for each machine the order should always be the same that is Input buffer, Load, Unload, Output buffer a separate swap method with a condition was coded just for this case and it is shown in Figure (4.15).

```

if (RBK == 1)
{
    a = rand() % ((2 * m) - 1) + 1;
    b = rand() % ((2 * m) - 1) + 1;
    while (a == b)
    {
        b = rand() % ((2 * m) - 1) + 1;
    }

    temp = Norder[0][2 * a];
    Norder[0][2 * a] = Norder[0][2 * b];
    Norder[0][2 * b] = temp;
    temp = Norder[1][2 * a];
    Norder[1][2 * a] = Norder[1][2 * b];
    Norder[1][2 * b] = temp;

    temp = Norder[0][(2 * a) - 1];
    Norder[0][(2 * a) - 1] = Norder[0][(2 * b) - 1];
    Norder[0][(2 * b) - 1] = temp;
    temp = Norder[1][(2 * a) - 1];
    Norder[1][(2 * a) - 1] = Norder[1][(2 * b) - 1];
    Norder[1][(2 * b) - 1] = temp;
}

```

Figure 4.15: Swapping Strategy for Robot Buffer Capacity 1

The code has a condition that if  $a$  and  $b$  are equal no swap will be conducted. In this case,  $a$  and  $b$  are random numbers between 1 and  $2m - 1$ . So if we deal with a 2 machine case with initial order as shown in Figure (4.10). Then  $a$  can either be 1, 2 or 3 and  $b$  can either be 1, 2 or 3 that means there are 6 possible swaps ( $a = 1, b = 2$ ), ( $a = 1, b = 3$ ), ( $a = 2, b = 1$ ), ( $a = 2, b = 3$ ), ( $a = 3, b = 1$ ) and ( $a = 3, b = 2$ ) and since 3 of them are the same to the other 3 that means there are 3 unique new orders. So in general there are  $(2m - 1)(2m - 2)/2$  unique new orders for  $m$  machine case.

It can also be seen from the code that there will be two sets of swaps since as stated before  $I$  should be followed by  $L$  and  $U$  should be followed by  $O$ . These 3 new orders for the 2 machine case are as shown in Figure (4.16).

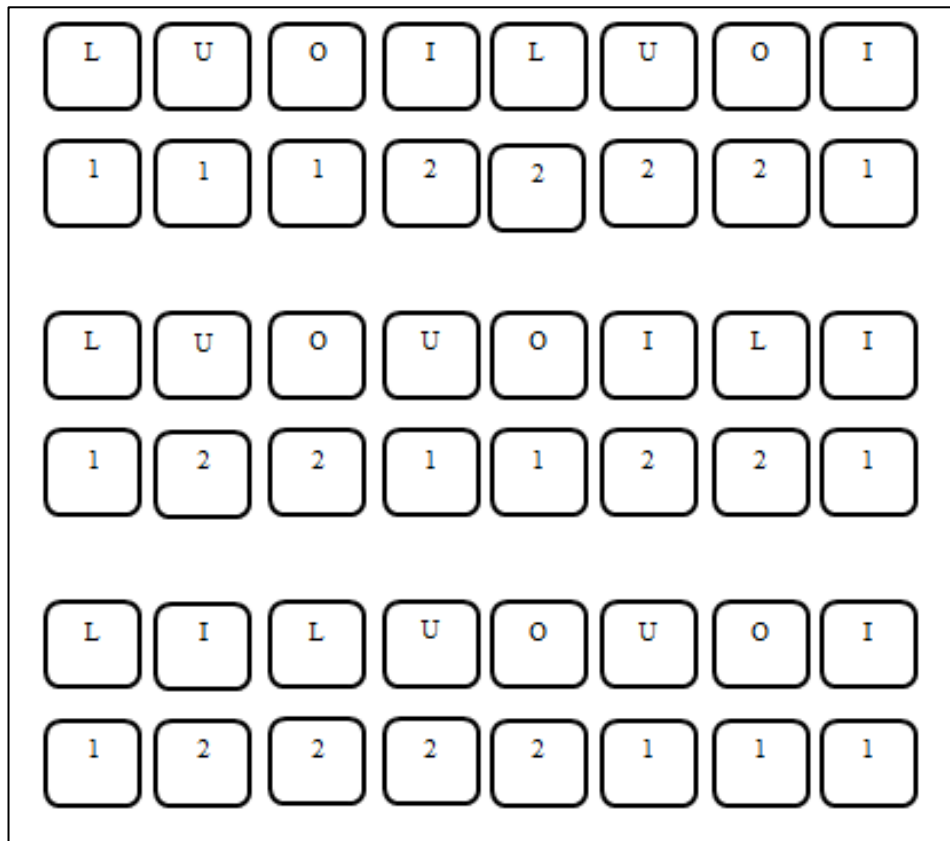


Figure 4.16: New Orders generated for 2 Machines with  $K = 1$

After that a new order is printed after the swap. And variables  $x[a][i][b][j]$  is set to lower bound of 1 so that only the new order  $x[a][i][b][j]$  variables are set to 1 while others are 0. And then the new objective is found by CPLEX solver.

#### 4.4.5 Condition for Finding a Better Solution

The first condition states that if the new objective is less than best objective found up to this point than the new objective is the best objective and the new order is the best order given that the order is feasible.

The second condition states that if the new objective is less than current objective found than the new objective is the current objective and the new order is the current order given that the order is feasible.

The third condition states that if  $\text{Random Number} < e^{\frac{\text{Current objective} - \text{New objective}}{T}}$  then we set the new objective to be equal to the current objective and the new order to be the current order with the control parameter being  $T = \lambda * T_1$  where the ratio  $\lambda$  is  $0 < \lambda < 1$  and  $T$  is known as temperature. This technique is known as the cooling schedule and it is used rather than some fixed number because the difference between the new and current objective is taken into account and thus the acceptance of the bad solution decrease if the difference is big or if the temperature decreases.

Otherwise, if the order is infeasible and if  $\text{Random Number} < e^{\frac{Worse}{T}}$  then we set the new objective to be equal to the current objective and the new order to be the current order which allows us to except infeasible orders. Where *Worse* is any negative big number such as -1000.



Also in the SA algorithm since the parameters ratio and worse affect the possibility of getting the optimal solution they had to be varied according to the number of machines. The parameter ratio was kept constant at 0.99 for all cases. However the parameter worse was -5000 for the 2 machine case and -10000 for the 3, 4 and 5 machine case.

Finally the variables  $x[a][i][b][j]$  for the new order is set back to lower bound 0. And then  $x[a][i][b][j]$  is set to lower bound of 1 so that only the best order  $x[a][i][b][j]$  variables are set to 1 while others are 0. Lastly, the best objective is computed by CPLEX solver if the order is feasible.

## Chapter 5

### RESULTS AND DISCUSSION

#### 5.1 Cycle Time Calculation

In this section, a two machine case with robot buffer capacity 1 and 2 are given as examples to show how the cycle time is calculated along with the other decision variables that include the completion time of the activities ( $t_a$ ), the number of parts on the robot that are unfinished at the end of activity a ( $Y_a^-$ ), the number of parts on the robot that are finished at the end of activity a ( $Y_a^+$ ) and the binary decision variables ( $x_{ab}$ ) and ( $z_i$ ). The schedule in this example is the optimal solution of these two cases that was found by the MIP model.

**Case 1:**  $m = 2, \delta = 2, \varepsilon = 1, K = 1, P = 22$

Table 5.1: Case when  $m = 2, \delta = 2, \varepsilon = 1, K = 1, P = 22$

Activity (a)	$L_1$	$U_2$	$O_2$	$I_2$	$L_2$	$U_1$	$O_1$	$I_1$	C
$X_{ab} = 1$	$X_{L_1U_2}$	$X_{U_2O_2}$	$X_{O_2I_2}$	$X_{I_2L_2}$	$X_{L_2U_1}$	$X_{U_1O_1}$	$X_{O_1I_1}$	$X_{I_1L_1}$	38
$t_a$	0	3	6	13	18	23	28	35	
$Y_a^+$	0	1	0	0	0	1	0	0	
$Y_a^-$	0	0	0	1	0	0	0	1	

Since the schedule of robot moves is as shown above that indicates that robot performs activity  $L_1$  before  $U_2$  ( $x_{L_1U_2} = 1$ ),  $U_2$  before  $O_2$  ( $x_{U_2O_2} = 1$ ),  $O_2$  before  $I_2$  ( $x_{O_2I_2} = 1$ ),  $I_2$  before  $L_2$  ( $x_{I_2L_2} = 1$ ),  $L_2$  before  $U_1$  ( $x_{L_2U_1} = 1$ ),  $U_1$  before  $O_1$  ( $x_{U_1O_1} =$

1),  $O_1$  before  $I_1$  ( $x_{O_1 I_1} = 1$ ) and finally  $I_1$  before  $L_2$  ( $x_{I_1 L_2} = 1$ ). Thus, other combination of activities will be zero and this is represented by equations (4.2) and (4.3) in the mathematical model.

$Y_{L_1}^-$  and  $Y_{L_2}^-$  are 0 because at the end of loading a machine with an unfinished part the robot has no parts left on it. In the model, this is represented by equations (4.23) and (4.24). Also it must be noted that  $Y_{L_1}^+$  and  $Y_{L_2}^+$  will not exist because the robot will not load the machine with a **finished** part. This is represented by equations (4.19) and (4.20) in the model. On the other hand,  $Y_{U_1}^+$  and  $Y_{U_2}^+$  are 1 because at the end of unloading a machine with a finished part the robot has one part. In the model, this is represented by equations (4.13) and (4.14). Also it must be noted that  $Y_{U_1}^-$  and  $Y_{U_2}^-$  will not exist because the robot will not unload the machine with an **unfinished** part. This is represented by equations (4.25) and (4.26) in the model. When a part is taken to the output buffer it means a finished part is put to the output buffer thus the robot has no parts left so  $Y_{O_1}^+$  and  $Y_{O_2}^+$  are 0. This is represented by equations (4.15) and (4.16) in the model. While equations (4.27) and (4.28) indicate that  $Y_{O_1}^-$  and  $Y_{O_2}^-$  will not exist since the robot will not put an **unfinished** part to the output buffer. And when a part is taken from the input buffer it means an unfinished part is picked up from the input buffer thus the robot has one part so  $Y_{I_1}^-$  and  $Y_{I_2}^-$  are 1. This is represented by equations (4.21) and (4.22) in the model. While equations (4.17) and (4.18) indicate that  $Y_{I_1}^+$  and  $Y_{I_2}^+$  will not exist since the robot will not pick up a **finished** part from the input buffer.

Table 5.2: Describing Completion Time Calculation for Case when  $m = 2, \delta = 2, \varepsilon = 1, K = 1, P = 22$

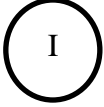


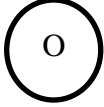
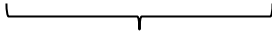
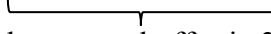
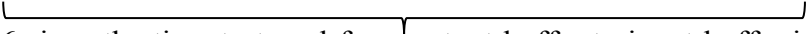
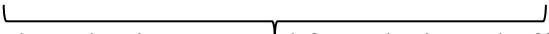
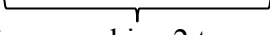

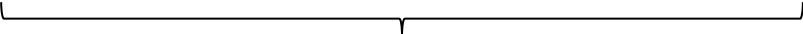
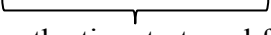
Buffer/ Machine $i$				
$L_1U_2$				
	$\delta = 2$ since the time to travel from machine 1 to machine 2 is 2 time unit and $\varepsilon = 1$ because at machine 2 the part is picked up so unloading time is 1 time unit, $d_{L_1U_2} = \varepsilon +  i - j \delta = 1 +  1 - 2  (2) = 1 + 2 = 3$ and $t_{U_2} = 3$			
$U_2O_2$				
	$\delta = 2$ since the time to travel from machine 2 to the output buffer is 2 time unit and $\varepsilon = 1$ because at the output buffer the part is put so loading time is 1 time unit, $d_{U_2O_2} = \varepsilon + (m - i + 1) \delta = 1 + (2 - 2 + 1) (2) = 1 + 2 = 3$ and $t_{O_2} = 3 + 3 = 6$			
$O_2I_2$				
	$3*\delta = 6$ since the time to travel from output buffer to input buffer is 6 time unit and $\varepsilon = 1$ because at the input buffer the part is picked up so unloading time is 1 time unit, $d_{O_2I_2} = \varepsilon + (m + 1) \delta = 1 + (2 + 1) (2) = 1 + 6 = 7$ and $t_{I_2} = 6 + 7 = 13$ .			
$I_2L_2$				
	$2*\delta = 4$ since the time to travel from the input buffer to the machine 2 is 4 time unit and $\varepsilon = 1$ because at machine 2 the part is put so loading time is 1 time unit, $d_{I_2L_2} = \varepsilon + i \delta = 1 + (2) (2) = 1 + 4 = 5$ and $t_{L_2} = 13 + 5 = 18$ .			
$L_2U_1$				
	$\delta = 2$ since the time to travel from machine 2 to machine 1 is 2 time unit and $\varepsilon = 1$ because at machine 1 the part is picked up so unloading time is 1 time unit, $d_{L_2U_1} = \varepsilon +  i - j \delta = 1 +  2 - 1  (2) = 1 + 2 = 3$ and $t_{U_1} = 18 + 3 + 2 = 23$ where 2 is waiting time since machine 1 had to be unloaded and process time of 22 time units was not fulfilled by the time the robot was ready.			
$U_1O_1$				
	$2*\delta = 4$ since the time to travel from machine 1 to the output buffer is 4 time unit and $\varepsilon = 1$ because at the output buffer the part is put so loading time is 1 time unit, $d_{U_1O_1} = \varepsilon + (m - i + 1) \delta = 1 + (2 - 1 + 1) (2) = 1 + 4 = 5$ and $t_{O_1} = 23 + 5 = 28$ .			
$O_1I_1$				
	$3*\delta = 6$ since the time to travel from output buffer to input buffer is 6 time unit and $\varepsilon = 1$ because at the input buffer the part is picked up so unloading time is 1 time unit, $d_{O_1I_1} = \varepsilon + (m + 1) \delta = 1 + (2 + 1) (2) = 1 + 6 = 7$ and $t_{I_1} = 28 + 7 = 35$ .			
$I_1L_1$				
	$\delta = 2$ since the time to travel from the input buffer to the machine 1 is 2 time unit and $\varepsilon = 1$ because at machine 1 the part is put so loading time is 1 time unit, $d_{I_1L_1} = \varepsilon + i \delta = 1 + (1) (2) = 1 + 2 = 3$ and $C = 35 + 3 = 38$ .			

Table (5.2) indicates how the completion times for each activity is calculated and equation (4.4) was used when finding the completion time of activity  $b$  while (4.6) was applied in finding  $t_{U_1}$  since the completion time of activity  $U_1$  must be greater or equal to the completion time of  $L_1$  which is  $0 +$  the unloading time which is  $1$  and the process time which is  $22$ , thus  $t_{U_1}$  must be greater or equal  $23$ . Equation (4.8) states that the cycle time is greater or equal to the completion time of  $I_1$  which is  $35 +$  (the distance matrix  $d_{I_1L_1}$  which is  $3$ )\*( $X_{I_1L_1}$  which is  $1$ ) thus  $C$  is greater or equal to  $38$ .

It must also be noted that in this case  $z_1 = 1$  since for machine 1 activity  $U_1$  is performed after  $L_1$ . However,  $z_2 = 0$  since for machine 2 activity  $U_2$  is performed before  $L_2$ .

The Gantt Chart for this case is shown in Figure (5.1). A Gantt chart is used to represent the processing time by each machine from start to end and for a certain period. And also for the robot it represents the start and end for completion of each of the activities in the chart and thus the horizontal axis is titled as time. For this 2 machine case it can be seen that for machine 1 after processing time of  $22$  time units the machine will stay idle for another  $16$  time units with no part on it. As for machine 2 processing of the part starts at time  $18$  and it processed for  $22$  time units. The machine is also idle with no part on it for  $16$  time units. The completion time of the activities are represented by two bars one bar for total travel time and one bar for the time taken for the robot to pick up or leave a part. It can also be seen that before the unloading operation of machine 1 there was some waiting time for the robot and this is because the machine did not finish processing. In some cases when two different activities are carried out on the same machine then there will be no travel time.

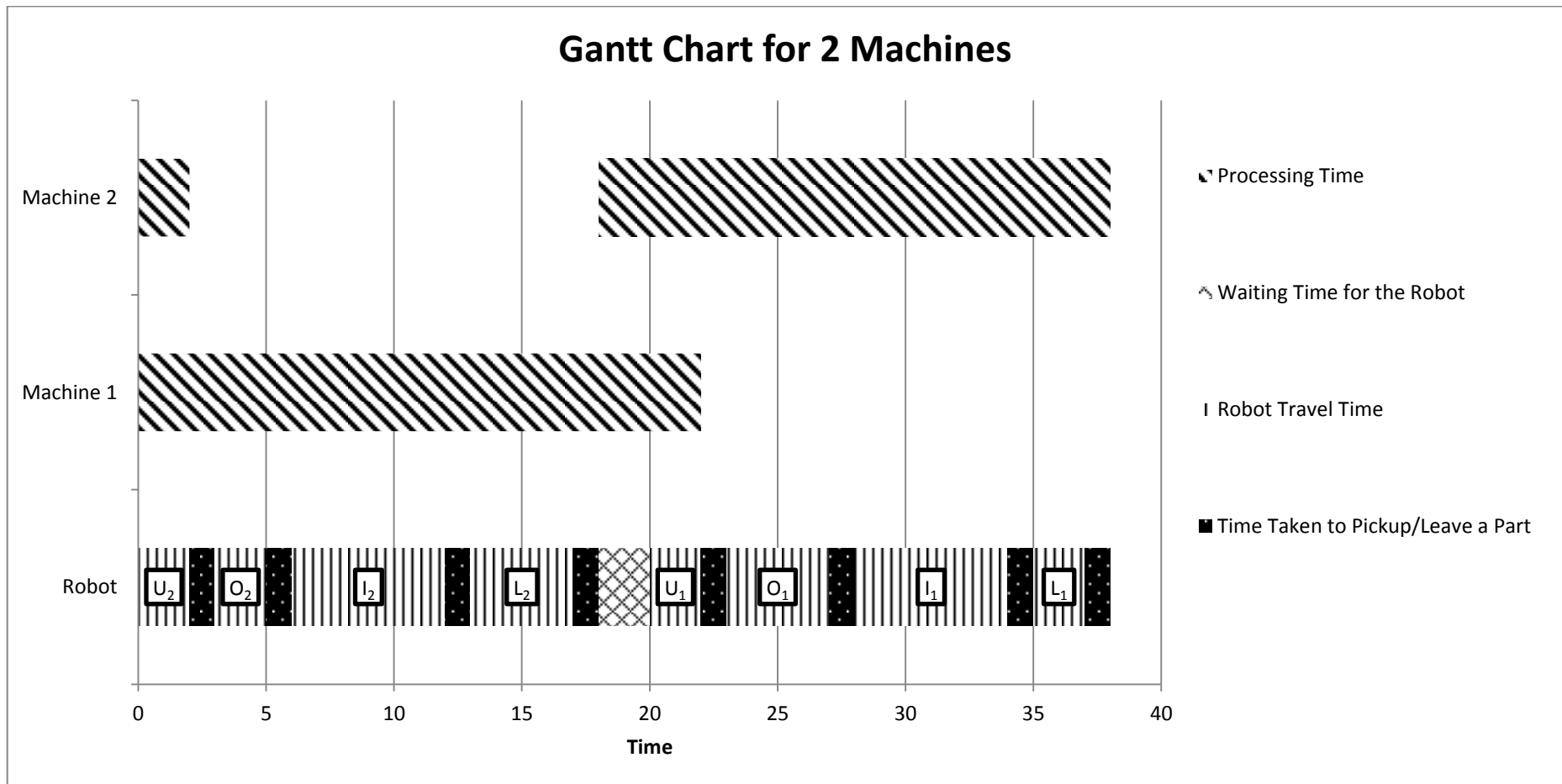


Figure 5.1: Gantt Chart for Case when  $m = 2$ ,  $\delta = 2$ ,  $\varepsilon = 1$ ,  $K = 1$ ,  $P = 22$

**Case 2:**  $m = 2, \delta = 2, \varepsilon = 1, K = 2, P = 22$

Table 5.3: Case when  $m = 2, \delta = 2, \varepsilon = 1, K = 2, P = 22$

Activity (a)	L <sub>1</sub>	I <sub>2</sub>	O <sub>1</sub>	U <sub>2</sub>	L <sub>2</sub>	O <sub>2</sub>	I <sub>1</sub>	U <sub>1</sub>	C
$X_{ab} = \mathbf{1}$	$X_{L_1I_2}$	$X_{I_2O_1}$	$X_{O_1U_2}$	$X_{U_2L_2}$	$X_{L_2O_2}$	$X_{O_2I_1}$	$X_{I_1U_1}$	$X_{U_1L_1}$	28
$t_a$	0	3	10	13	14	17	24	27	
$Y_a^+$	1	1	0	1	1	0	0	1	
$Y_a^-$	0	1	1	1	0	0	1	1	

One of the most important realizations of how increase in robot buffer capacity can lead to minimization in cycle time is that when robot buffer capacity is just 1 that indicates that the schedule of robot moves will always be constant in such a way that after loading a machine with an unfinished part then the robot can perform either of the following:

1. If the second machine is empty then the robot can go back to the input buffer and take an unfinished part and load the second machine.
2. If the second machine had a part and processing of that part is finished then the robot can move to the second machine and unload the finished part. Obviously, after unloading a finished part the robot has to go to the output buffer for unloading the finished part, only then can it go back to the input buffer to pick up an unfinished part.

However, it can be seen from the schedule of robot moves for the case with robot buffer capacity of size 2 that these restrictions are minimized since after picking up an unfinished part from the input buffer the robot can also unload a machine that finished processing as a second activity and having two unloading activities one after

another indicates that two parts are held by the robot at the same time. Also, after that when a machine is loaded with the unfinished part and the robot is still holding a finished part the robot can go back to the input buffer to pick up another part and at this point the robot is again holding two parts at the same time. After that it can be seen that the finished part is put to the output buffer but the robot is still holding one unfinished part and then machine 2 is unloaded with a finished part and at this point the robot is again holding two parts. The last two activities of the robot in that cycle include loading machine 1 with an unfinished part and then putting a finished part on the output buffer and then at this point it can be seen that the robot has no parts.

Thus, this sequence of activities has seen to reduce the cycle time which proves that increase in robot buffer capacity further minimizes cycle time.

## **2.2 Cycle Time for Robot Buffer Capacity $> m$ and $\leq 2m$**

In this section we discuss the impact of increasing in the robot buffer capacity to a size that is greater than the number of machines and less than or equal to a size that is two times the number of machines. So if we continue to consider a 2 machine case, we will have two more cases which are 2 machines with robot buffer capacity of 3 and 2 machines with robot buffer capacity of 4.



**Case 1:**  $m = 2, \delta = 2, \varepsilon = 1, K = 3, P = 22$

Table 5.4: Case when  $m = 2, \delta = 2, \varepsilon = 1, K = 3, P = 22$

Activity (a)	$L_1$	$O_2$	$U_2$	$L_2$	$O_1$	$I_1$	$I_2$	$U_1$	$C$
$X_{ab} = \mathbf{1}$	$X_{L_1O_2}$	$X_{O_2U_2}$	$X_{U_2L_2}$	$X_{L_2O_1}$	$X_{O_2I_1}$	$X_{I_1I_2}$	$X_{I_2U_1}$	$X_{U_1L_1}$	24
$t_a$	0	5	8	9	12	19	20	23	
$Y_a^+$	1	0	1	1	0	0	0	1	
$Y_a^-$	1	1	1	0	0	1	2	2	

When the buffer capacity is more than  $m$  which in this case is 3 the robot has the ability to hold two unfinished parts and one finished part or two finished parts and one unfinished part at the same time which indicates that rather than visiting the input buffer once followed by unloading a machine once now the robot has the ability to either visit the input buffer two times and hold two unfinished parts followed by visiting a machine and unloading a finished part or visiting the input buffer once and unloading two machines with 2 finished parts.

It can be seen from the schedule of robot moves above that the robot visits the input buffer and carries an unfinished part and then it carries another unfinished part from the input buffer. After that the robot visits machine 1 and unloads a finished part and at this moment the robot is holding 3 parts. Then the robot loads machine 1 with an unfinished part and after that it visits the output buffer to put a finished part. After that the robot moves to machine 2 to unload a finished part and then load the machine with an unfinished part. Lastly, the robot moves to the output buffer to put a finished part and at the end of this activity the robot has no parts being held.

The cycle time was seen to have further minimized with increase in the robot buffer capacity and this is due to the further flexibility in the robot move schedule.

**Case 2:**  $m = 2, \delta = 2, \varepsilon = 1, K = 4, P = 22$

Table 5.5: Case when  $m = 2, \delta = 2, \varepsilon = 1, K = 4, P = 22$

Activity (a)	L <sub>1</sub>	U <sub>2</sub>	L <sub>2</sub>	I <sub>1</sub>	I <sub>2</sub>	O <sub>2</sub>	O <sub>1</sub>	U <sub>1</sub>	C
$X_{ab} = \mathbf{1}$	$X_{L_1U_2}$	$X_{U_2L_2}$	$X_{L_2I_1}$	$X_{I_1I_2}$	$X_{I_2O_2}$	$X_{O_2O_1}$	$X_{O_1U_1}$	$X_{U_1L_1}$	24
$t_a$	0	3	4	9	10	17	18	23	
$Y_a^+$	1	2	2	2	2	1	0	1	
$Y_a^-$	1	1	0	1	2	2	2	2	

When robot buffer capacity is exactly two times the number of machines which in this case it is 4 it means that the robot can hold two finished parts and two unfinished parts at the same time indicating that it can visit the output buffer and input buffer one time because it means at some point when the robot is holding 2 finished parts after unloading two machines and then it visits the input buffer and picks up 2 unfinished parts it can then move to the output buffer to put those 2 finished parts to the output buffer.

The schedule of cycle moves above was as follows the robot goes to the input buffer to pick up an unfinished part having already carried two finished parts from a previous cycle and then it picks up another unfinished part from the input buffer and at this point the robot is holding 4 parts. Then it moves to the output buffer to put both the finished parts. After that the robot moves to machine 1 to unload a finished

part and then loads an unfinished part and then it moves to machine 2 to unload a finished part and then load an unfinished part.

It can be seen that for this case the cycle time could no longer be minimized and there may be two main reasons for this:

1. Process time effect.
2. Cumulative distance matrix plus completion time effect.

In section 5.4 the effect of process time on the cycle time with increasing robot buffer capacity will be further discussed.

### **5.3 Comparison between MIP Model and SA Algorithm Results**

In this section, the cycle time found by the MIP model for 2, 3 and 4 machine case with one buffer capacity is compared to the cycle time found by the SA algorithm for 2, 3, 4 and 5 machine cases with solution time recorded.

Six cases were considered for travelling time of 2 time units and loading/unloading time of 1 time units and the cases differed with differing process times ranging from 0, 22, 40, 50, 100 and 5000. These cases were considered so that the effect of process time along with waiting time can be later studied and discussed. Also the buffer capacity size ranged from size 1 to  $2m$  for each machine case.

It must also be noted that even though the solution time for the SA algorithm might seem bigger for the 2 machine case when compared to the MIP model, the time by which the algorithm reaches the optimal solution is actually shorter but the solution time was recorded when all the iterations were completed. This solution time convergence was portrayed by a graph with solution time on the X-axis and cycle

time on the Y-axis. These graphs were made for all the cases and the will be presented in Appendix A for Case 1, Case 2, Case 3, Case 4, and Case 5 and Appendix B for Case 6.

### 5.3.1 Case 1: $\delta = 2, \varepsilon = 1, P = 0$

Table 5.6: Case when  $\delta = 2, \varepsilon = 1, P = 0$

MIP Model				Simulated Annealing Algorithm			
No. of machines (m)	Buffer Capacity (K)	Cycle Time (time unit)	Solution Time (s)	No. of machines (m)	Buffer Capacity (K)	Cycle Time (time unit)	Solution Time (s)
2	1	32	0.06	2	1	32	42
	2	20	0.39		2	20	31
	3	20	0.42		3	20	40
	4	20	0.47		4	20	45
3	1	60	0.19	3	1	60	46
	2	44	60.56		2	44	30
	3	28	357.84		3	28	8
	4	28	560.95		4	28	43
	5	28	561.30		5	28	45
	6	28	635.67		6	28	47
4	1	96	1.86	4	1	96	124
					2	56	81
					3	56	22
					4	36	113
					5	36	124
					6	36	128
					7	36	132
					8	36	29
5				5	1	140	44
					2	92	112
					3	68	122
					4	68	139
					5	48	147
					6	44	169
					7	44	176
					8	44	177
					9	44	187
					10	44	187

It can be seen that the optimal solution was found by the SA algorithm when comparing the cycle time of the 2 and 3 machine cases with that of the MIP model as shown in Figure (5.2) and Figure (5.3), also with reduced time since most of the optimal solutions were actually found at 1 s or less even though completion of the iterations took longer. This proves that the SA algorithm modeled is adequate enough to be used.

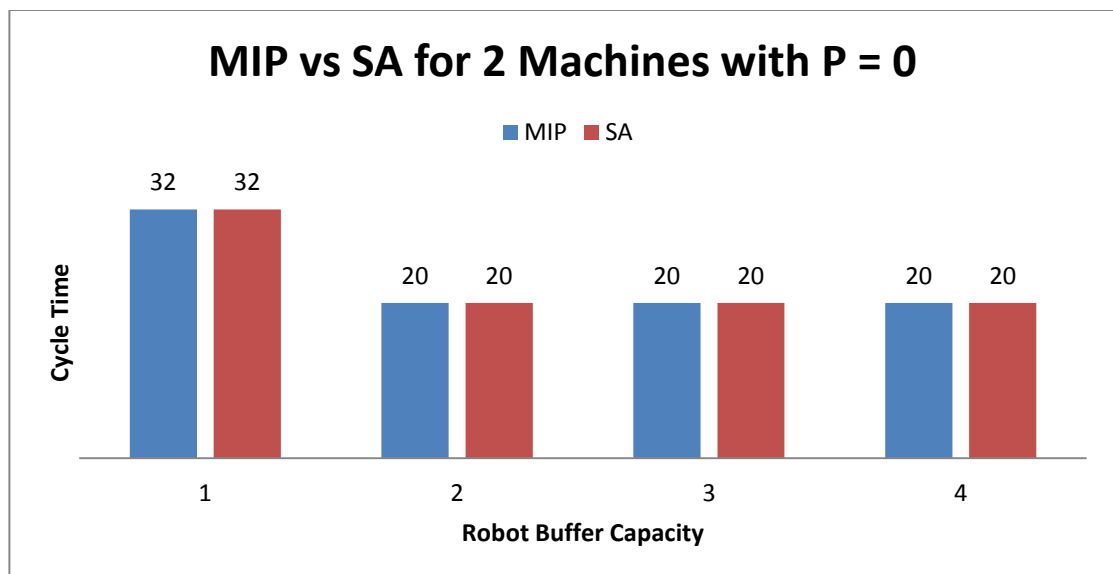


Figure 5.2: Comparison between MIP and SA Cycle Time for 2 Machines with P = 0

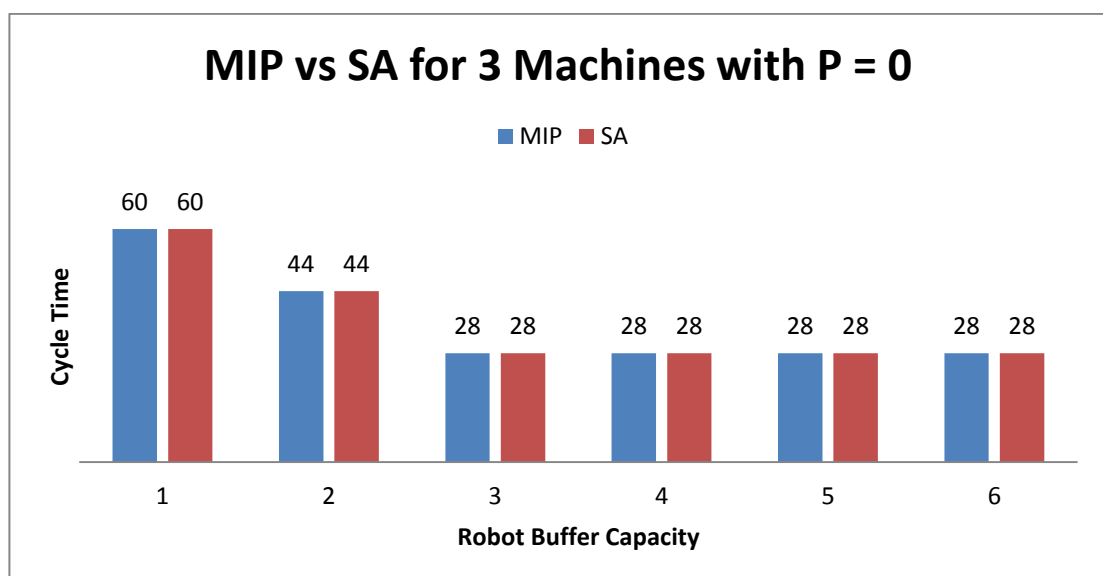


Figure 5.3: Comparison between MIP and SA Cycle Time for 3 Machines with P = 0

It must also be noted that when the MIP model was run for a 4 machine case it took longer than 24 hours without providing the optimal solution thus even though the optimality of some of the cycle time provided by the SA algorithm for the 4 and 5 machine case is not guaranteed, the solution time is extremely small compared to how long it would have taken the MIP model to find the optimal solution.

For the case when  $m = 4$ ,  $\delta = 2$ ,  $\varepsilon = 1$ ,  $P = 0$ ,  $K = 2$  it can be seen in Table (5.7) that since process time has no impact here forming the optimal schedule is easy since the buffer can hold two parts that means two loading/unloading activities followed by two output/input buffer putting/picking up activities will give the optimal schedule and hence we can prove whether the cycle time found by the SA algorithm was optimal. And it was seen that the cycle time was in fact optimal.

Similarly for the case when the buffer capacity is 3 which means the robot can hold three parts indicating three putting/picking up activities followed by three output/input buffer putting/picking up activities will give the optimal schedule and hence we can prove as shown in Table (5.8) whether the cycle time found by the SA algorithm was optimal. And it was seen that the cycle time was in fact optimal.

Table 5.7: Case when  $m = 4, \delta = 2, \varepsilon = 1, K = 2, P = 0$

<b>a</b>	<b>L<sub>1</sub></b>	<b>U<sub>1</sub></b>	<b>L<sub>2</sub></b>	<b>U<sub>2</sub></b>	<b>O<sub>1</sub></b>	<b>O<sub>2</sub></b>	<b>I<sub>1</sub></b>	<b>I<sub>2</sub></b>	<b>L<sub>3</sub></b>	<b>U<sub>3</sub></b>	<b>L<sub>4</sub></b>	<b>U<sub>4</sub></b>	<b>O<sub>3</sub></b>	<b>O<sub>4</sub></b>	<b>I<sub>3</sub></b>	<b>I<sub>4</sub></b>	<b>C</b>
<b>t<sub>a</sub></b>	0	1	4	5	12	13	24	25	32	33	36	37	40	41	52	53	56
<b>Y<sub>a</sub><sup>+</sup></b>	0	1	1	2	1	0	0	0	0	1	0	2	1	0	0	0	
<b>Y<sub>a</sub><sup>-</sup></b>	1	1	0	0	0	0	1	2	1	0	0	0	0	0	1	2	

Table 5.8: Case when  $m = 4, \delta = 2, \varepsilon = 1, K = 3, P = 0$

<b>a</b>	<b>L<sub>1</sub></b>	<b>U<sub>1</sub></b>	<b>L<sub>2</sub></b>	<b>U<sub>2</sub></b>	<b>L<sub>3</sub></b>	<b>U<sub>3</sub></b>	<b>O<sub>1</sub></b>	<b>O<sub>2</sub></b>	<b>O<sub>3</sub></b>	<b>I<sub>1</sub></b>	<b>I<sub>2</sub></b>	<b>I<sub>3</sub></b>	<b>L<sub>4</sub></b>	<b>U<sub>4</sub></b>	<b>O<sub>4</sub></b>	<b>I<sub>4</sub></b>	<b>C</b>
<b>t<sub>a</sub></b>	0	1	4	5	8	9	14	15	16	27	28	29	38	39	42	53	56
<b>Y<sub>a</sub><sup>+</sup></b>	0	1	1	2	2	3	2	1	0	0	0	0	0	1	0	0	
<b>Y<sub>a</sub><sup>-</sup></b>	2	2	1	1	0	0	0	0	0	1	2	3	2	2	2	3	

Similarly for the case when the buffer capacity is 4 which means the robot can hold four parts indicating four loading/unloading activities followed by four output/input buffer putting/picking up activities will give the optimal schedule and hence we can prove as shown in Table (5.9) whether the cycle time found by the SA algorithm was optimal. Also in this case the optimal solution was found.

For the cases when the robot buffer capacity was 5, 6, 7 and 8 it does not need to be proved that the optimal solution was found since 36 is the minimum cycle time that can be found in this case since the schedule discussed above is the minimum in terms of cumulative distance matrix plus completion time.

As per the same discussions the cycle times for the 5 machine case for buffer capacity 1, 2, 3, 4 and 5 will also be proved by optimal schedules when process time has no affect in Tables (5.10), (5.11), (5.12), (5.13) and (5.14). And for those cases it was also seen that all the cycle times were optimal except for the case when buffer capacity was 5. For the cases when the  $K$  was 6, 7, 8, 9 and 10 optimality did not need to be proved since 44 is the minimum cycle time that can be found.



Table 5.9: Case when  $m = 4, \delta = 2, \varepsilon = 1, K = 4, P = 0$

<b>a</b>	<b>L<sub>1</sub></b>	<b>U<sub>1</sub></b>	<b>L<sub>2</sub></b>	<b>U<sub>2</sub></b>	<b>L<sub>3</sub></b>	<b>U<sub>3</sub></b>	<b>L<sub>4</sub></b>	<b>U<sub>4</sub></b>	<b>O<sub>1</sub></b>	<b>O<sub>2</sub></b>	<b>O<sub>3</sub></b>	<b>O<sub>4</sub></b>	<b>I<sub>1</sub></b>	<b>I<sub>2</sub></b>	<b>I<sub>3</sub></b>	<b>I<sub>4</sub></b>	<b>C</b>
<b>t<sub>a</sub></b>	0	1	4	5	8	9	12	13	16	17	18	19	30	31	32	33	36
<b>Y<sub>a</sub><sup>+</sup></b>	0	1	1	2	2	3	3	4	3	2	1	0	0	0	0	0	
<b>Y<sub>a</sub><sup>-</sup></b>	3	3	2	2	1	1	0	0	0	0	0	0	1	2	3	4	

Table 5.10: Case when  $m = 5, \delta = 2, \varepsilon = 1, K = 1, P = 0$

<b>a</b>	<b>L<sub>1</sub></b>	<b>U<sub>1</sub></b>	<b>O<sub>1</sub></b>	<b>I<sub>1</sub></b>	<b>L<sub>2</sub></b>	<b>U<sub>2</sub></b>	<b>O<sub>2</sub></b>	<b>I<sub>2</sub></b>	<b>L<sub>3</sub></b>	<b>U<sub>3</sub></b>	<b>O<sub>3</sub></b>	<b>I<sub>3</sub></b>	<b>L<sub>4</sub></b>	<b>U<sub>4</sub></b>	<b>O<sub>4</sub></b>	<b>I<sub>4</sub></b>	<b>L<sub>5</sub></b>	<b>U<sub>5</sub></b>	<b>O<sub>5</sub></b>	<b>I<sub>5</sub></b>	<b>C</b>
<b>t<sub>a</sub></b>	0	1	12	25	30	31	40	53	60	61	68	81	90	91	96	109	120	121	124	137	140
<b>Y<sub>a</sub><sup>+</sup></b>	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	
<b>Y<sub>a</sub><sup>-</sup></b>	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	

Table 5.11: Case when  $m = 5, \delta = 2, \varepsilon = 1, K = 2, P = 0$

<b>a</b>	<b>L<sub>1</sub></b>	<b>U<sub>1</sub></b>	<b>L<sub>2</sub></b>	<b>U<sub>2</sub></b>	<b>O<sub>1</sub></b>	<b>O<sub>2</sub></b>	<b>I<sub>1</sub></b>	<b>I<sub>2</sub></b>	<b>L<sub>3</sub></b>	<b>U<sub>3</sub></b>	<b>L<sub>4</sub></b>	<b>U<sub>4</sub></b>	<b>O<sub>3</sub></b>	<b>O<sub>4</sub></b>	<b>I<sub>3</sub></b>	<b>I<sub>4</sub></b>	<b>L<sub>5</sub></b>	<b>U<sub>5</sub></b>	<b>O<sub>5</sub></b>	<b>I<sub>5</sub></b>	<b>C</b>	
<b>t<sub>a</sub></b>	0	1	4	5	14	15	28	29	36	37	40	41	46	47	60	61	72	73	76	89	92	
<b>Y<sub>a</sub><sup>+</sup></b>	0	1	1	2	1	2	0	0	0	1	1	2	1	0	0	0	0	1	0	0		
<b>Y<sub>a</sub><sup>-</sup></b>	1	1	0	0	0	0	1	2	1	1	0	0	0	0	1	2	1	1	1	2		

Table 5.12: Case when  $m = 5, \delta = 2, \varepsilon = 1, K = 3, P = 0$

<b>a</b>	<b>L<sub>1</sub></b>	<b>U<sub>1</sub></b>	<b>L<sub>2</sub></b>	<b>U<sub>2</sub></b>	<b>L<sub>3</sub></b>	<b>U<sub>3</sub></b>	<b>O<sub>1</sub></b>	<b>O<sub>2</sub></b>	<b>O<sub>3</sub></b>	<b>I<sub>1</sub></b>	<b>I<sub>2</sub></b>	<b>I<sub>3</sub></b>	<b>L<sub>4</sub></b>	<b>U<sub>4</sub></b>	<b>L<sub>5</sub></b>	<b>U<sub>5</sub></b>	<b>O<sub>4</sub></b>	<b>O<sub>5</sub></b>	<b>I<sub>4</sub></b>	<b>I<sub>5</sub></b>	<b>C</b>	
<b>t<sub>a</sub></b>	0	1	4	5	8	9	16	17	18	31	32	33	42	43	46	47	50	51	64	65	68	
<b>Y<sub>a</sub><sup>+</sup></b>	0	1	1	2	2	3	2	1	0	0	0	0	0	1	1	2	1	0	0	0		
<b>Y<sub>a</sub><sup>-</sup></b>	2	2	1	1	0	0	0	0	0	1	2	3	2	2	1	1	1	1	2	3		

Table 5.13: Case when  $m = 5, \delta = 2, \varepsilon = 1, K = 4, P = 0$

<b>a</b>	<b>L<sub>1</sub></b>	<b>U<sub>1</sub></b>	<b>L<sub>2</sub></b>	<b>U<sub>2</sub></b>	<b>L<sub>3</sub></b>	<b>U<sub>3</sub></b>	<b>L<sub>4</sub></b>	<b>U<sub>4</sub></b>	<b>O<sub>1</sub></b>	<b>O<sub>2</sub></b>	<b>O<sub>3</sub></b>	<b>O<sub>4</sub></b>	<b>I<sub>1</sub></b>	<b>I<sub>2</sub></b>	<b>I<sub>3</sub></b>	<b>I<sub>4</sub></b>	<b>L<sub>5</sub></b>	<b>U<sub>5</sub></b>	<b>O<sub>5</sub></b>	<b>I<sub>5</sub></b>	<b>C</b>
<b>t<sub>a</sub></b>	0	1	4	5	8	9	12	13	18	19	20	21	34	35	36	37	48	49	52	65	68
<b>Y<sub>a</sub><sup>+</sup></b>	0	1	1	2	2	3	3	4	3	2	1	0	0	0	0	0	0	1	0	0	
<b>Y<sub>a</sub><sup>-</sup></b>	3	3	2	2	1	1	0	0	0	0	0	0	1	2	3	4	3	3	3	4	

Table 5.14: Case when  $m = 5, \delta = 2, \varepsilon = 1, K = 5, P = 0$

<b>a</b>	<b>L<sub>1</sub></b>	<b>U<sub>1</sub></b>	<b>L<sub>2</sub></b>	<b>U<sub>2</sub></b>	<b>L<sub>3</sub></b>	<b>U<sub>3</sub></b>	<b>L<sub>4</sub></b>	<b>U<sub>4</sub></b>	<b>L<sub>5</sub></b>	<b>U<sub>5</sub></b>	<b>O<sub>1</sub></b>	<b>O<sub>2</sub></b>	<b>O<sub>3</sub></b>	<b>O<sub>4</sub></b>	<b>O<sub>5</sub></b>	<b>I<sub>1</sub></b>	<b>I<sub>2</sub></b>	<b>I<sub>3</sub></b>	<b>I<sub>4</sub></b>	<b>I<sub>5</sub></b>	<b>C</b>
<b>t<sub>a</sub></b>	0	1	4	5	8	9	12	13	16	17	20	21	22	23	24	37	38	39	40	41	44
<b>Y<sub>a</sub><sup>+</sup></b>	0	1	1	2	2	3	3	4	4	5	4	3	2	1	0	0	0	0	0	0	
<b>Y<sub>a</sub><sup>-</sup></b>	4	4	3	3	2	2	1	1	0	0	0	0	0	0	0	1	2	3	4	5	

**5.3.2 Case 2:  $\delta = 2, \varepsilon = 1, P = 22$**

Table 5.15: Case when  $\delta = 2, \varepsilon = 1, P = 22$

MIP Model				Simulated Annealing Algorithm			
No. of machines (m)	Buffer Capacity (K)	Cycle Time (time unit)	Solution Time (s)	No. of machines (m)	Buffer Capacity (K)	Cycle Time (time unit)	Solution Time (s)
2	1	38	0.09	2	1	38	62
	2	28	0.26		2	28	42
	3	24	0.33		3	24	71
	4	24	0.45		4	24	58
3	1	60	0.31	3	1	60	56
	2	44	21.52		2	44	37
	3	40	227.83		3	40	46
	4	28	65.70		4	28	54
	5	28	70.02		5	28	59
	6	28	71.78		6	28	61
4	1	96	1.70	4	1	96	61
					2	68	66
					3	56	92
					4	52	103
					5	36	141
					6	36	118
					7	36	125
					8	36	117
5				5	1	140	167
					2	92	90
					3	72	101
					4	68	128
					5	60	275
					6	44	311
					7	44	325
					8	44	306
					9	44	355
					10	44	345

It can be seen that the optimal solution was found by the SA algorithm when comparing the cycle time of the 2 and 3 machine cases with that of the MIP model as shown in Figure (5.4) and (5.5), also with reduced time since most of the optimal solutions were actually found at times ranging between 0 and 2 s for the 2 machine

case and times ranging between 7 and 35 s for the 3 machine case even though iteration completion took a longer time.

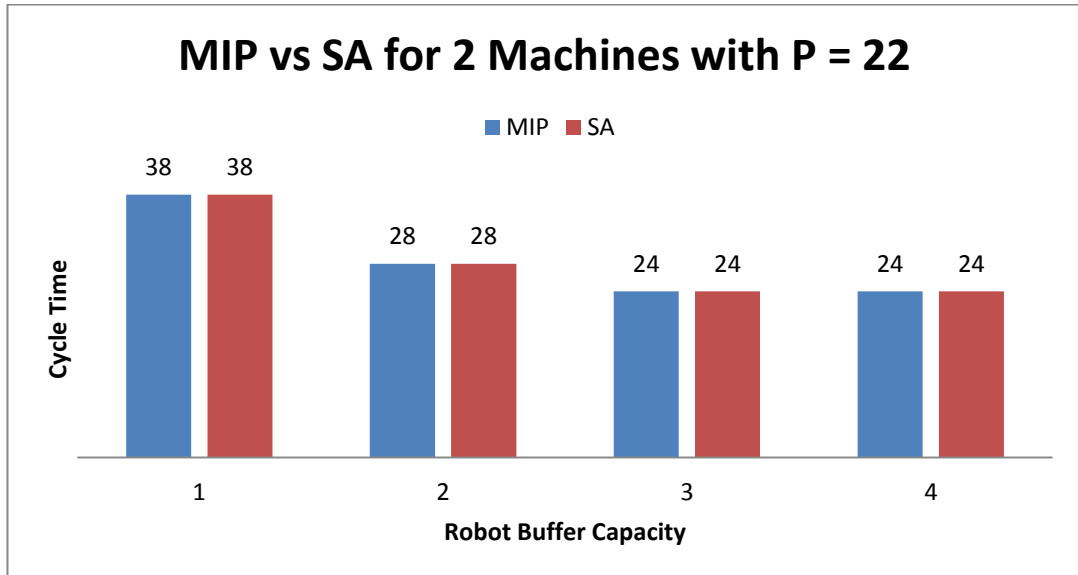


Figure 5.4: Comparison between MIP and SA Cycle Time for 2 Machines with P = 22

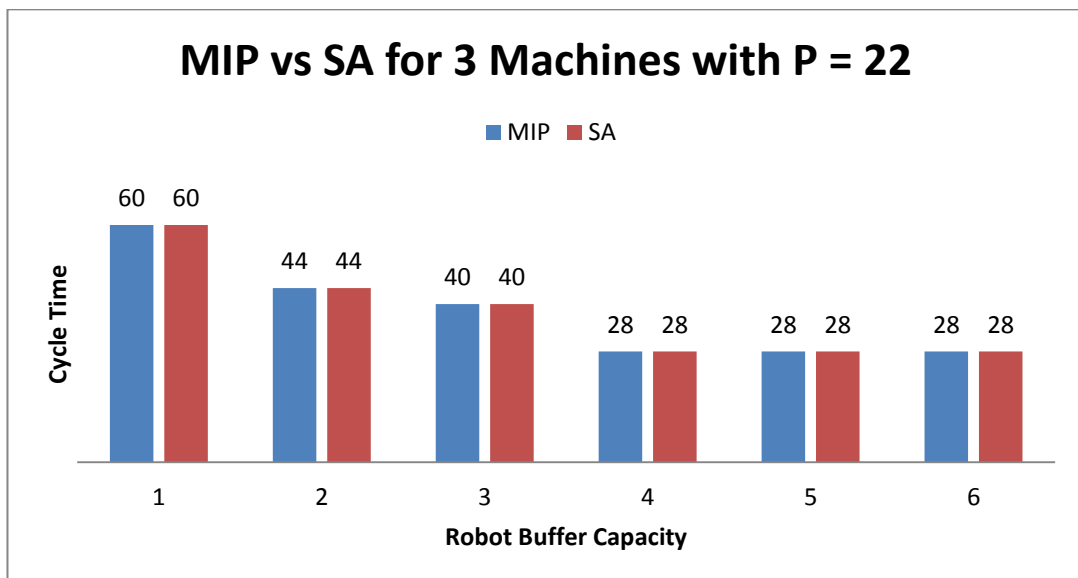


Figure 5.5: Comparison between MIP and SA Cycle Time for 3 Machines with P = 22

It is logically evident that with increase in process time the cycle time can never be smaller when compared to the cycle time of a process time that was smaller. So since

the cycle time for all buffer capacities in 4 and 5 machine case was equivalent to that of the cycle time when process time was 0 indicates that the cycle time is optimal. However, it can be seen that the cycle time for  $m = 4$ ,  $K = 2$  and  $K = 4$  as well as  $m = 5$ ,  $K = 3$  and  $K = 5$  were not equal to the cycle times when process time was 0.

For the case when  $m = 4$ ,  $\delta = 2$ ,  $\varepsilon = 1$ ,  $P = 22$ ,  $K = 2$  it can be seen that since process time has an impact forming the optimal schedule is not as easy. Since the buffer can hold two parts that means two loading/unloading activities followed by two output/input buffer putting/picking up activities will give the optimal schedule and hence we can prove whether the cycle time found by the SA algorithm was optimal. However, now along with that in mind we should also consider a schedule in such a way that we can avoid waiting time so that means that between a loading and unloading activity of any machine the total completion time of unloading must be at least the process time plus the time taken to pick up the finished part. Thus my suggestion was that the optimal schedule will look something like Table (5.16) where two input buffer activities are directly followed by two loading activities and also two unloading activities are directly followed by two output buffer activities. And it can be seen that the cycle time found was optimal.

Table 5.16: Case when  $m = 4, \delta = 2, \varepsilon = 1, K = 2, P = 22$

<b>a</b>	<b>L<sub>1</sub></b>	<b>L<sub>2</sub></b>	<b>I<sub>3</sub></b>	<b>I<sub>4</sub></b>	<b>L<sub>3</sub></b>	<b>U<sub>4</sub></b>	<b>L<sub>4</sub></b>	<b>U<sub>1</sub></b>	<b>O<sub>1</sub></b>	<b>O<sub>2</sub></b>	<b>U<sub>2</sub></b>	<b>U<sub>3</sub></b>	<b>O<sub>3</sub></b>	<b>O<sub>4</sub></b>	<b>I<sub>1</sub></b>	<b>I<sub>2</sub></b>	<b>C</b>
<b>t<sub>a</sub></b>	0	3	8	9	16	19	20	27	36	37	44	47	52	53	64	65	68
<b>m<sub>2</sub></b>		0	5	6	13	16	17	24	33	34	41						
<b>m<sub>3</sub></b>					0	3	4	11	20	21	28	31					
<b>m<sub>4</sub></b>	48	51	56	57	64	67	0	7	16	17	24	27	32	33	44	45	
<b>Y<sub>a</sub><sup>+</sup></b>	0	0	0	0	0	1	0	2	1	0	1	2	1	0	0	0	
<b>Y<sub>a</sub><sup>-</sup></b>	1	0	1	2	1	1	0	0	0	0	0	0	0	0	1	2	

The Gantt chart for this case is shown in Figure (5.6). In the Gantt chart for 2 machine case it was seen that the time a part stays on the machine between loading and unloading is exactly the process time. However in this case it can be seen that after 22 time units of processing for machine 1 the part stayed on the machine idle for 4 time units and for machine 2 the part stayed on the machine for 18 time units. For machine 3 the part stayed on the machine idle for 8 time units and for machine 4 the part stayed on the machine idle for 44 time units.

Since this case is a robot buffer with the ability to hold two parts then two output buffer activities can follow each other, two input buffer activities can follow each other or unloading activity can be followed by a loading activity.

Similarly for the case when the buffer capacity is 4 which means the robot can hold four parts indicating four putting/picking up activities followed by four output/input buffer loading/unloading activities will give the optimal schedule. However, since process again plays a role here a better schedule is having 4 input buffer activities followed by 4 loading/unloading activities and finally 4 output buffer activities as shown in Table (5.17). Also in this case the optimal solution was found.

As per the same discussions the cycle times for the 5 machine case for buffer capacity 3 and 5 are proved by optimal schedules in Table (5.18) and (5.19) when process time has an effect. The concept of a 5 machine with robot buffer capacity 3 is similar to that of a 4 machine with buffer capacity 2 and thus the generalized schedule of the case is applied. Also the 5 machine with robot buffer capacity 5 is similar to that of a 4 machine with buffer capacity 4 and thus a generalized schedule of the case will be applied and in both cases cycle time was optimal.



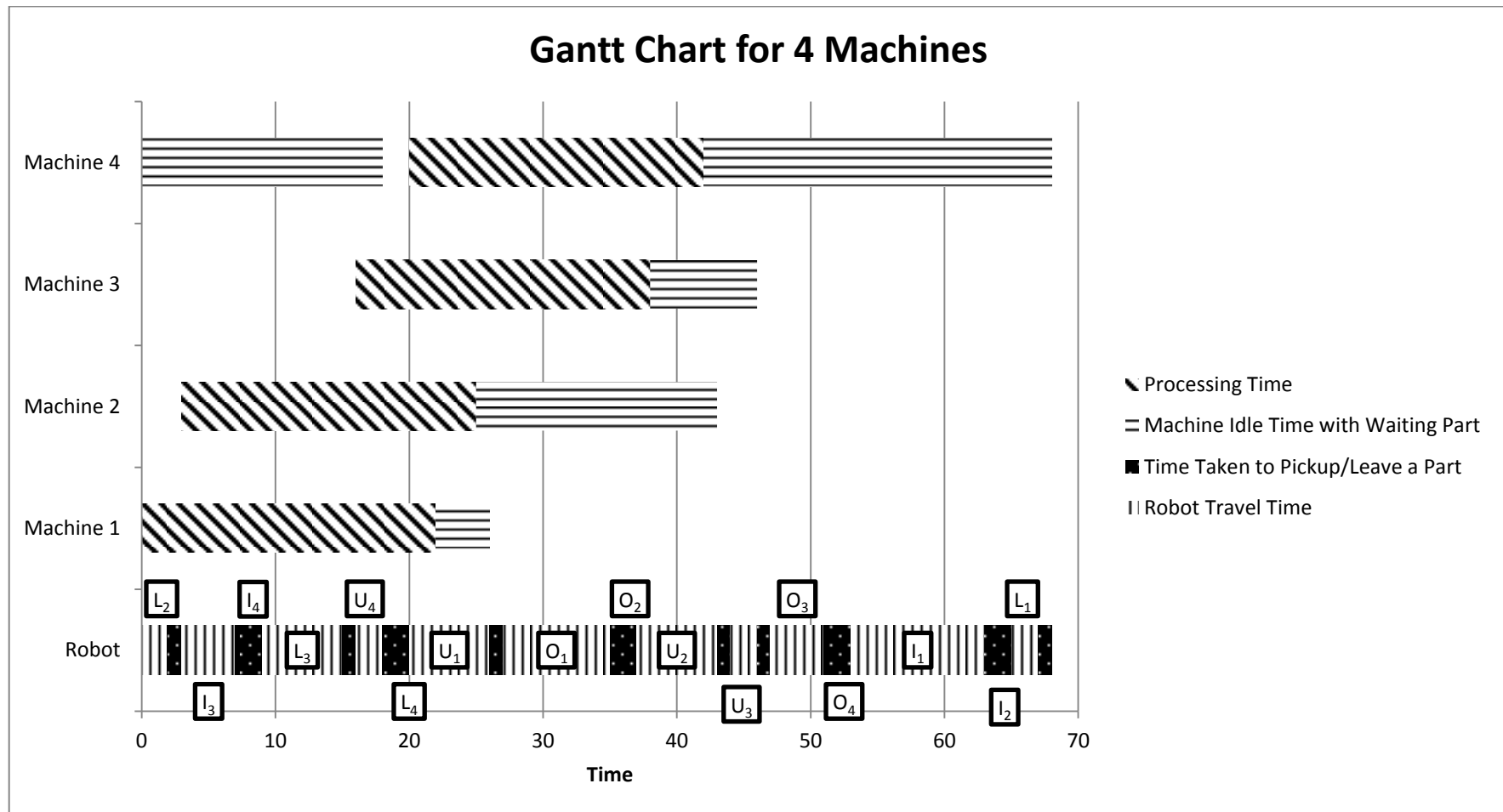


Figure 5.6: Gantt Chart for Case when  $m = 4$ ,  $\delta = 2$ ,  $\epsilon = 1$ ,  $K = 2$ ,  $P = 22$

Table 5.17: Case when  $m = 4$ ,  $\delta = 2$ ,  $\varepsilon = 1$ ,  $K = 4$ ,  $P = 22$

<b>a</b>	<b>L<sub>1</sub></b>	<b>U<sub>2</sub></b>	<b>L<sub>2</sub></b>	<b>U<sub>3</sub></b>	<b>L<sub>3</sub></b>	<b>U<sub>4</sub></b>	<b>L<sub>4</sub></b>	<b>U<sub>1</sub></b>	<b>O<sub>1</sub></b>	<b>O<sub>2</sub></b>	<b>O<sub>3</sub></b>	<b>O<sub>4</sub></b>	<b>I<sub>1</sub></b>	<b>I<sub>2</sub></b>	<b>I<sub>3</sub></b>	<b>I<sub>4</sub></b>	<b>C</b>
<b>t<sub>a</sub></b>	0	3	4	7	8	11	12	23	32	33	34	35	46	47	48	49	52
<b>m<sub>2</sub></b>	48	51	0	3	4	7	8	20	28	29	30	31	42	43	44	45	
<b>m<sub>3</sub></b>	44	47	48	51	0	3	4	15	24	25	26	27	38	39	40	41	
<b>m<sub>4</sub></b>	40	43	44	47	48	51	0	11	20	21	22	23	34	35	36	37	
<b>Y<sub>a</sub><sup>+</sup></b>	0	1	1	2	2	3	3	4	3	2	1	0	0	0	0	0	
<b>Y<sub>a</sub><sup>-</sup></b>	3	3	2	2	1	1	0	0	0	0	0	0	1	2	3	4	

Table 5.18: Case when  $m = 5, \delta = 2, \varepsilon = 1, K = 3, P = 22$

<b>a</b>	<b>L<sub>1</sub></b>	<b>L<sub>2</sub></b>	<b>I<sub>3</sub></b>	<b>I<sub>4</sub></b>	<b>I<sub>5</sub></b>	<b>L<sub>3</sub></b>	<b>U<sub>4</sub></b>	<b>L<sub>4</sub></b>	<b>U<sub>5</sub></b>	<b>L<sub>5</sub></b>	<b>O<sub>1</sub></b>	<b>O<sub>2</sub></b>	<b>U<sub>1</sub></b>	<b>U<sub>2</sub></b>	<b>U<sub>3</sub></b>	<b>O<sub>3</sub></b>	<b>O<sub>4</sub></b>	<b>O<sub>5</sub></b>	<b>I<sub>1</sub></b>	<b>I<sub>2</sub></b>	<b>C</b>
<b>t<sub>a</sub></b>	0	3	8	9	10	17	20	21	24	25	28	29	40	43	46	53	54	55	68	69	72
<b>m<sub>2</sub></b>		0	5	6	7	14	17	18	21	22	25	26	37	40							
<b>m<sub>3</sub></b>						0	3	4	7	8	11	12	23	26	29						
<b>m<sub>4</sub></b>	51	53	59	60	61	68	71	0	3	4	7	8	19	22	25	32	33	34	47	48	
<b>m<sub>5</sub></b>	47	50	55	56	57	64	67	68	71	0	3	4	15	18	21	28	29	30	43	44	
<b>Y<sub>a</sub><sup>+</sup></b>	0	0	0	0	0	0	1	1	2	2	1	0	1	2	3	2	1	0	0	0	
<b>Y<sub>a</sub><sup>-</sup></b>	1	0	1	2	3	2	2	1	1	0	0	0	0	0	0	0	0	0	1	2	

Table 5.19: Case when  $m = 5$ ,  $\delta = 2$ ,  $\varepsilon = 1$ ,  $K = 5$ ,  $P = 22$

<b>a</b>	<b>L<sub>1</sub></b>	<b>U<sub>2</sub></b>	<b>L<sub>2</sub></b>	<b>U<sub>3</sub></b>	<b>L<sub>3</sub></b>	<b>U<sub>4</sub></b>	<b>L<sub>4</sub></b>	<b>U<sub>5</sub></b>	<b>L<sub>5</sub></b>	<b>U<sub>1</sub></b>	<b>O<sub>1</sub></b>	<b>O<sub>2</sub></b>	<b>O<sub>3</sub></b>	<b>O<sub>4</sub></b>	<b>O<sub>5</sub></b>	<b>I<sub>1</sub></b>	<b>I<sub>2</sub></b>	<b>I<sub>3</sub></b>	<b>I<sub>4</sub></b>	<b>I<sub>5</sub></b>	<b>C</b>
<b>t<sub>a</sub></b>	0	3	4	7	8	11	12	15	16	25	36	37	38	39	40	53	54	55	56	57	60
<b>m<sub>2</sub></b>	57	0	1	4	5	8	9	12	13	22	33	34	35	36	37	50	51	52	53	54	
<b>m<sub>3</sub></b>	53	56	57	0	1	4	5	8	9	18	29	30	31	32	33	46	47	48	49	50	
<b>m<sub>4</sub></b>	49	52	53	56	57	0	1	4	5	14	25	26	27	28	29	42	43	44	45	46	
<b>m<sub>5</sub></b>	45	48	49	52	53	56	57	0	1	10	21	22	23	24	25	38	39	40	41	42	
<b>Y<sub>a</sub><sup>+</sup></b>	0	1	1	2	2	3	3	4	4	5	4	3	2	1	0	0	0	0	0	0	
<b>Y<sub>a</sub><sup>-</sup></b>	4	4	3	3	2	2	1	1	0	0	0	0	0	0	0	1	2	3	4	5	

### 5.3.3 Case 3: $\delta = 2$ , $\varepsilon = 1$ , $P = 40$

It can be seen that the optimal solution was found by the SA algorithm when comparing the cycle time of the 2 and 3 machine cases with that of the MIP model as shown in Figure (5.7) and (5.8), also with reduced time since most of the optimal solutions were actually found at times ranging between 0 and 4 s for the 2 machine case and times ranging between 0 and 9 s for the 3 machine case.

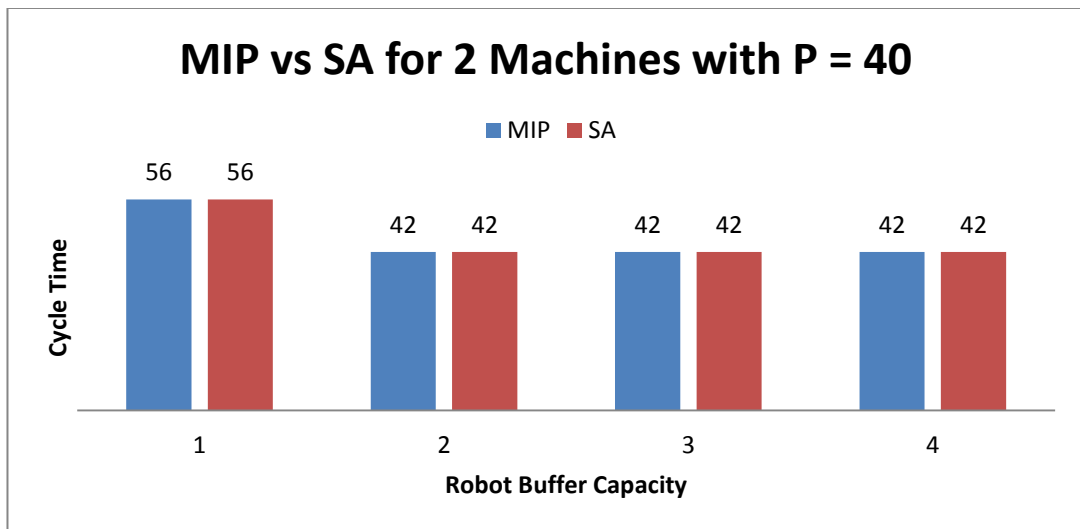


Figure 5.7: Comparison between MIP and SA Cycle Time for 2 Machines with P = 40

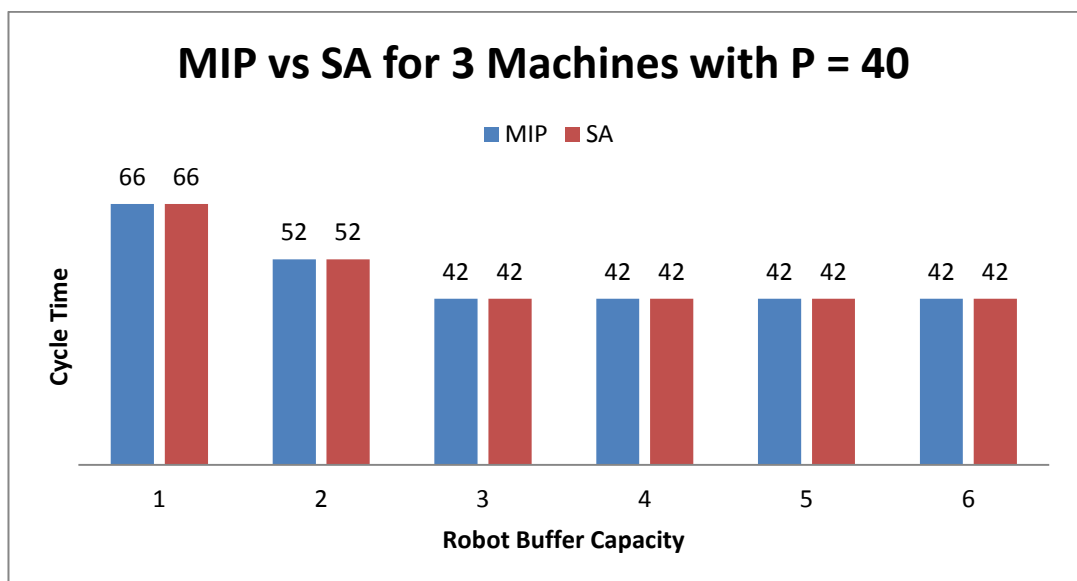


Figure 5.8: Comparison between MIP and SA Cycle Time for 3 Machines with P = 40

Since the optimality of the cycle times was proven for the case when process time was 22 and it can be seen that all the cycle times for 4 and 5 machine case are equivalent to that for process time equal to 40 thus it is sure that optimality was reached. However case of 5 machine and buffer capacity of size 6 seems to have a produced a cycle time of 48.

Table 5.20: Case when  $\delta = 2$ ,  $\varepsilon = 1$ ,  $P = 40$

MIP Model				Simulated Annealing Algorithm			
No. of machines (m)	Buffer Capacity (K)	Cycle Time (time unit)	Solution Time (s)	No. of machines (m)	Buffer Capacity (K)	Cycle Time (time unit)	Solution Time (s)
2	1	56	0.06	2	1	56	55
	2	42	0.39		2	42	37
	3	42	0.42		3	42	53
	4	42	0.39		4	42	59
3	1	66	0.28	3	1	66	65
	2	52	16.11		2	52	35
	3	42	73.36		3	42	50
	4	42	38.75		4	42	60
	5	42	64.59		5	42	64
	6	42	42.64		6	42	66
4	1	96	0.94	4	1	96	68
					2	68	92
					3	56	81
					4	52	117
					5	42	155
					6	42	169
					7	42	160
					8	42	165
5				5	1	140	213
					2	92	110
					3	72	158
					4	68	180
					5	60	174
					6	48	169
					7	44	185
					8	44	217
					9	44	221
					10	44	221

In order to prove that the case for buffer capacity 6 was not optimal, the optimal schedule shown in Table (5.21) that is generalized for this case is pretty simple since the robot can hold 6 parts that means 5 input buffer activities followed by 5 unloading/loading activities which is followed by 5 output buffer activities will give optimal solution. And since the unloading activity for each machine is exactly at the end of the cycle process time will not affect the schedule. And it can be seen that the cycle time found was indeed not the optimal solution.

The Gantt chart for this 5 machine case is shown in Figure (5.9), the processing of a part on each machine took 40 time units and the part stayed on the machine idle waiting for the robot for 2 time units and this was the case in all the 5 machines. Also, since the robot buffer has the ability to hold 6 parts that indicates that 5 output buffer activities can be followed by 5 input buffer activities.

Table 5.21: Case when  $m = 5$ ,  $\delta = 2$ ,  $\varepsilon = 1$ ,  $K = 6$ ,  $P = 40$

<b>a</b>	<b>L<sub>1</sub></b>	<b>U<sub>2</sub></b>	<b>L<sub>2</sub></b>	<b>U<sub>3</sub></b>	<b>L<sub>3</sub></b>	<b>U<sub>4</sub></b>	<b>L<sub>4</sub></b>	<b>U<sub>5</sub></b>	<b>L<sub>5</sub></b>	<b>O<sub>1</sub></b>	<b>O<sub>2</sub></b>	<b>O<sub>3</sub></b>	<b>O<sub>4</sub></b>	<b>O<sub>5</sub></b>	<b>I<sub>1</sub></b>	<b>I<sub>2</sub></b>	<b>I<sub>3</sub></b>	<b>I<sub>4</sub></b>	<b>I<sub>5</sub></b>	<b>U<sub>1</sub></b>	<b>C</b>
<b>t<sub>a</sub></b>	0	3	4	7	8	11	12	15	16	19	20	21	22	23	36	37	38	39	40	43	44
<b>m<sub>2</sub></b>	40	43	0	3	4	7	8	11	12	15	16	17	18	19	32	33	34	35	36	39	
<b>m<sub>3</sub></b>	36	39	40	43	0	3	4	7	8	11	12	13	14	15	28	29	30	31	32	35	
<b>m<sub>4</sub></b>	32	35	36	39	40	43	0	3	4	7	8	9	10	11	24	25	26	27	28	31	
<b>m<sub>5</sub></b>	28	31	32	35	36	39	40	43	0	3	4	5	6	7	20	21	22	23	24	27	
<b>Y<sub>a</sub><sup>+</sup></b>	1	2	2	3	3	4	4	5	5	4	3	2	1	0	0	0	0	0	0	1	
<b>Y<sub>a</sub><sup>-</sup></b>	4	4	3	3	2	2	1	1	0	0	0	0	0	0	1	2	3	4	5	5	



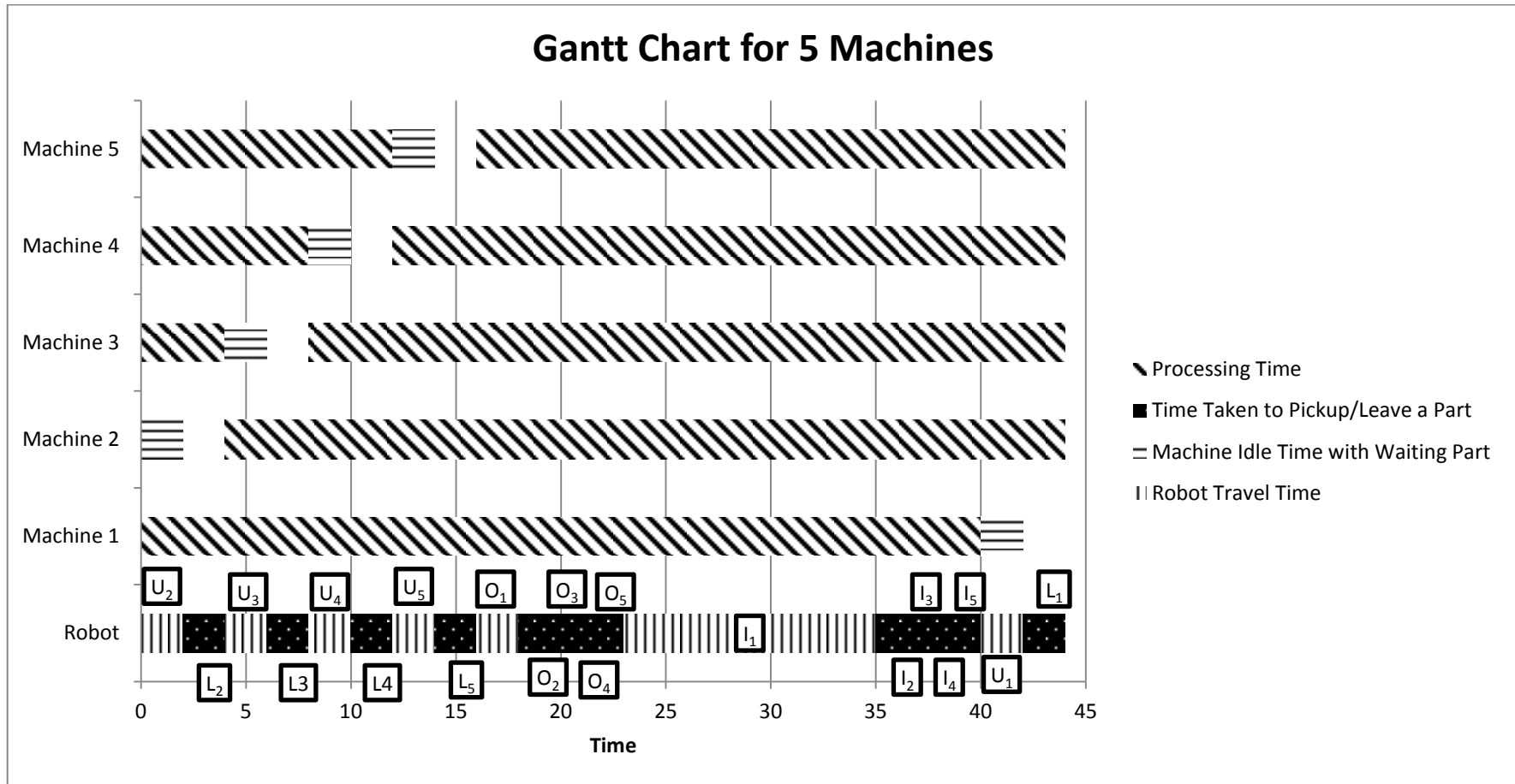


Figure 5.9: Gantt Chart for Case when  $m = 5$ ,  $\delta = 2$ ,  $\epsilon = 1$ ,  $K = 6$ ,  $P = 40$

### 5.3.4 Case 4: $\delta = 2$ , $\varepsilon = 1$ , $P = 50$

It can be seen that the optimal solution was found by the SA algorithm when comparing the cycle time of the 2 and 3 machine cases with that of the MIP model as shown in Figure (5.10) and (5.11) also with reduced time since most of the optimal solutions were actually found at times ranging between 0 and 1 s for the 2 machine case and times ranging between 1 and 24 s for the 3 machine case.

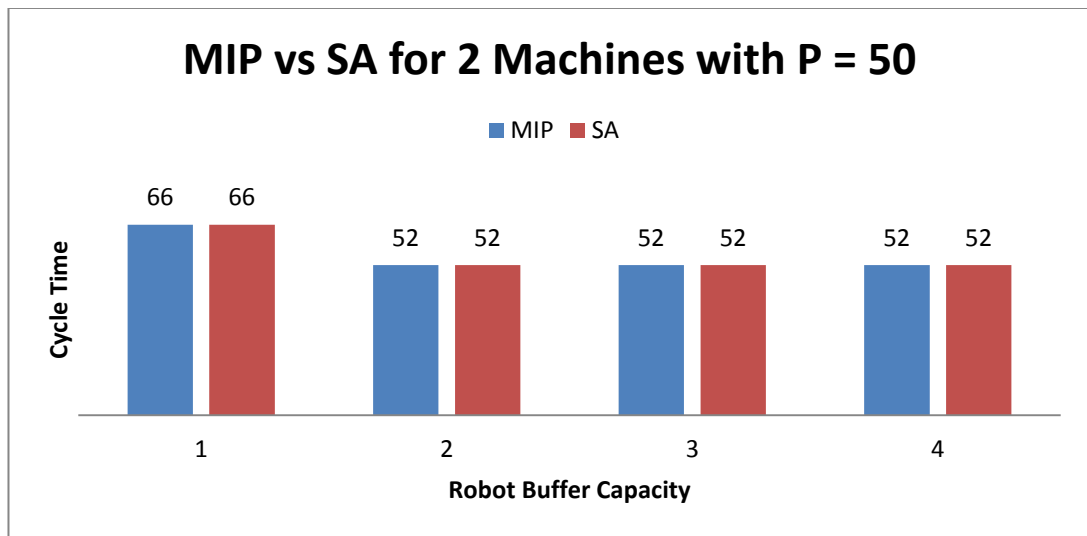


Figure 5.10: Comparison between MIP and SA Cycle Time for 2 Machines with  $P = 50$

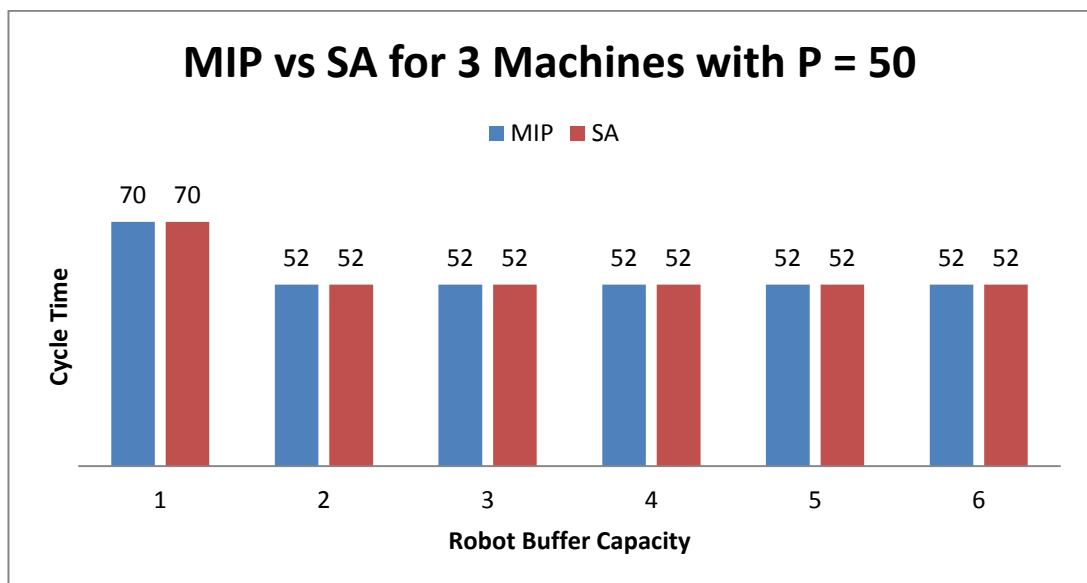


Figure 5.11: Comparison between MIP and SA Cycle Time for 3 Machines with  $P = 50$

Table 5.22: Case when  $\delta = 2, \varepsilon = 1, P = 50$

MIP Model				Simulated Annealing Algorithm			
No. of machines (m)	Buffer Capacity (K)	Cycle Time (time unit)	Solution Time (s)	No. of machines (m)	Buffer Capacity (K)	Cycle Time (time unit)	Solution Time (s)
2	1	66	0.13	2	1	66	54
	2	52	0.22		2	37	
	3	52	0.42		3	52	
	4	52	0.66		4	59	
3	1	70	0.28	3	1	70	66
	2	52	20.03		2	41	
	3	52	31.34		3	49	
	4	52	47.14		4	64	
	5	52	49.67		5	58	
	6	52	273.26		6	64	
4	1	96	0.75	4	1	96	171
					2	76	110
					3	56	54
					4	52	140
					5	52	161
					6	52	181
					7	52	170
					8	52	180
5				5	1	140	196
					2	92	111
					3	82	158
					4	68	183
					5	64	187
					6	52	196
					7	52	246
					8	52	261
					9	52	290
					10	52	271

As for the 4 and 5 machine cases when robot buffer capacity is greater than m it can be seen that the cycle time is 52 and this is in fact the optimal solution without using an optimal schedule as a proof since one of the constraints indicate that the completion time of unloading any machine must be greater than the completion time of loading that machine by at least the process time = 50 + the time taken to pick up the part which is equal to 1 and because the cycle time is equal to the completion

time of the last activity which is unloading and it is  $51 +$  the distance matrix from unloading machine  $i$  to loading machine  $i$  which is 1 the minimum cycle time that can be reached for process time 50 is in fact 52.

While other cycle times found were equivalent to the case when process time was 40, it was seen that when robot buffer capacity was 2 for 4 machine case and when robot buffer capacity was 3 and 5 for 5 machine case the cycle time differed from that of process time 40. Proving the optimality of these cases seemed to be extremely strenuous and thus their optimality could not be guaranteed.

### 5.3.5 Case 5: $\delta = 2, \varepsilon = 1, P = 100$

It can be seen that the optimal solution was found by the SA algorithm when comparing the cycle time of the 2 and 3 machine cases with that of the MIP model as shown in Figure (5.12) and (5.13), also with reduced time since most of the optimal solutions were actually found at times ranging between 0 and 5 s for the 2 machine case and times ranging between 0 and 14 s for the 3 machine case.

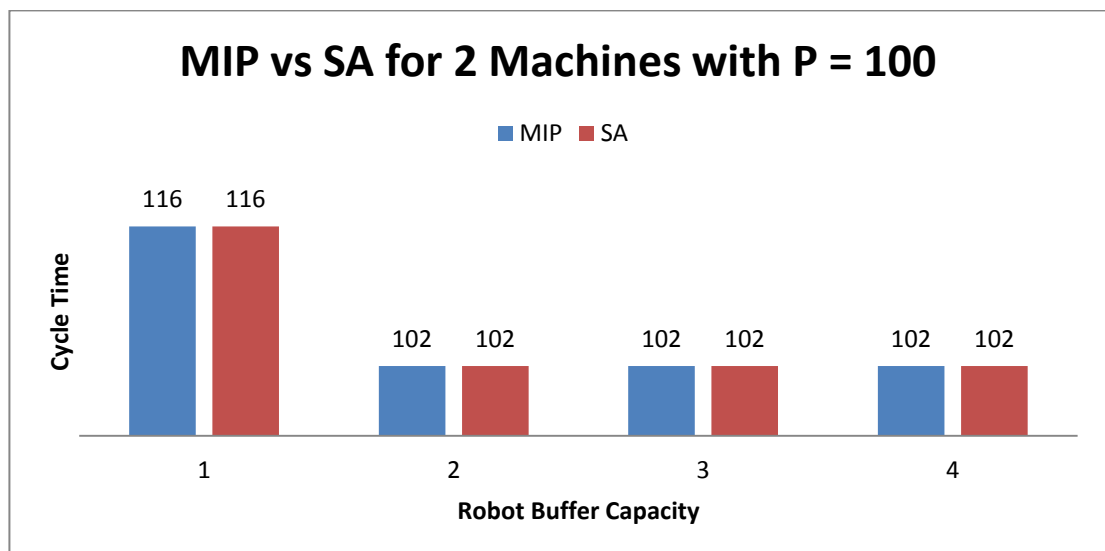


Figure 5.12: Comparison between MIP and SA Cycle Time for 2 Machines with P = 100

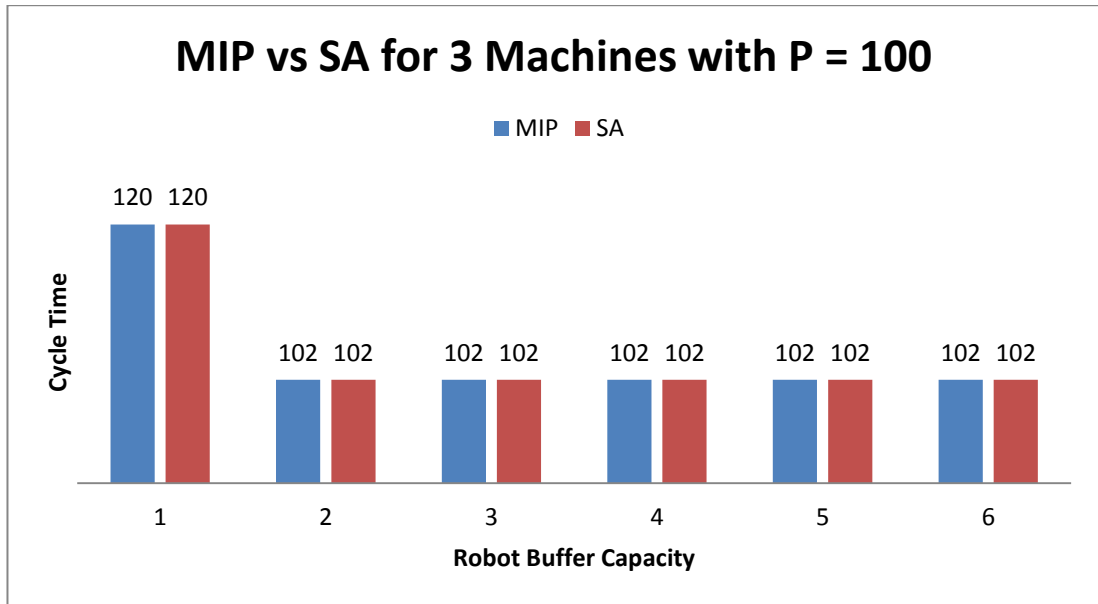


Figure 5.13: Comparison between MIP and SA Cycle Time for 3 Machines with P = 100

As for the 4 and 5 machine cases when robot buffer capacity is greater than 1 it can be seen that the cycle time is 102 and this is in fact the optimal solution without using an optimal schedule as a proof since one of the constraints indicate that the completion time of unloading any machine must be greater than the completion time of loading that machine by at least the process time =  $100 +$  the time taken to pick up a part which is equal to 1 and because the cycle time is equal to the completion time of the last activity which is unloading and it is  $101 +$  the distance matrix from unloading machine  $i$  to loading machine  $i$  which is 1 the minimum cycle time that can be reached for process time 100 is in fact 102.

Table 5.23: Case when  $\delta = 2$ ,  $\varepsilon = 1$ ,  $K = 2$ ,  $P = 100$

MIP Model				Simulated Annealing Algorithm			
No. of machines (m)	Buffer Capacity (K)	Cycle Time (time unit)	Solution Time (s)	No. of machines (m)	Buffer Capacity (K)	Cycle Time (time unit)	Solution Time (s)
2	1	116	0.13	2	1	116	56
	2	102	0.31		2	41	
	3	102	0.66		3	55	
	4	102	0.61		4	58	
3	1	120	0.27	3	1	120	63
	2	102	17.11		2	40	
	3	102	33.16		3	44	
	4	102	81.28		4	54	
	5	102	72.99		5	61	
	6	102	85.95		6	65	
4	1	124	0.61	4	1	124	182
					2	102	66
					3	102	117
					4	102	142
					5	102	168
					6	102	183
					7	102	179
					8	102	178
5				5	1	142	217
					2	120	132
					3	102	127
					4	102	145
					5	102	190
					6	102	186
					7	102	194
					8	102	207
					9	102	227
					10	102	199

The only cases that might not be optimal are 5 machine cases when robot buffer capacity is 1 and 2. In order to prove optimality of the case when robot buffer capacity is 1 it is evident that since the robot can only hold one part every unloading activity must be followed by output buffer activity which is followed by input buffer activity and finally loading activity and process time effect is considered as shown in Table (5.24). Thus, it is proven that 142 is not the optimal solution.

Table 5.24: Case when  $m = 5, \delta = 2, \varepsilon = 1, K = 1, P = 100$

<b>a</b>	<b>L<sub>1</sub></b>	<b>I<sub>5</sub></b>	<b>L<sub>5</sub></b>	<b>U<sub>2</sub></b>	<b>O<sub>2</sub></b>	<b>U<sub>4</sub></b>	<b>O<sub>4</sub></b>	<b>I<sub>2</sub></b>	<b>L<sub>2</sub></b>	<b>I<sub>4</sub></b>	<b>L<sub>4</sub></b>	<b>U<sub>3</sub></b>	<b>O<sub>3</sub></b>	<b>I<sub>3</sub></b>	<b>L<sub>3</sub></b>	<b>U<sub>1</sub></b>	<b>O<sub>1</sub></b>	<b>U<sub>5</sub></b>	<b>O<sub>5</sub></b>	<b>I<sub>1</sub></b>	<b>C</b>
<b>t<sub>a</sub></b>	0	3	14	21	30	35	40	53	58	63	72	75	82	95	102	107	118	121	124	137	140
<b>m<sub>2</sub></b>	82	85	96	103					0	5	14	17	24	37	44	49	60	63	6	79	
<b>m<sub>3</sub></b>	41	52	59	68	73	78	91	96	101	110	113				0	5	16	19	22	35	
<b>m<sub>4</sub></b>	68	71	82	89	98	103					0	3	10	23	30	35	46	49	52	65	
<b>m<sub>5</sub></b>			0	7	16	21	26	39	44	49	58	61	68	81	88	93	104	107			
<b>Y<sub>a</sub><sup>+</sup></b>	0	0	0	1	0	1	0	0	0	0	0	1	0	0	0	1	0	1	0	0	
<b>Y<sub>a</sub><sup>-</sup></b>	0	1	0	0	0	0	0	1	0	1	0	0	0	1	0	0	0	0	0	1	

Table 5.25: Case when  $m = 5, \delta = 2, \varepsilon = 1, K = 2, P = 100$

<b>a</b>	<b>L<sub>1</sub></b>	<b>I<sub>4</sub></b>	<b>O<sub>5</sub></b>	<b>U<sub>4</sub></b>	<b>L<sub>4</sub></b>	<b>O<sub>4</sub></b>	<b>I<sub>3</sub></b>	<b>U<sub>3</sub></b>	<b>L<sub>3</sub></b>	<b>O<sub>1</sub></b>	<b>I<sub>5</sub></b>	<b>U<sub>2</sub></b>	<b>L<sub>2</sub></b>	<b>I<sub>1</sub></b>	<b>O<sub>2</sub></b>	<b>U<sub>5</sub></b>	<b>L<sub>5</sub></b>	<b>O<sub>3</sub></b>	<b>I<sub>2</sub></b>	<b>U<sub>1</sub></b>	<b>C</b>
<b>t<sub>a</sub></b>	0	3	16	21	22	27	40	47	48	55	68	73	74	79	92	95	96	99	112	115	116
<b>m<sub>2</sub></b>	42	45	58	63	64	69	82	89	90	97	110	115	0	5	18	21	22	25	38	41	
<b>m<sub>3</sub></b>	68	71	84	89	90	95	108	115	0	7	20	25	26	31	44	47	48	51	64	67	
<b>m<sub>4</sub></b>	94	97	110	115	0	5	18	25	26	33	46	51	52	57	70	73	74	77	90	93	
<b>m<sub>5</sub></b>	20	23	36	41	42	47	60	67	68	75	88	93	94	99	112	115	0	3	16	19	
<b>Y<sub>a</sub><sup>+</sup></b>	1	1	0	1	1	0	0	1	1	0	0	1	1	1	0	1	1	0	0	1	
<b>Y<sub>a</sub><sup>-</sup></b>	0	1	1	1	0	0	1	1	0	0	1	1	0	1	1	1	0	0	1	1	



For the case when robot buffer capacity is 2 the schedule shown in Table (5.25) was found by using the optimal schedule when process time is 5000 and then if the cycle time found using that schedule was less than 120 that indicates that cycle time 120 is not optimal. It is seen that the cycle time found was 116 which is less than 120 thus 120 was not optimal solution.

### 5.3.6 Case 6: $\delta = 2$ , $\varepsilon = 1$ , $P = 5000$

It can be seen that the optimal solution was found by the SA algorithm when comparing the cycle time of the 2 and 3 machine cases with that of the MIP model as shown in Figure (5.14) and (5.15), also with reduced time since most of the optimal solutions were actually found at times ranging between 0 and 9 s for the 2 machine case and times ranging between 0 and 6 s for the 3 machine case even though iteration completion took a longer time.

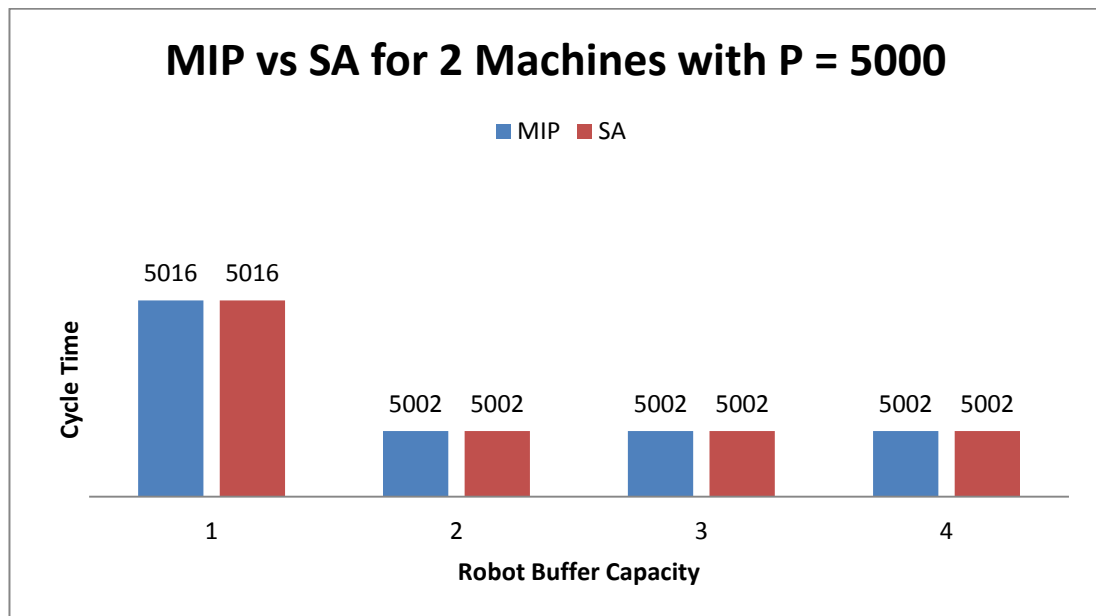


Figure 5.14: Comparison between MIP and SA Cycle Time for 2 Machines with P = 5000

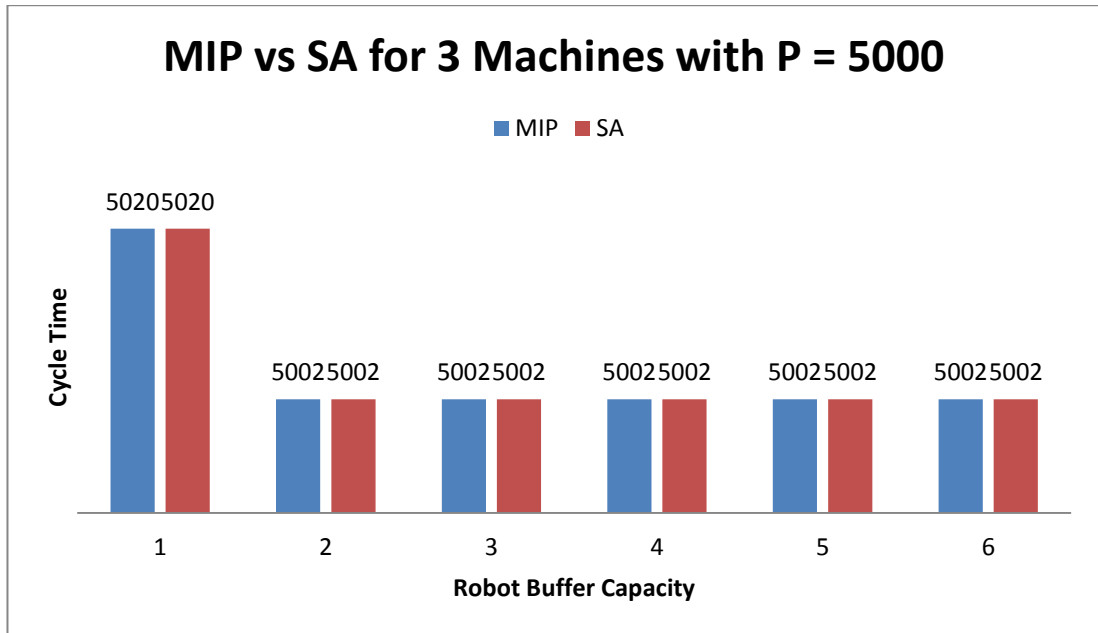


Figure 5.15: Comparison between MIP and SA Cycle Time for 3 Machines with P = 5000

As for the 4 and 5 machine cases when robot buffer capacity is greater than 1 it can be seen that the cycle time is 5002 and this is in fact the optimal solution without using an optimal schedule as a proof since one of the constraints indicate that the completion time of unloading any machine must be greater than the completion time of loading that machine by at least the process time = 5000 + the time taken to pick up a part which is equal to 1 and because the cycle time is equal to the completion time of the last activity which is unloading and it is 5001 + the distance matrix from unloading machine  $i$  to loading machine  $i$  which is 1 the minimum cycle time that can be reached for process time 5000 is in fact 5002.

Table 5.26: Case when  $\delta = 2$ ,  $\varepsilon = 1$ ,  $P = 5000$

MIP Model				Simulated Annealing Algorithm			
No. of machines (m)	Buffer Capacity (K)	Cycle Time (time unit)	Solution Time (s)	No. of machines (m)	Buffer Capacity (K)	Cycle Time (time unit)	Solution Time (s)
2	1	5016	1.36	2	1	5016	66
	2	5002	0.55		2	44	
	3	5002	0.76		3	60	
	4	5002	0.69		4	57	
3	1	5020	0.23	3	1	5020	73
	2	5002	18.14		2	38	
	3	5002	97.66		3	54	
	4	5002	140.69		4	64	
	5	5002	216.19		5	73	
	6	5002	259.91		6	83	
4	1	5024	0.67	4	1	5024	203
					2	5002	107
					3	5002	135
					4	5002	168
					5	5002	192
					6	5002	225
					7	5002	219
					8	5002	212
5				5	1	5038	304
					2	5002	131
					3	5002	170
					4	5002	205
					5	5002	250
					6	5002	276
					7	5002	282
					8	5002	296
					9	5002	172
					10	5002	265

The only cases that might not be optimal are 5 machine cases when robot buffer capacity is 1. In order to prove optimality of the case the same schedule used for process time 100 is applied in Table (5.27). This concludes that cycle time 5038 is in fact the optimal solution.

Table 5.27: Case when  $m = 5$ ,  $\delta = 2$ ,  $\varepsilon = 1$ ,  $K = 1$ ,  $P = 5000$

<b>a</b>	<b>L<sub>1</sub></b>	<b>I<sub>5</sub></b>	<b>L<sub>5</sub></b>	<b>U<sub>2</sub></b>	<b>O<sub>2</sub></b>	<b>U<sub>4</sub></b>	<b>O<sub>4</sub></b>	<b>I<sub>2</sub></b>	<b>L<sub>2</sub></b>	<b>I<sub>4</sub></b>	<b>L<sub>4</sub></b>	<b>U<sub>3</sub></b>	<b>O<sub>3</sub></b>	<b>I<sub>3</sub></b>	<b>L<sub>3</sub></b>	<b>U<sub>1</sub></b>	<b>O<sub>1</sub></b>	<b>U<sub>5</sub></b>	<b>O<sub>5</sub></b>	<b>I<sub>1</sub></b>	<b>C</b>
<b>t<sub>a</sub></b>	0	3	14	21	30	35	40	53	58	63	72	75	82	95	102	5005	5016	5020	5022	5035	5038
<b>m<sub>2</sub></b>	4980	4983	4994	5001					0	5	14	17	24	37	44	4947	4958	4961	4964	4977	
<b>m<sub>3</sub></b>	4939	4950	4957	4966	4971	4976	4989	4994	4999	5008	5017				0	4903	4914	4917	4920	4933	
<b>m<sub>4</sub></b>	4966	4969	4980	4987	4996	5001					0	3	10	23	30	4933	4944	4947	4950	4963	
<b>m<sub>5</sub></b>			0	7	16	21	26	39	44	49	58	61	68	81	88	4991	5002	5005			
<b>Y<sub>a</sub><sup>+</sup></b>	0	0	0	1	0	1	0	0	0	0	0	1	0	0	0	1	0	1	0	0	
<b>Y<sub>a</sub><sup>-</sup></b>	0	1	0	0	0	0	0	1	0	1	0	0	0	1	0	0	0	0	0	1	

## 5.4 Effect of Robot Buffer Capacity on Cycle Time

In this section, effect of robot buffer capacity on the cycle time is portrayed by Figure (5.16) and (5.18) for the 2 and 3 machine case with process times 0, 22, 40, 50 and 100. While the process time of 5000 is shown in Figure (5.17) and (5.19).

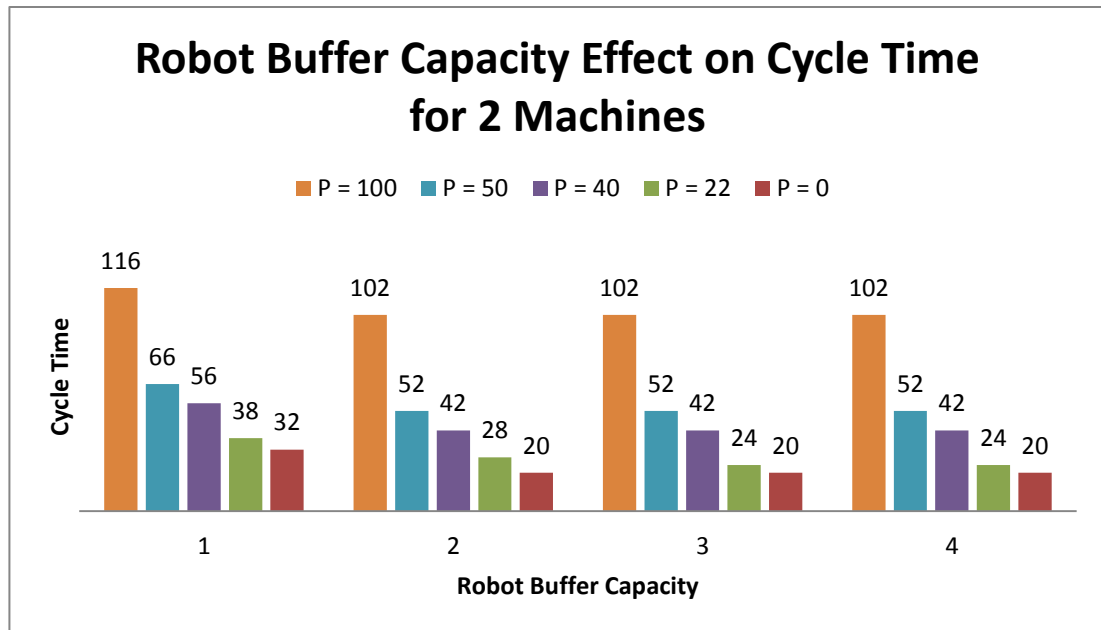


Figure 5.16: Robot Buffer Capacity Effect on Cycle Time for 2 Machines

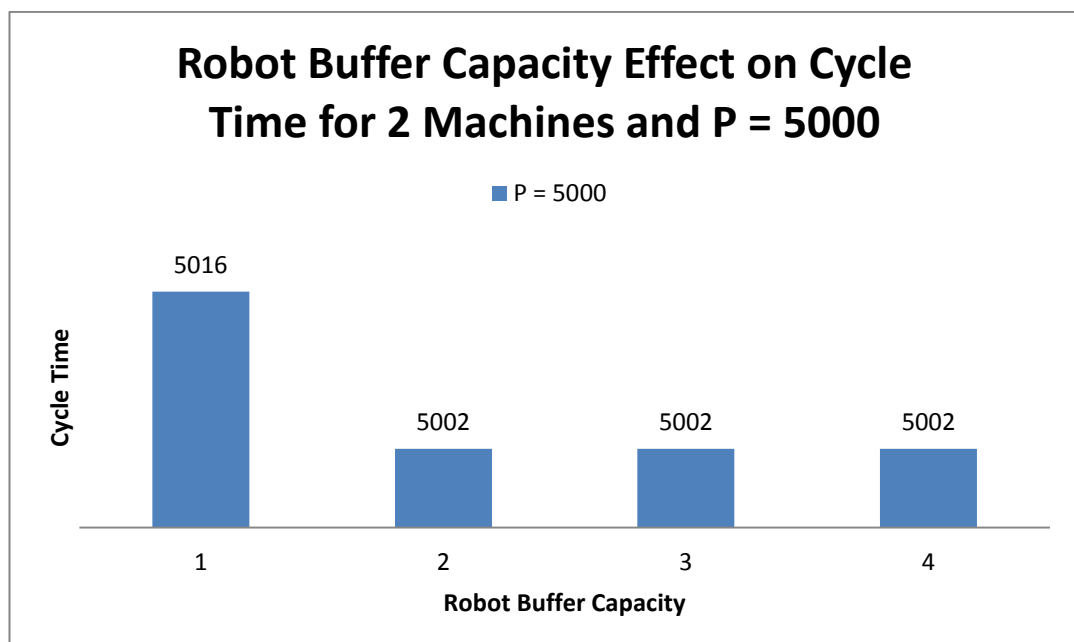


Figure 5.17: Robot Buffer Capacity Effect on Cycle Time for 2 Machines and P = 5000

It can be seen that for all process times reduction in cycle time was seen when robot buffer capacity is 2. However, robot buffer capacity above 2 showed no impact on cycle time reduction except for the case when process time was 22.

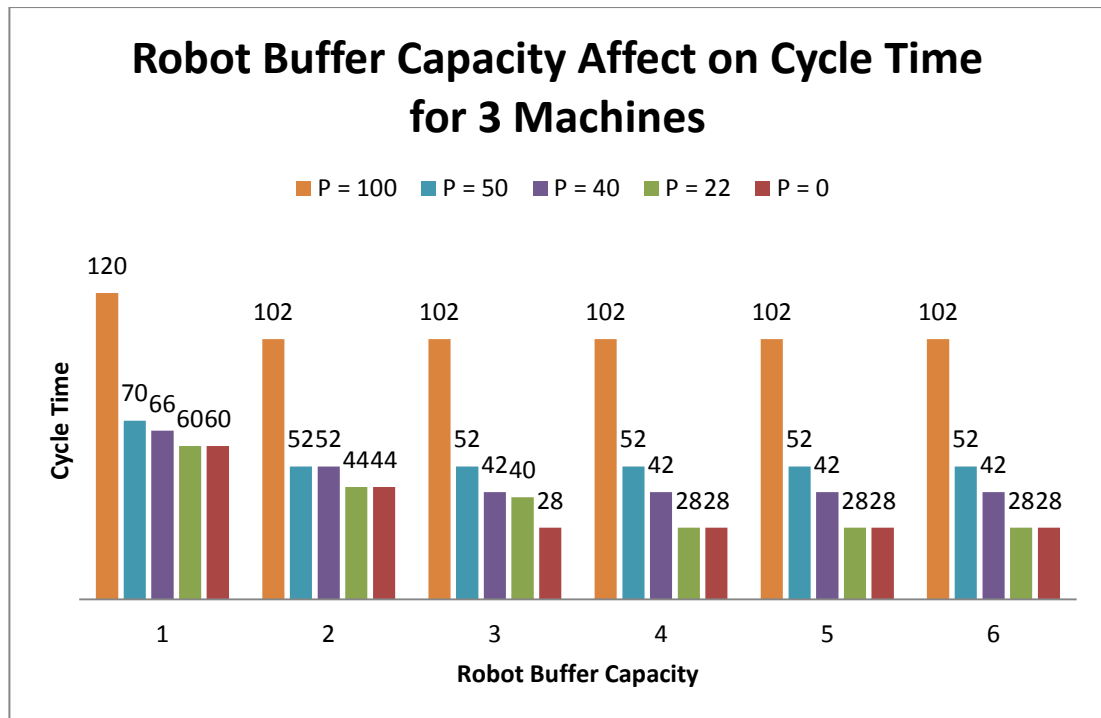


Figure 5.18: Robot Buffer Capacity Effect on Cycle Time for 3 Machines

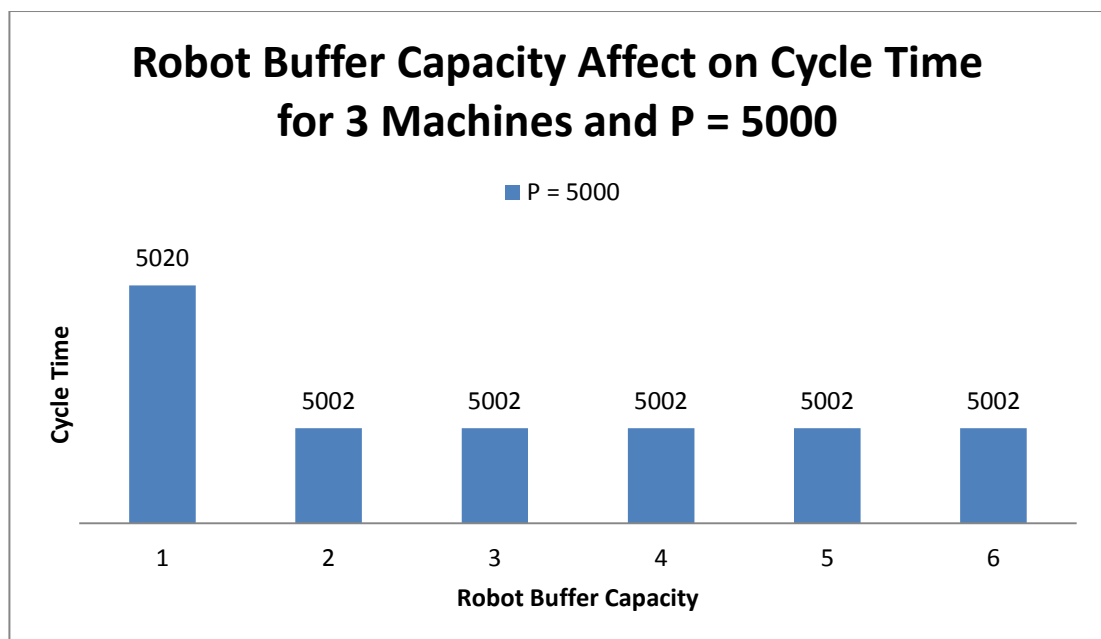


Figure 5.19: Robot Buffer Capacity Effect on Cycle Time for 3 Machines and P = 5000

It can be seen that for process times 50, 100 and 5000 reduction in cycle time was seen when robot buffer capacity is 2. However, robot buffer capacity above 2 showed no impact on cycle time reduction. But for process time 0 and 40 reduction in cycle time was also seen when robot buffer capacity was 3 and no impact was seen after that and this was true except for process time 22 where reduction in cycle time was also realized for robot buffer capacity of size 4.

Effect of robot buffer capacity on the cycle time is portrayed by Figure (5.20) and (5.22) for the 4 and 5 machine case with process times 0, 22, 40, 50 and 100. While the process time of 5000 is shown in Figure (5.21) and (5.23).

It can be seen that for 4 machine case process times 100 and 5000 reduction in cycle time was seen when robot buffer capacity is 2. However, robot buffer capacity above 2 showed no impact on cycle time reduction. But for process time 0 and 50 reduction in cycle time was also seen when robot buffer capacity was 3 and 4 and no impact was seen after that and this was true except for process time 22 and 40 where reduction in cycle time was also realized for robot buffer capacity of size 5.

And for 5 machine case when process times were 100 and 5000 reduction in cycle time was seen when robot buffer capacity is 2. However, robot buffer capacity above 2 showed no impact on cycle time reduction. But for process time 22, 40 and 50 reduction in cycle time was also seen when robot buffer capacity was 3, 4, 5 and 6 and no impact was seen after that and this was true except for process time 0 where reduction in cycle time was realized until buffer capacity of size 5.

## Robot Buffer Capacity Affect on Cycle Time for 4 Machines

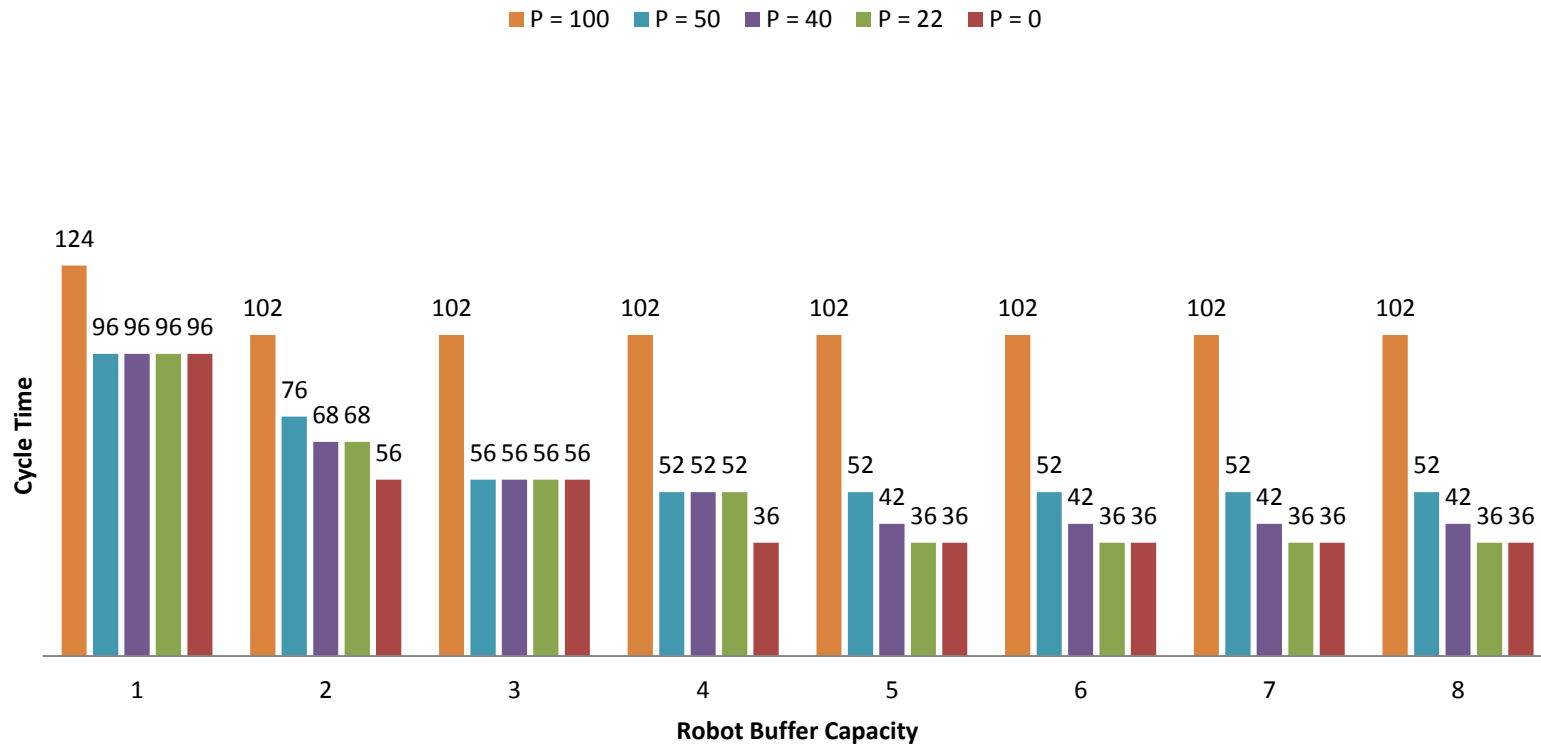


Figure 5.20: Robot Buffer Capacity Effect on Cycle Time for 4 Machines



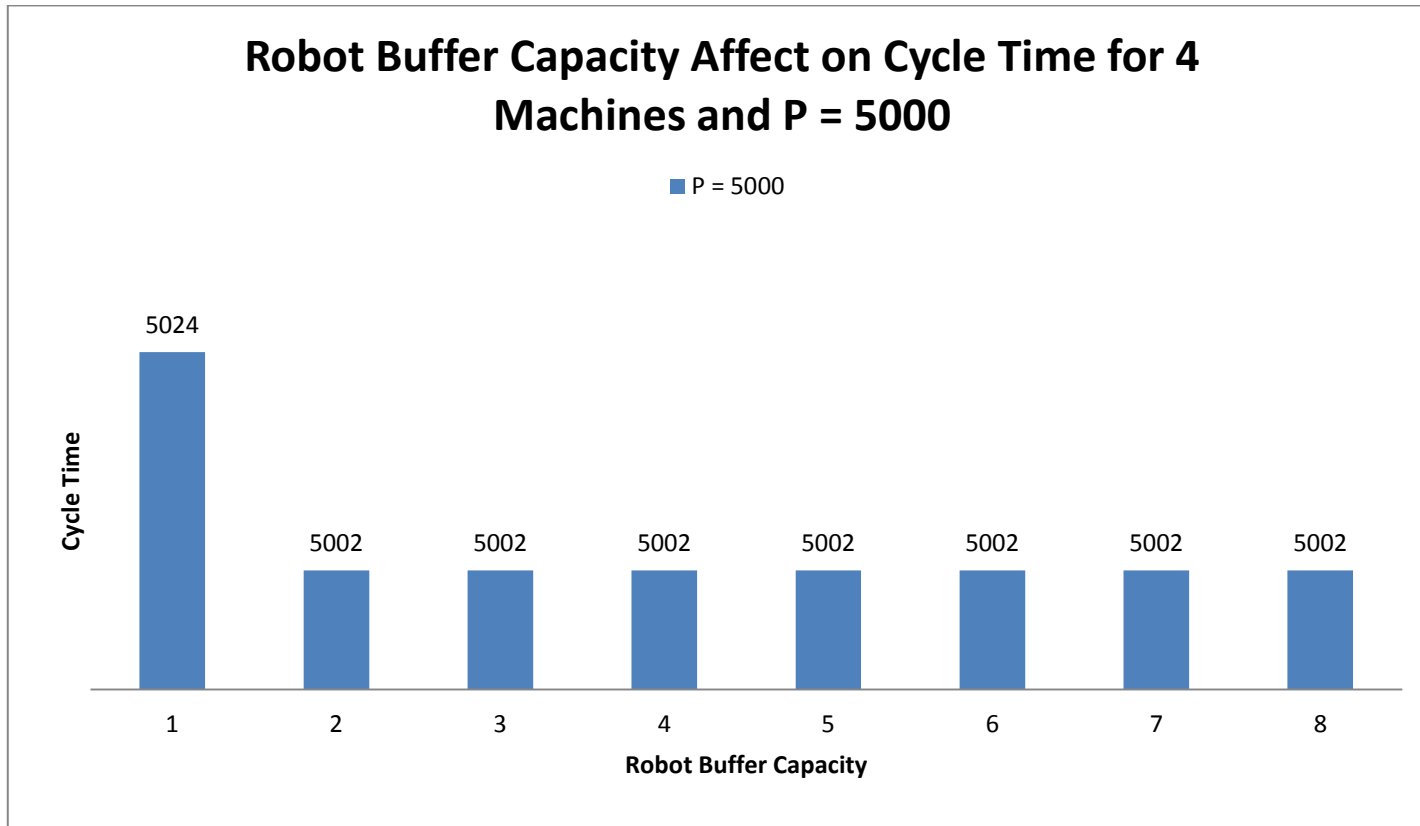


Figure 5.21: Robot Buffer Capacity Effect on Cycle Time for 4 Machines and P = 5000

## Robot Buffer Capacity Affect on Cycle Time for 5 Machines

■ P = 100 ■ P = 50 ■ P = 40 ■ P = 22 ■ P = 0

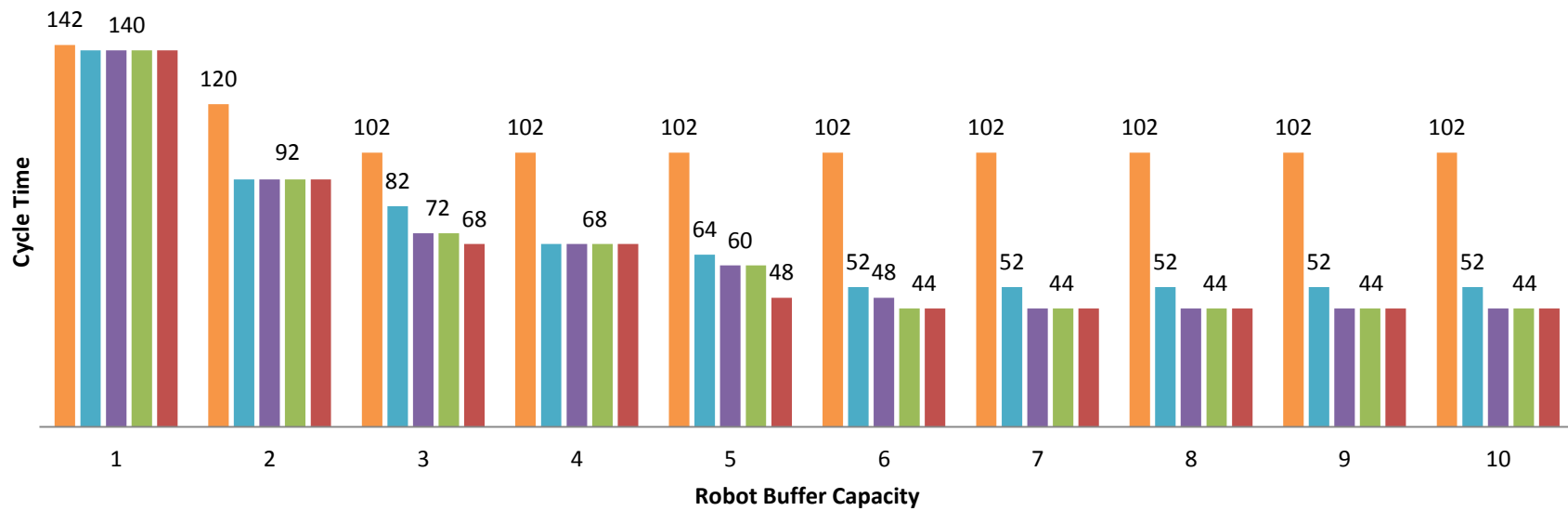


Figure 5.22: Robot Buffer Capacity Effect on Cycle Time for 5 Machines

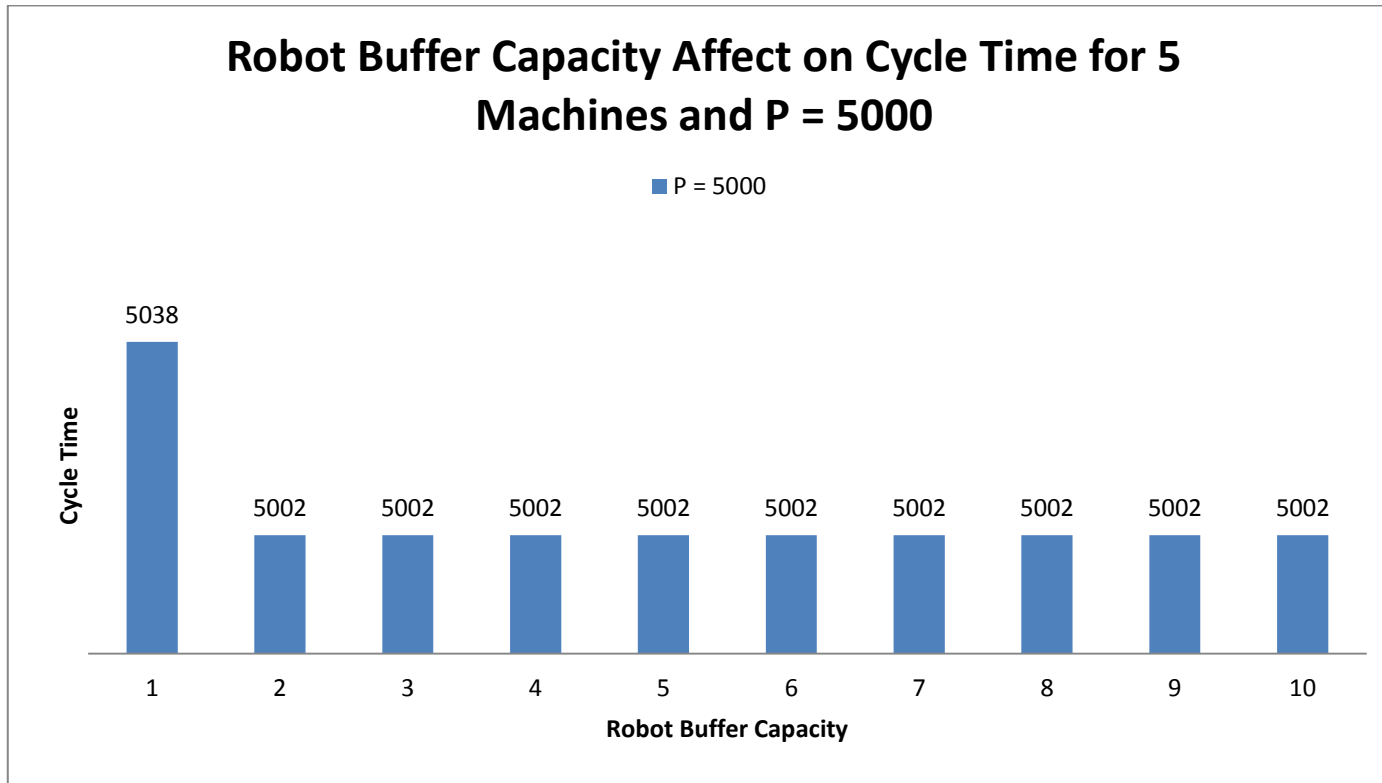


Figure 5.23: Robot Buffer Capacity Effect on Cycle Time for 5 Machines and P = 5000

## 5.5 Cycle Time is same for Various Process Times

In this section the impact of process time on the cycle time is discussed and why cycle time can be constant for various process times. As an example, we will take the optimal schedule found for the 3 machine case with  $K = 1$  but with process times 0 and 22 as shown in Table (5.28) and (5.29).

The distance matrix  $d_{ab}$  is not a function of process time and thus the process time does not affect the cycle time of a robot unless there is waiting time which causes the robot to wait until the machine finishes processing which leads to increase in cycle time. It can be seen in the case when process time was 0 and 22 that between loading and unloading any of machine 1, 2 or

3 that by the time the unloading activity was reached processing of the part on that machine was finished and thus there was no waiting time and because there was no waiting time, the effect of process time was none and the cycle time for both process times was equal.

## 5.6 Cycle Time Increase with Increase in Process Time

On the other hand, in the case where process times were 40, 50, 100 and 5000 it was seen that there was some waiting time because the cumulative  $d_{ab}$  plus completion time of last activity was less than the process time meaning that the robot had to wait until processing was completed and that waiting time was the cause of increase in cycle time and these cases are shown in Tables (5.30), (5.31), (5.32) and (5.33).

Table 5.28: Case when  $m = 3, \delta = 2, \varepsilon = 1, K = 1, P = 0$

<b>a</b>	<b>L<sub>1</sub></b>	<b>I<sub>2</sub></b>	<b>L<sub>2</sub></b>	<b>I<sub>3</sub></b>	<b>L<sub>3</sub></b>	<b>U<sub>1</sub></b>	<b>O<sub>1</sub></b>	<b>U<sub>2</sub></b>	<b>O<sub>2</sub></b>	<b>U<sub>3</sub></b>	<b>O<sub>3</sub></b>	<b>I<sub>1</sub></b>	<b>C</b>
<b>d<sub>ab</sub></b>	0	2 + 1	4 + 1	4 + 1	6 + 1	4 + 1	6 + 1	4 + 1	4 + 1	2 + 1	2 + 1	8 + 1	2 + 1
<b>t<sub>a</sub></b>	0	3	8	13	20	25	32	37	42	45	48	57	60
<b>m<sub>2</sub></b>			0	5	12	17	24	29					
<b>m<sub>3</sub></b>					0	5	12	17	22	25			

Table 5.29: Case when  $m = 3, \delta = 2, \varepsilon = 1, K = 1, P = 22$

<b>a</b>	<b>L<sub>1</sub></b>	<b>I<sub>2</sub></b>	<b>L<sub>2</sub></b>	<b>I<sub>3</sub></b>	<b>L<sub>3</sub></b>	<b>U<sub>1</sub></b>	<b>O<sub>1</sub></b>	<b>U<sub>2</sub></b>	<b>O<sub>2</sub></b>	<b>U<sub>3</sub></b>	<b>O<sub>3</sub></b>	<b>I<sub>1</sub></b>	<b>C</b>
<b>d<sub>ab</sub></b>	0	2 + 1	4 + 1	4 + 1	6 + 1	4 + 1	6 + 1	4 + 1	4 + 1	2 + 1	2 + 1	8 + 1	2 + 1
<b>t<sub>a</sub></b>	0	3	8	13	20	25	32	37	42	45	48	57	60
<b>m<sub>2</sub></b>			0	5	12	17	24	29					
<b>m<sub>3</sub></b>					0	5	12	17	22	25			

Table 5.30: Case when  $m = 3, \delta = 2, \varepsilon = 1, K = 1, P = 40$

<b>a</b>	<b>L<sub>1</sub></b>	<b>I<sub>3</sub></b>	<b>L<sub>3</sub></b>	<b>U<sub>2</sub></b>	<b>O<sub>2</sub></b>	<b>I<sub>2</sub></b>	<b>L<sub>2</sub></b>	<b>U<sub>1</sub></b>	<b>O<sub>1</sub></b>	<b>U<sub>3</sub></b>	<b>O<sub>3</sub></b>	<b>I<sub>1</sub></b>	<b>C</b>
<b>d<sub>ab</sub></b>	0	2 + 1	6 + 1	2 + 1	4 + 1	8 + 1	4 + 1	2 + 1	6 + 1	2 + 1	2 + 1	8 + 1	2 + 1
<b>w<sub>i</sub></b>								6					
<b>t<sub>a</sub></b>	0	3	10	13	18	27	32	41	48	51	54	63	66
<b>m<sub>2</sub></b>	34	37	44	47			0	9	16	19	22	31	
<b>m<sub>3</sub></b>			0	3	8	17	22	31	38	41			

Table 5.31: Case when  $m = 3, \delta = 2, \varepsilon = 1, K = 1, P = 50$

<b>a</b>	<b>L<sub>1</sub></b>	<b>U<sub>2</sub></b>	<b>O<sub>2</sub></b>	<b>I<sub>2</sub></b>	<b>L<sub>2</sub></b>	<b>U<sub>3</sub></b>	<b>O<sub>3</sub></b>	<b>I<sub>3</sub></b>	<b>L<sub>3</sub></b>	<b>U<sub>1</sub></b>	<b>O<sub>1</sub></b>	<b>I<sub>1</sub></b>	<b>C</b>
<b>d<sub>ab</sub></b>	0	2 + 1	4 + 1	8 + 1	4 + 1	2 + 1	2 + 1	8 + 1	6 + 1	4 + 1	6 + 1	8 + 1	2 + 1
<b>w<sub>i</sub></b>										2			
<b>t<sub>a</sub></b>	0	3	8	17	22	25	28	37	44	51	58	67	70
<b>m<sub>2</sub></b>	48	51			0	3	6	15	22	29	36	45	
<b>m<sub>3</sub></b>	26	29	34	43	48	51			0	7	14	23	

Table 5.32: Case when  $m = 3, \delta = 2, \varepsilon = 1, K = 1, P = 100$

<b>a</b>	<b>L<sub>1</sub></b>	<b>U<sub>3</sub></b>	<b>O<sub>3</sub></b>	<b>I<sub>3</sub></b>	<b>L<sub>3</sub></b>	<b>U<sub>2</sub></b>	<b>O<sub>2</sub></b>	<b>I<sub>2</sub></b>	<b>L<sub>2</sub></b>	<b>U<sub>1</sub></b>	<b>O<sub>1</sub></b>	<b>I<sub>1</sub></b>	<b>C</b>
<b>d<sub>ab</sub></b>	0	4 + 1	2 + 1	8 + 1	6 + 1	2 + 1	4 + 1	8 + 1	4 + 1	2 + 1	6 + 1	8 + 1	2 + 1
<b>w<sub>i</sub></b>										52			
<b>t<sub>a</sub></b>	0	5	8	17	24	27	32	41	46	101	108	117	120
<b>m<sub>2</sub></b>	74	79	82	91	98	101			0	55	62	71	
<b>m<sub>3</sub></b>	96	101			0	3	8	17	22	77	84	93	

Table 5.33: Case when  $m = 3, \delta = 2, \varepsilon = 1, K = 1, P = 5000$

<b>a</b>	<b>L<sub>1</sub></b>	<b>U<sub>2</sub></b>	<b>O<sub>2</sub></b>	<b>I<sub>2</sub></b>	<b>L<sub>2</sub></b>	<b>U<sub>3</sub></b>	<b>O<sub>3</sub></b>	<b>I<sub>3</sub></b>	<b>L<sub>3</sub></b>	<b>U<sub>1</sub></b>	<b>O<sub>1</sub></b>	<b>I<sub>1</sub></b>	<b>C</b>
<b>d<sub>ab</sub></b>	0	2 + 1	4 + 1	8 + 1	4 + 1	2 + 1	2 + 1	8 + 1	6 + 1	4 + 1	6 + 1	8 + 1	2 + 1
<b>w<sub>i</sub></b>										4952			
<b>t<sub>a</sub></b>	0	3	8	17	22	25	28	37	44	5001	5008	5017	5020
<b>m<sub>2</sub></b>	4998	5001			0	3	6	15	22	4979	4986	4995	
<b>m<sub>3</sub></b>	4976	4979	4984	4993	5001				0	4957	4964	4973	

## **5.7 Cycle Time after Robot Buffer Capacity > 1 is Constant for Large Process Times**

Another important realization is that as the process time increased the optimal cycle time remained constant after robot buffer capacity greater than 1. For example in the case when process time is 40 it can be seen that this was true only for the 2 machine case and in the case when process time is 50 that was true for both 2 and 3 machine case. However, in process time 100 and 5000 this was true for 2, 3, 4 and 5 machine cases. So another realization is that with increase in process time the cycle time stays constant after robot buffer capacity 1 for increasing number of machines. Table (5.34) shows the relation of process time, number of machines and robot buffer capacity.

This is mainly because with increase process time in order to avoid waiting time the schedule for all the cases is in such way that the loading and unloading activity of any machine is placed further apart and thus the maximum cumulative distance matrix + completion time is surpassed which leads to the same cycle time for all cases.

It must be noted that this indicates that with increased process time the impact of the robot buffer capacity is negligible. However, it can be seen that there will always be a difference between a single gripper and dual gripper robot since a dual gripper robot in our study is represented by a single gripper robot with a buffer capacity of size 2.



Table 5.34: Relation of Process Times, Number of Machines and Robot Buffer Capacity

No. of machines (m)	Buffer Capacity (K)	Cycle Time (time unit)			
		Process Time (40)	Process Time (50)	Process Time (100)	Process Time (5000)
2	1	56	66	116	5016
	2	42	52	102	5002
	3	42	52	102	5002
	4	42	52	102	5002
3	1	66	70	120	5020
	2	52	52	102	5002
	3	42	52	102	5002
	4	42	52	102	5002
	5	42	52	102	5002
	6	42	52	102	5002
4	1	96	96	124	5024
	2	68	76	102	5002
	3	56	56	102	5002
	4	52	52	102	5002
	5	42	52	102	5002
	6	42	52	102	5002
	7	42	52	102	5002
	8	42	52	102	5002
5	1	140	140	142	5038
	2	92	92	120	5002
	3	72	82	102	5002
	4	68	68	102	5002
	5	60	64	102	5002
	6	48	52	102	5002
	7	44	52	102	5002
	8	44	52	102	5002
	9	44	52	102	5002
	10	44	52	102	5002

### 5.8 Waiting Time is considered for One Machine

In this section it can be shown that if waiting time is considered for one machine than no other machine will have waiting time as long as all the loading activities of other machines come before the unloading activity of the machine with waiting time. As an example the case with 2 and 3 machines for process time 50 and 100 and robot buffer capacity 2 are shown in Tables (5.35), (5.36), (5.37) and (5.38)

Table 5.35: Case when  $m = 2, \delta = 2, \varepsilon = 1, K = 2, P = 50$

Activity (a)	L <sub>1</sub>	I <sub>1</sub>	O <sub>1</sub>	U <sub>2</sub>	L <sub>2</sub>	I <sub>2</sub>	O <sub>2</sub>	U <sub>1</sub>	C
<b>d<sub>ab</sub></b>	0	2 + 1	6 + 1	2 + 1	1	4 + 1	6 + 1	2 + 1	1
<b>w<sub>i</sub></b>								22	
<b>t<sub>a</sub></b>	0	3	10	13	14	19	26	51	52
<b>m<sub>2</sub></b>	38	41	48	51	0	5	12	37	

Table 5.36: Case when  $m = 2, \delta = 2, \varepsilon = 1, K = 2, P = 100$

Activity (a)	L <sub>1</sub>	O <sub>1</sub>	I <sub>1</sub>	U <sub>2</sub>	L <sub>2</sub>	O <sub>2</sub>	I <sub>2</sub>	U <sub>1</sub>	C
<b>d<sub>ab</sub></b>	0	4 + 1	6 + 1	4 + 1	1	2 + 1	6 + 1	2 + 1	1
<b>w<sub>i</sub></b>								70	
<b>t<sub>a</sub></b>	0	5	12	17	18	21	28	101	102
<b>m<sub>2</sub></b>	84	89	96	101	0	3	10	83	

Thus, it can be seen that since the waiting time was considered during unloading of machine 1 and all the loading of activities were before the unloading activity of machine 1 then the waiting time is considered only once.

On the other hand, if waiting time was considered for a machine and some of the loading activities of other machines came after it may be noticed that waiting time could be considered twice.

From these optimal schedules we can also note that with increasing process time the waiting time is increased which in turn leads to increased cycle time

Table 5.37: Case when  $m = 3, \delta = 2, \varepsilon = 1, K = 2, P = 50$

<b>a</b>	<b>L<sub>1</sub></b>	<b>I<sub>2</sub></b>	<b>O<sub>3</sub></b>	<b>U<sub>2</sub></b>	<b>L<sub>2</sub></b>	<b>I<sub>3</sub></b>	<b>O<sub>2</sub></b>	<b>U<sub>3</sub></b>	<b>L<sub>3</sub></b>	<b>O<sub>1</sub></b>	<b>I<sub>1</sub></b>	<b>U<sub>1</sub></b>	<b>C</b>
<b>d<sub>ab</sub></b>	0	2 + 1	8 + 1	4 + 1	1	4 + 1	8 + 1	2 + 1	1	2 + 1	8 + 1	2 + 1	1
<b>w<sub>i</sub></b>													
<b>t<sub>a</sub></b>	0	3	12	17	18	23	32	35	36	39	48	51	52
<b>m<sub>2</sub></b>	34	37	46	51	0	5	14	17	18	21	30	33	
<b>m<sub>3</sub></b>	16	19	28	33	34	39	48	51	0	3	12	15	

Table 5.38: Case when  $m = 3, \delta = 2, \varepsilon = 1, K = 2, P = 100$

<b>a</b>	<b>L<sub>1</sub></b>	<b>O<sub>3</sub></b>	<b>I<sub>3</sub></b>	<b>U<sub>2</sub></b>	<b>L<sub>2</sub></b>	<b>I<sub>1</sub></b>	<b>O<sub>2</sub></b>	<b>U<sub>3</sub></b>	<b>L<sub>3</sub></b>	<b>O<sub>1</sub></b>	<b>I<sub>2</sub></b>	<b>U<sub>1</sub></b>	<b>C</b>
<b>d<sub>ab</sub></b>	0	6 + 1	8 + 1	4 + 1	1	4 + 1	8 + 1	2 + 1	1	2 + 1	8 + 1	2 + 1	1
<b>w<sub>i</sub></b>												46	
<b>t<sub>a</sub></b>	0	7	16	21	22	27	36	39	40	43	52	101	102
<b>m<sub>2</sub></b>	80	87	96	101	0	5	14	17	18	21	30	79	
<b>m<sub>3</sub></b>	62	69	78	83	84	89	98	101	0	3	12	61	

The Gantt chart for the 3 machine case with process time 50 and robot buffer capacity of size 2 is shown in Figure (5.24). In this chart it can be seen that the processing of a part for each machine was 50 time units with 2 time units' idle time with no part on it. And in this case unloading activity of a machine is followed by loading activity and thus there is no travel time just pick up or leaving time for a part since the robot buffer can hold 2 parts reduction in cycle time was realized.

These charts indicate that the processing of a part does not go through all machines and that is because those machines are in different stages thus it is a parallel machine flowshop with one machine in each stage.

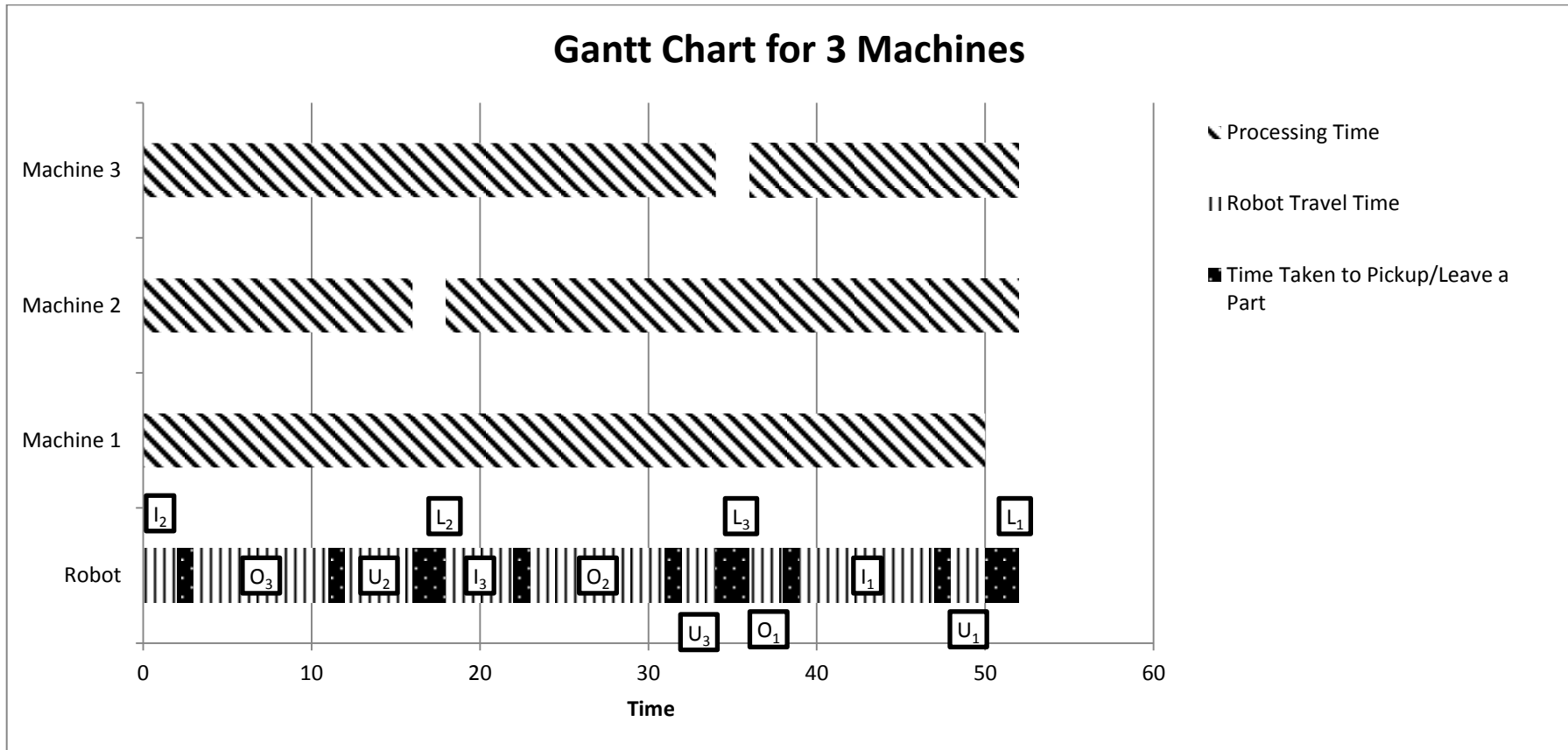


Figure 5.24: Gantt Chart for Case when  $m = 5$ ,  $\delta = 2$ ,  $\varepsilon = 1$ ,  $K = 6$ ,  $P = 40$

## Chapter 6

### CONCLUSION AND FUTURE RECOMMENDATIONS

The main objective of this study was to analyze how increase in robot buffer capacity impacted the cycle time by scheduling the robot moves in a manner by which the cycle time was minimized for a flexible robotic cell with a single self-buffered robot and single gripper for an  $m$  machine case in an inline layout where identical parts are produced and 1-unit is produced in each cycle.

The contribution of the study was twofold. The first contribution was formulating a general scheduling model for an  $m$  machine case for the self-buffered robot which has not been implemented before and the second contribution was comparing the optimal cycle time found by the mixed integer programming model to that found by the simulated annealing algorithm for the same problem.

#### 6.1 Conclusions

1. Simulated Annealing algorithm produced optimal solutions 94% of the time when compared to the optimal solutions found by the MIP model since out of a total of 168 runs made, optimality was not guaranteed for only 10 of those cases.
2. Solution time was reduced while applying the SA algorithm since the time taken to find the solution was much less when compared to the time taken by the MIP model.

3. SA algorithm can be applied in future studies since even though optimality is not guaranteed, it can be found more often than not.
4. Robot buffer capacity does reduce the cycle time which indicates that the performance of the system can be further improved and this is an advantage that can be used by production managers in several industries.
5. Effect of robot buffer capacity diminishes with increase in process time since cycle time remains constant after robot buffer capacity more than 1. This is due to the fact that minimum cycle time is reached earlier with larger process times.
6. Since there was always a change realized in cycle time between robot buffer capacity 1 and 2 that means there will always be significant difference between a single and dual gripper robot if we consider a dual gripper robot to represent a single self-buffered robot with robot buffer capacity of size 2.
7. Increase in process time leads to increase in waiting time which in turn leads to increase in cycle time. However, this is not always true since sometimes the waiting time is less with increased process time even though cycle time is increased and that is because the optimal schedule differs with different process time and the effect of those schedules is also taken into account.
8. Cycle time is same for various process times when the difference between process times is small and process time has no effect on cycle time calculation.
9. Waiting time is only considered for one machine if and only if all the loading activities of other machines come before the unloading activity of the machine with waiting time.

## 6.2 Future Recommendations

1. Robotic cell with a self-buffered single gripper robot and a circular layout since this study and the previous study that included self-buffered robot proposed employing an inline layout. So impact of the circular layout on the performance of the FMS system with a self-buffered robot can be studied.
2. Robotic cell with a self-buffered dual arm robot. The study that previously included a self-buffered robot did a comparison between single gripper self-buffered robot and a bufferless dual gripper robot. However, comparison to a dual arm self-buffered robot was not employed.
3. Comparison of inline and circular layout for a self-buffered single gripper robot. To study whether the performance of an FMS system with a self-buffered robot is further improved for a system with a circular layout.
4. Proof of generalized cyclic schedules that lead to optimal solutions for self-buffered single gripper robot. Because if generalized schedules were created than optimality of an  $m$  machine case can be easily proved for solutions found by the simulated annealing algorithm.
5. Proof of single waiting time for self-buffered single gripper robot. The question here will be if all the loading activities of machines come after the unloading activity of the machine with waiting time is it in fact true that waiting time will only be considered once?



## REFERENCES

- [1] Crama, Y., & Van De Klundert, J. (1997). Cyclic scheduling of identical parts in a robotic cell. *Operations Research*, 45(6), 952-965.
- [2] Crama, Y., & Van de Klundert, J. (1997). Robotic flowshop scheduling is strongly NP-complete.
- [3] Levner, E., Kats, V., & Levit, V. E. (1997). An improved algorithm for cyclic flowshop scheduling in a robotic cell. *European Journal of Operational Research*, 97(3), 500-508.
- [4] Hall, N. G., Kamoun, H., & Sriskandarajah, C. (1998). Scheduling in robotic cells: Complexity and steady state analysis. *European Journal of Operational Research*, 109(1), 43-65.
- [5] Levner, E., & Kats, V. (1998). A parametric critical path problem and an application for cyclic scheduling. *Discrete Applied Mathematics*, 87(1-3), 149-158.
- [6] Sriskandarajah, C., Hall, N. G., & Kamoun, H. (1998). Scheduling large robotic cells without buffers. *Annals of Operations Research*, 76, 287-321.
- [7] Crama, Y., & Van de Klundert, J. (1999). Cyclic scheduling in 3-machine robotic flow shops. *Journal of Scheduling*, 2, 35-54.

- [8] Kamoun, H., Hall, N. G., & Sriskandarajah, C. (1999). Scheduling in robotic cells: Heuristics and cell design. *Operations Research*, 47(6), 821-835.
- [9] Brauner, N., & Finke, G. (1999). On a conjecture about robotic cells: new simplified proof for the three-machine case. *INFOR: Information Systems and Operational Research*, 37(1), 20-36.
- [10] Agnetis, A., & Pacciarelli, D. (2000). Part sequencing in three-machine no-wait robotic cells. *Operations Research Letters*, 27(4), 185-192.
- [11] Agnetis, A. (2000). Scheduling no-wait robotic cells with two and three machines. *European Journal of Operational Research*, 123(2), 303-314.
- [12] Brauner, N., & Finke, G. (2001). Cycles and permutations in robotic cells. *Mathematical and Computer Modelling*, 34(5-6), 565-591.
- [13] Sethi, S. P., Sidney, J. B., & Sriskandarajah, C. (2001). Scheduling in dual gripper robotic cells for productivity gains. *IEEE Transactions on Robotics and Automation*, 17(3), 324-341.
- [14] Brauner, N., Finke, G., & Kubiak, W. (2003). Complexity of one-cycle robotic flow-shops. *Journal of Scheduling*, 6(4), 355-372.
- [15] Mangione, F., Brauner, N., & Penz, B. (2003). Optimal cycles for the robotic balanced no-wait flow shop. In *International Conference of Industrial Engineering and Production Management-IEPM* (p. cdrom).

- [16] Drobouchevitch, I. G., Sethi, S., Sidney, J., & Sriskandarajah, C. (2004). A note on scheduling multiple parts in two-machine dual gripper robotic cell: Heuristic algorithm and performance guarantee. *International Journal of Operations and Quantitative Management*, 10(4), 297-314.
- [17] Sriskandarajah, C., Drobouchevitch, I., Sethi, S. P., & Chandrasekaran, R. (2004). Scheduling multiple parts in a robotic cell served by a dual-gripper robot. *Operations Research*, 52(1), 65-82.
- [18] Geismar, H. N., Dawande, M., & Sriskandarajah, C. (2005). Approximation algorithms for k-unit cyclic solutions in robotic cells. *European Journal of Operational Research*, 162(2), 291-309.
- [19] Akturk, M. S., Gultekin, H., & Karasan, O. E. (2005). Robotic cell scheduling with operational flexibility. *Discrete Applied Mathematics*, 145(3), 334-348.
- [20] Gultekin, H., Akturk, M. S., & Karasan, O. E. (2006). Cyclic scheduling of a 2-machine robotic cell with tooling constraints. *European Journal of Operational Research*, 174(2), 777-796.
- [21] Geismar, H. N., Dawande, M. W., & Sethi, S. P. (2005). Dominance of cyclic solutions and challenges in the scheduling of robotic cells. *SIAM review*, 47(4), 709-721.

- [22] Gultekin, H., Akturk, M. S., & Karasan, O. E. (2007). Scheduling in a three-machine robotic flexible manufacturing cell. *Computers & operations research*, 34(8), 2463-2477.
- [23] Geismar, H. N., Dawande, M., & Sriskandarajah, C. (2007). A (10/7)-approximation algorithm for an optimum cyclic solution in additive travel-time robotic cells. *IIE Transactions*, 39(2), 217-227.
- [24] Yan, P., Chu, C., Che, A., & Yang, N. (2008, December). An algorithm for optimal cyclic scheduling in a robotic cell with flexible processing times. In *Industrial Engineering and Engineering Management, 2008. IEEM 2008. IEEE International Conference on* (pp. 153-157). IEEE.
- [25] Geismar, H. N., Chan, L. M. A., Dawande, M., & Sriskandarajah, C. (2008). Approximations to Optimal k-Unit Cycles for Single-Gripper and Dual-Gripper Robotic Cells. *Production and Operations Management*, 17(5), 551-563.
- [26] Geismar, H. N., Dawande, M., & Sriskandarajah, C. (2006). Throughput Optimization in Constant Travel-Time Dual Gripper Robotic Cells with Parallel Machines. *Production and Operations Management*, 15(2), 311-328.
- [27] Dawande, M., Pinedo, M., & Sriskandarajah, C. (2009). Multiple part-type production in robotic cells: equivalence of two real-world models. *Manufacturing & Service Operations Management*, 11(2), 210-228.

- [28] Dawande, M., Geismar, H. N., Pinedo, M., & Sriskandarajah, C. (2009). Throughput optimization in dual-gripper interval robotic cells. *IEEE Transactions*, 42(1), 1-15.
- [29] Yan, P., Chu, C., Yang, N., & Che, A. (2010). A branch and bound algorithm for optimal cyclic scheduling in a robotic cell with processing time windows. *International Journal of Production Research*, 48(21), 6461-6480.
- [30] Drobouchevitch, I. G., Geismar, H. N., & Sriskandarajah, C. (2010). Throughput optimization in robotic cells with input and output machine buffers: A comparative study of two key models. *European Journal of Operational Research*, 206(3), 623-633.
- [31] Rajapakshe, T., Dawande, M., & Sriskandarajah, C. (2011). Quantifying the impact of layout on productivity: An analysis from robotic-cell manufacturing. *Operations Research*, 59(2), 440-454.
- [32] Geismar, N., Dawande, M., & Sriskandarajah, C. (2011). Productivity improvement from using machine buffers in dual-gripper cluster tools. *IEEE Transactions on Automation Science and Engineering*, 8(1), 29-41.
- [33] Yildiz, S., Karasan, O. E., & Akturk, M. S. (2012). An analysis of cyclic scheduling problems in robot centered cells. *Computers & Operations Research*, 39(6), 1290-1299.

- [34] Foumani, M., & Jenab, K. (2013). Analysis of flexible robotic cells with improved pure cycle. *International Journal of Computer Integrated Manufacturing*, 26(3), 201-215.
- [35] Foumani, M., & Jenab, K. (2012). Cycle time analysis in reentrant robotic cells with swap ability. *International Journal of Production Research*, 50(22), 6372-6387.
- [36] Zhou, Z., Che, A., & Yan, P. (2012). A mixed integer programming approach for multi-cyclic robotic flowshop scheduling with time window constraints. *Applied Mathematical Modelling*, 36(8), 3621-3629.
- [37] Geismar, N., Manoj, U. V., Sethi, A., & Sriskandarajah, C. (2012). Scheduling robotic cells served by a dual-arm robot. *IIE Transactions*, 44(3), 230-248.
- [38] Zarandi, M. F., Mosadegh, H., & Fattahi, M. (2013). Two-machine robotic cell scheduling problem with sequence-dependent setup times. *Computers & Operations Research*, 40(5), 1420-1434.
- [39] Foumani, M., Ibrahim, M. Y., & Gunawan, I. (2013, May). Scheduling dual gripper robotic cells with a hub machine. In *Industrial Electronics (ISIE), 2013 IEEE International Symposium on* (pp. 1-6). IEEE.
- [40] Lei, W., Che, A., & Chu, C. (2014). Optimal cyclic scheduling of a robotic flowshop with multiple part types and flexible processing times. *European Journal of Industrial Engineering*, 8(2), 143-167.

- [41] Elmi, A., & Topaloglu, S. (2014). Scheduling multiple parts in hybrid flow shop robotic cells served by a single robot. *International Journal of Computer Integrated Manufacturing*, 27(12), 1144-1159.
- [42] Jung, K. S., Geismar, H. N., Pinedo, M., & Sriskandarajah, C. (2015). Approximations to optimal sequences in single-gripper and dual-gripper robotic cells with circular layouts. *IIE Transactions*, 47(6), 634-652.
- [43] Gundogdu, E., & Gultekin, H. (2016). Scheduling in two-machine robotic cells with a self-buffered robot. *IIE Transactions*, 48(2), 170-191.
- [44] Güden, H., & Meral, S. (2016). An adaptive simulated annealing algorithm-based approach for assembly line balancing and a real-life case study. *The International Journal of Advanced Manufacturing Technology*, 84(5-8), 1539-1559.

## **APPENDICES**



## Appendix A: Simulated Annealing Solution Time Convergence

Graphs for Case when  $\delta = 2$ ,  $\varepsilon = 1$ ,  $P = 0$ ,  $P = 22$ ,  $P = 40$ ,  $P = 50$  and

$P = 100$

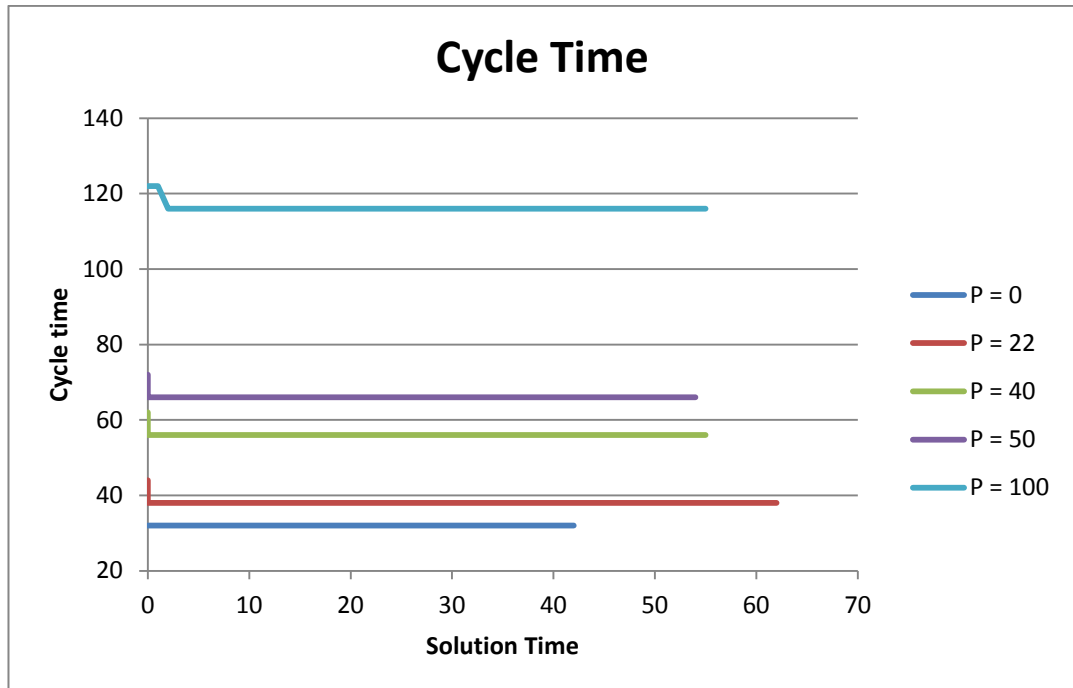


Figure A.1: Cycle Time vs Solution Time Convergence Graph for  $m = 2$ ,  $K = 1$

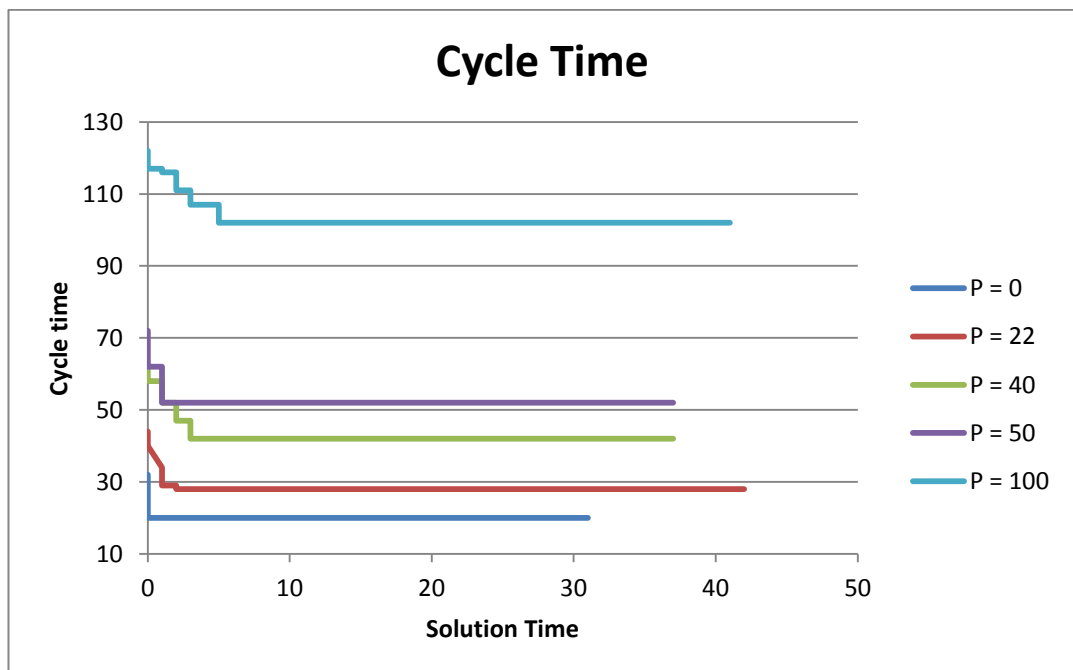


Figure A.2: Cycle Time vs Solution Time Convergence Graph for  $m = 2$ ,  $K = 2$

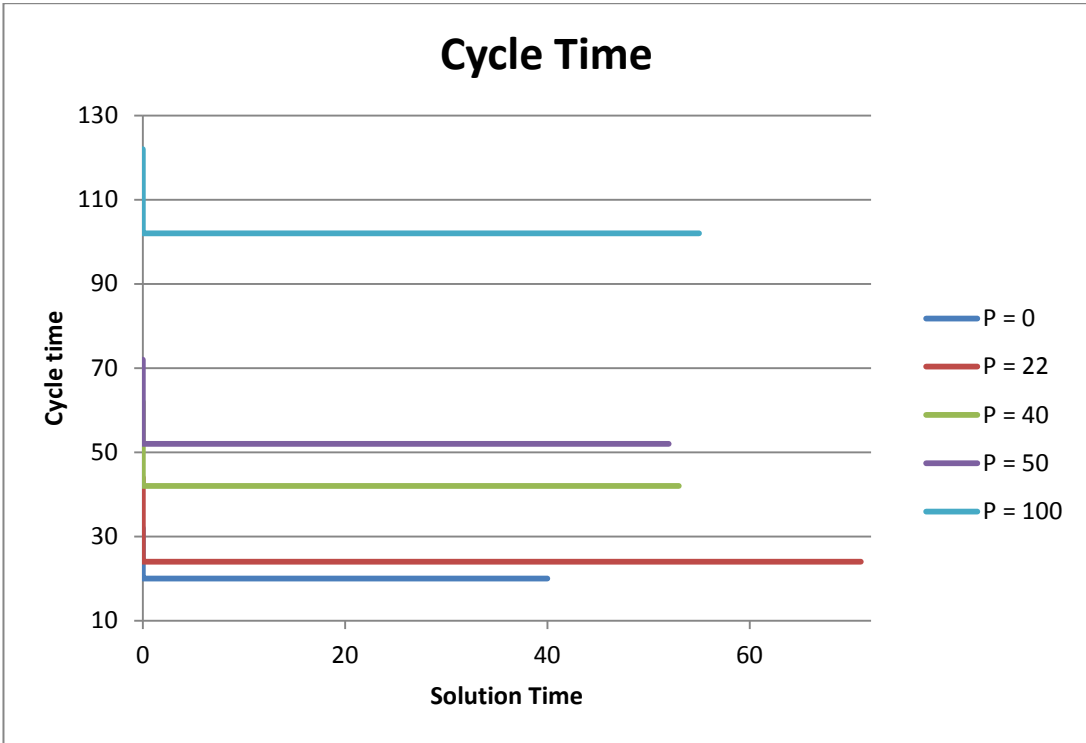


Figure A.3: Cycle Time vs Solution Time Convergence Graph for  $m = 2$ ,  $K = 3$

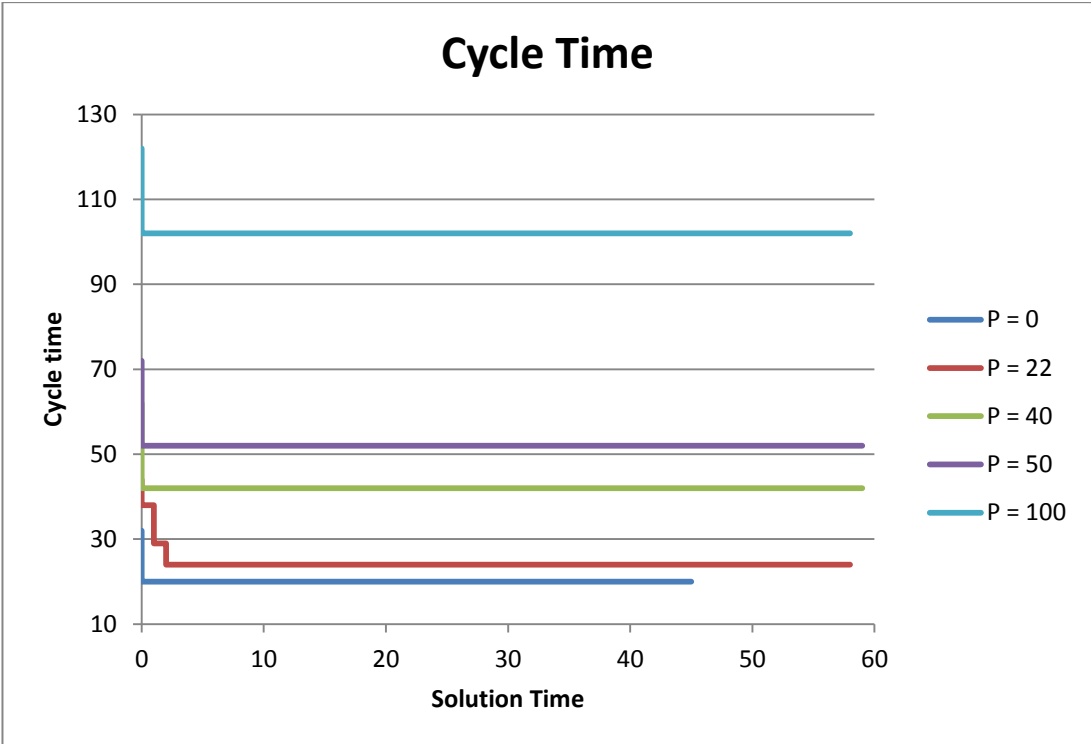


Figure A.4: Cycle Time vs Solution Time Convergence Graph for  $m = 2$ ,  $K = 4$

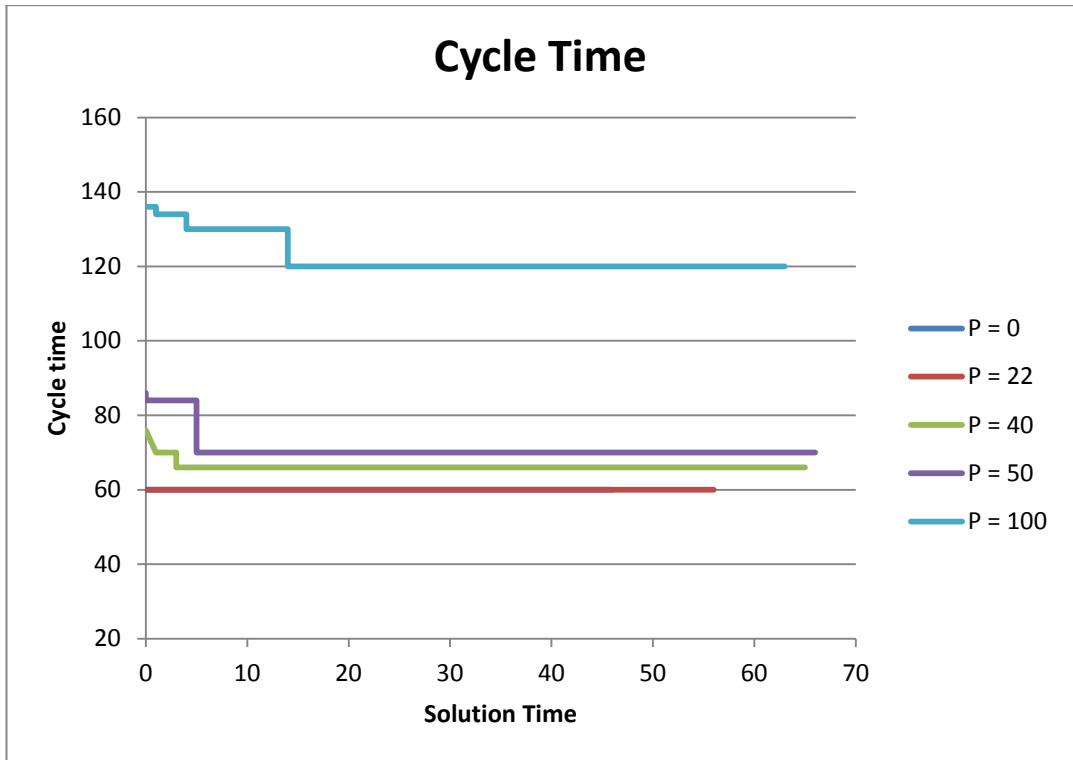


Figure A.5: Cycle Time vs Solution Time Convergence Graph for  $m = 3$ ,  $K = 1$

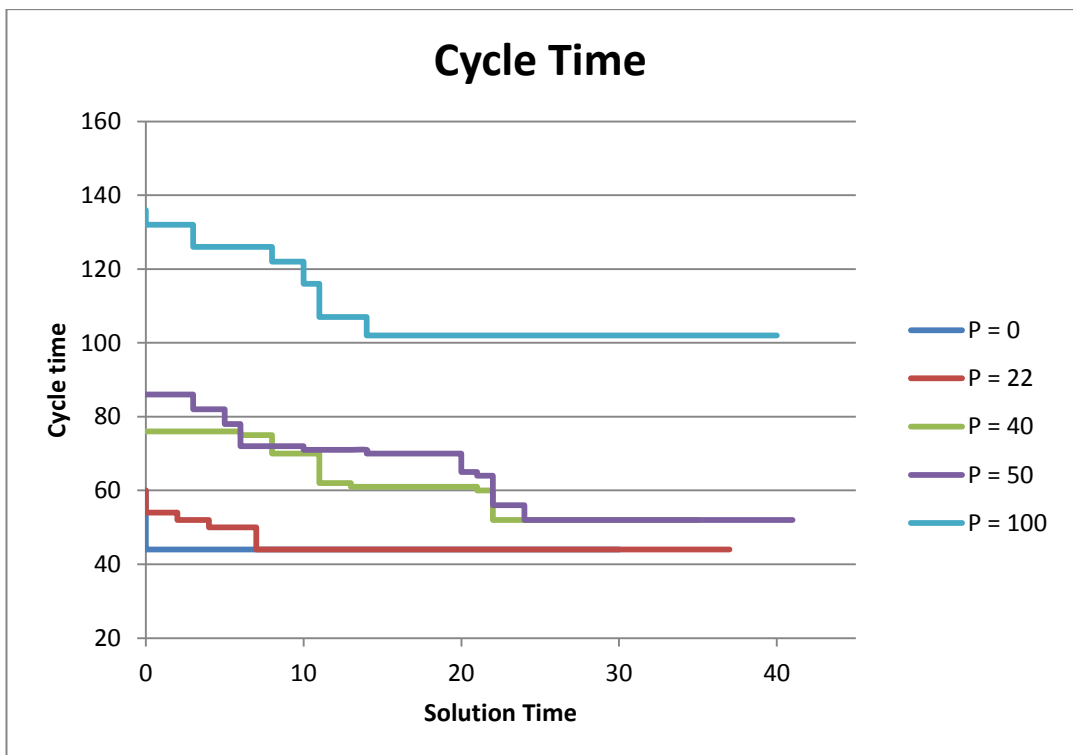


Figure A.6: Cycle Time vs Solution Time Convergence Graph for  $m = 3$ ,  $K = 2$

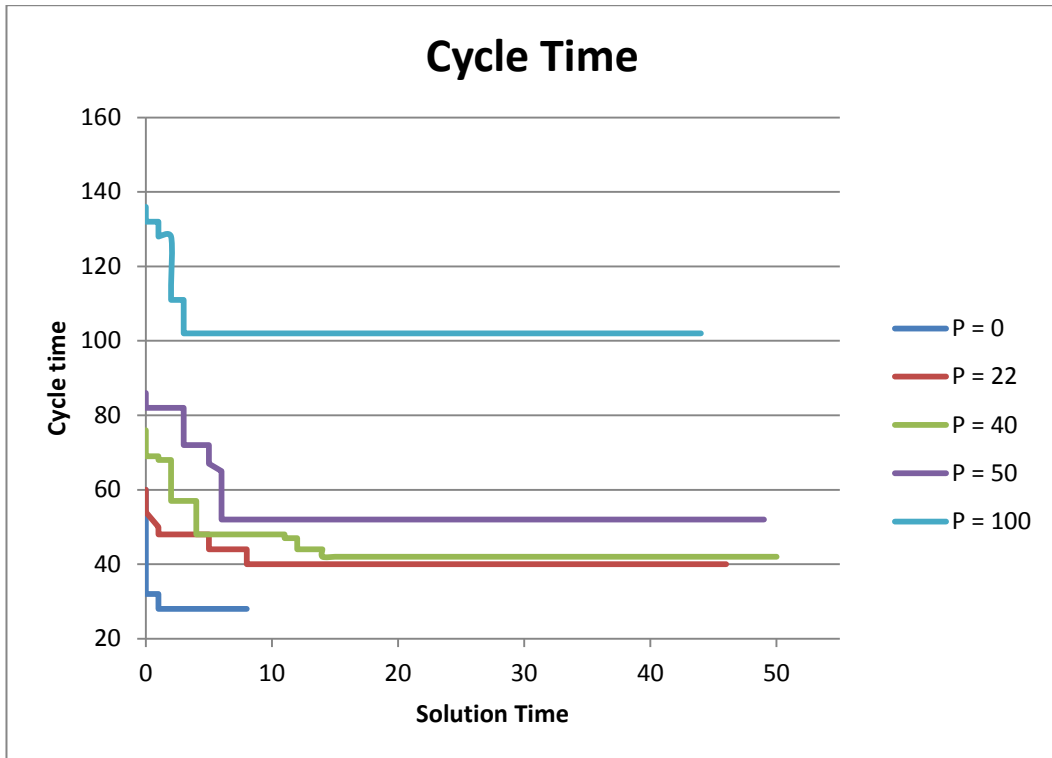


Figure A.7: Cycle Time vs Solution Time Convergence Graph for  $m = 3$ ,  $K = 3$

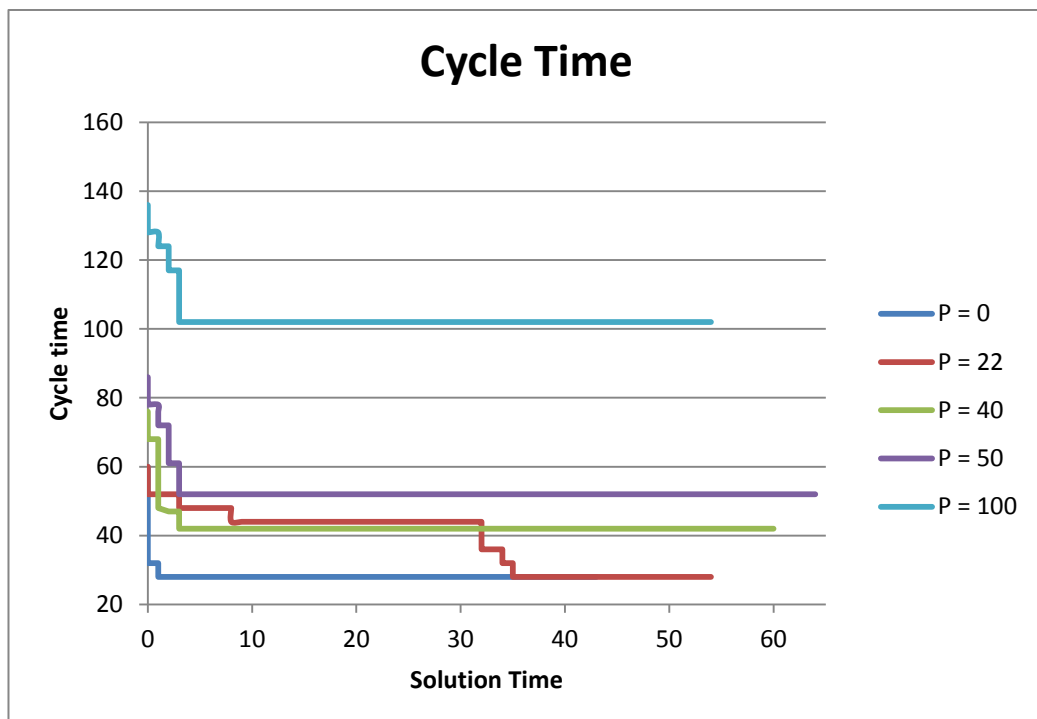


Figure A.8: Cycle Time vs Solution Time Convergence Graph for  $m = 3$ ,  $K = 4$

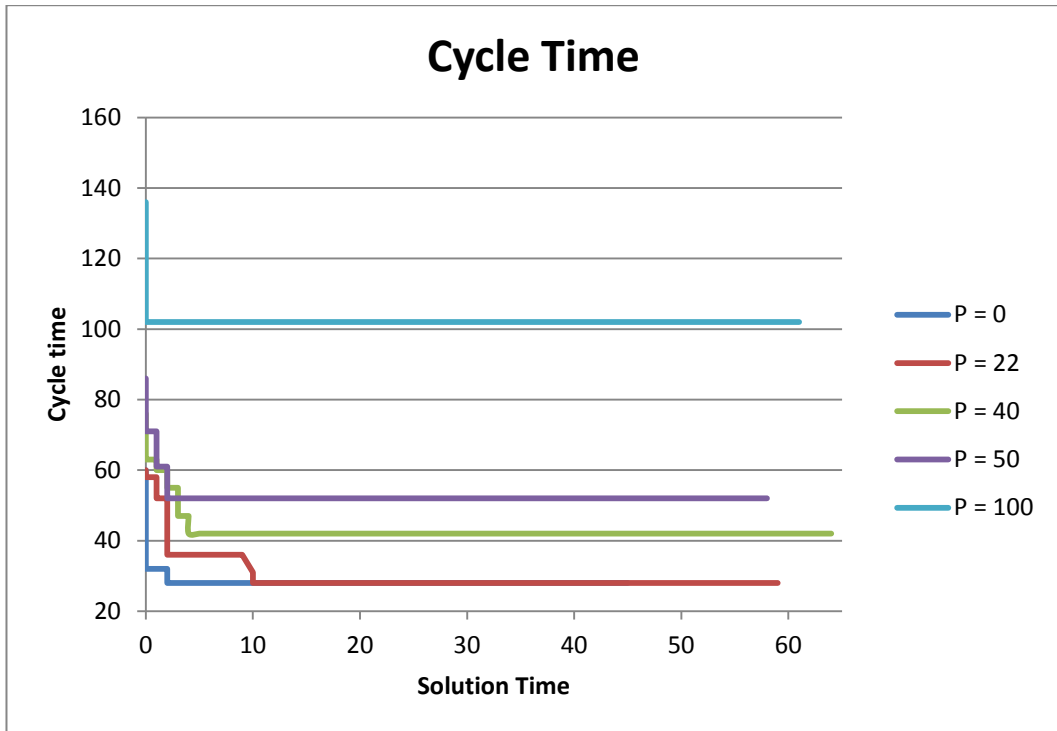


Figure A.9: Cycle Time vs Solution Time Convergence Graph for  $m = 3$ ,  $K = 5$

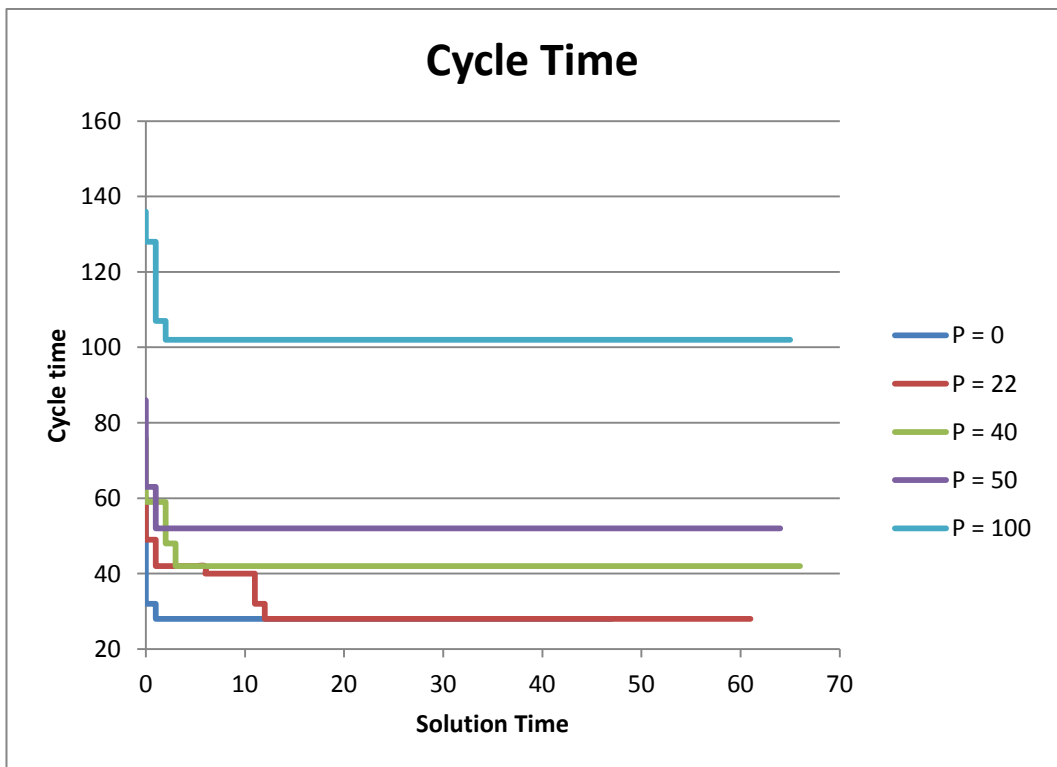


Figure A.10: Cycle Time vs Solution Time Convergence Graph for  $m = 3$ ,  $K = 6$

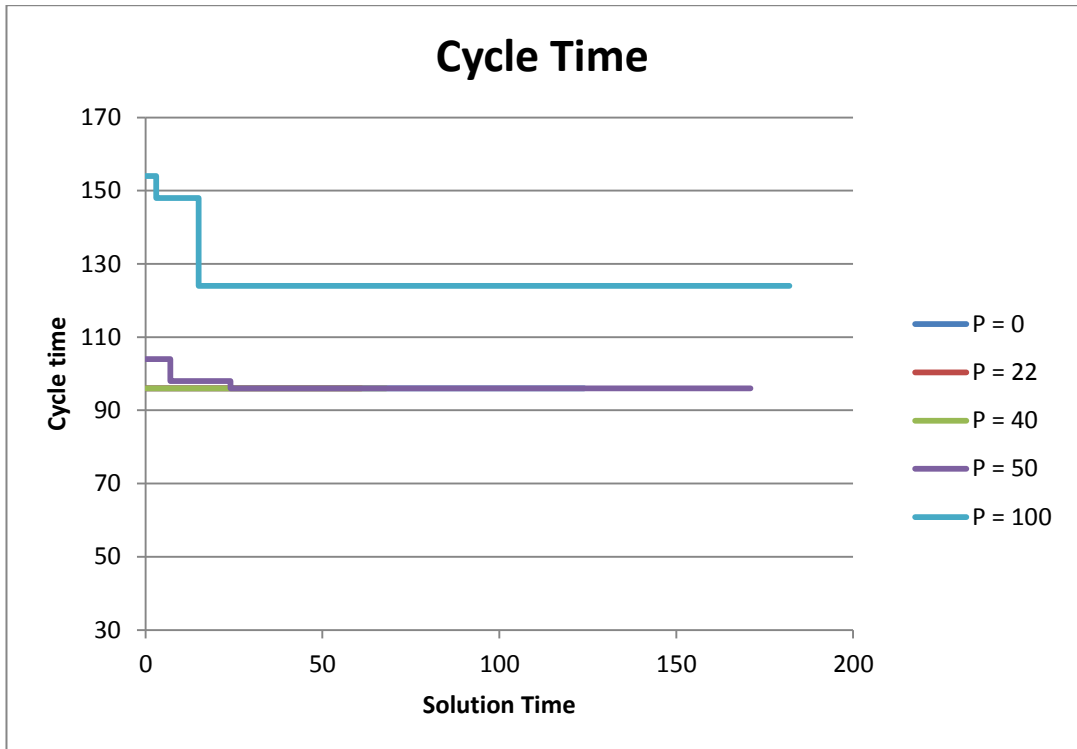


Figure A.11: Cycle Time vs Solution Time Convergence Graph for  $m = 4, K = 1$

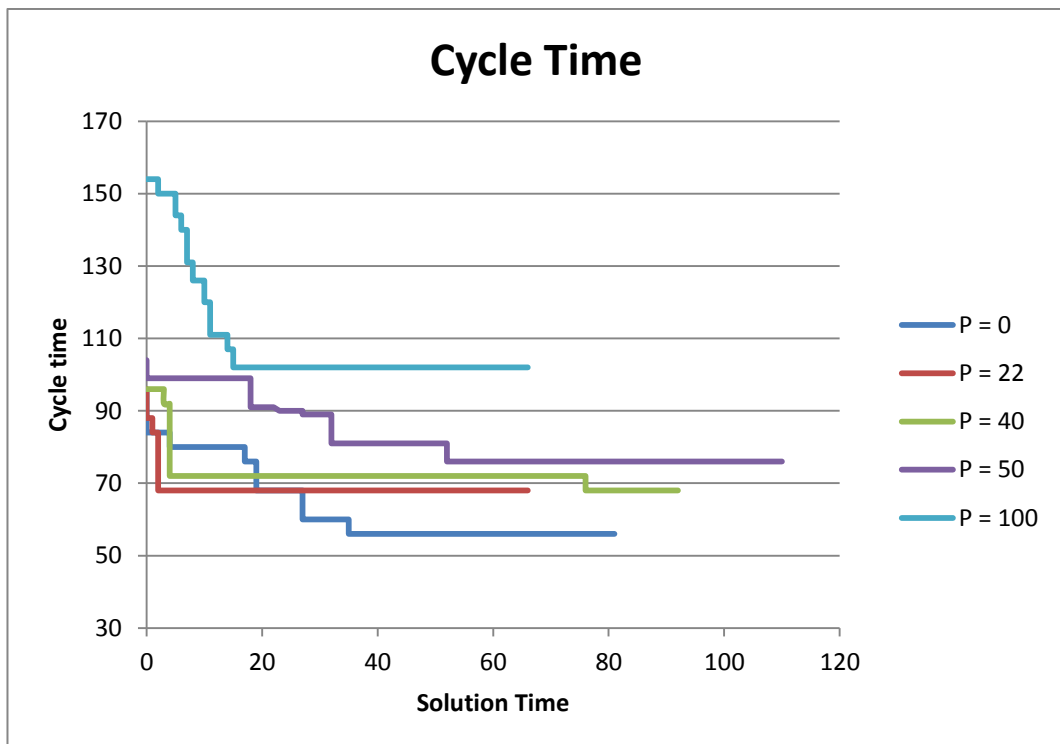


Figure A.12: Cycle Time vs Solution Time Convergence Graph for  $m = 4, K = 2$

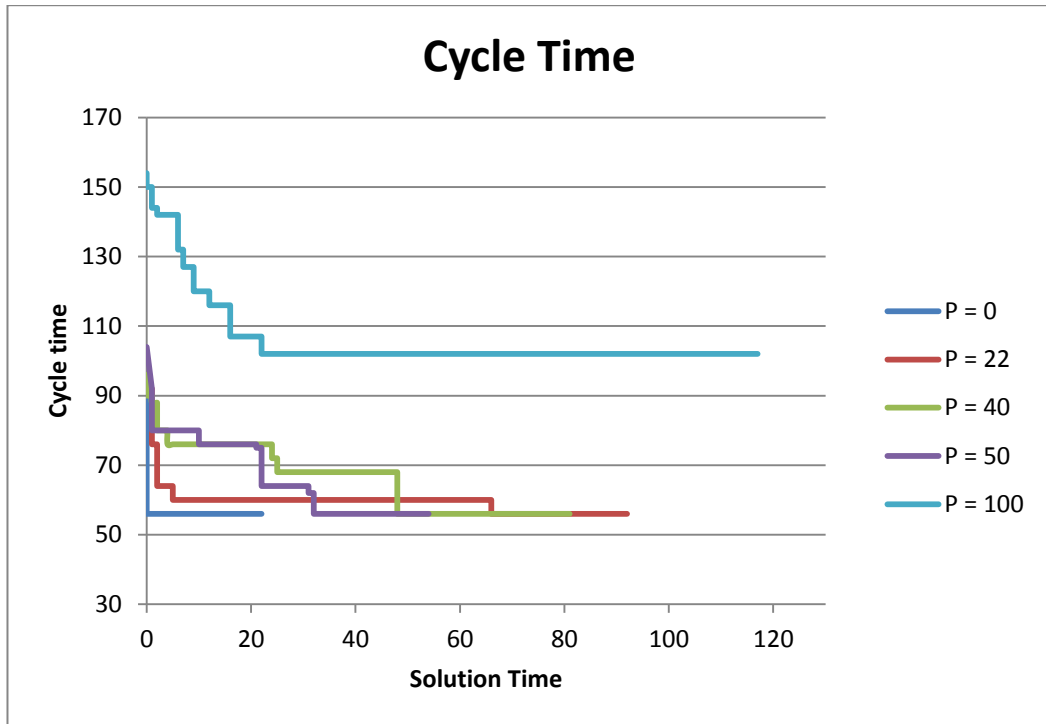


Figure A.13: Cycle Time vs Solution Time Convergence Graph for  $m = 4$ ,  $K = 3$

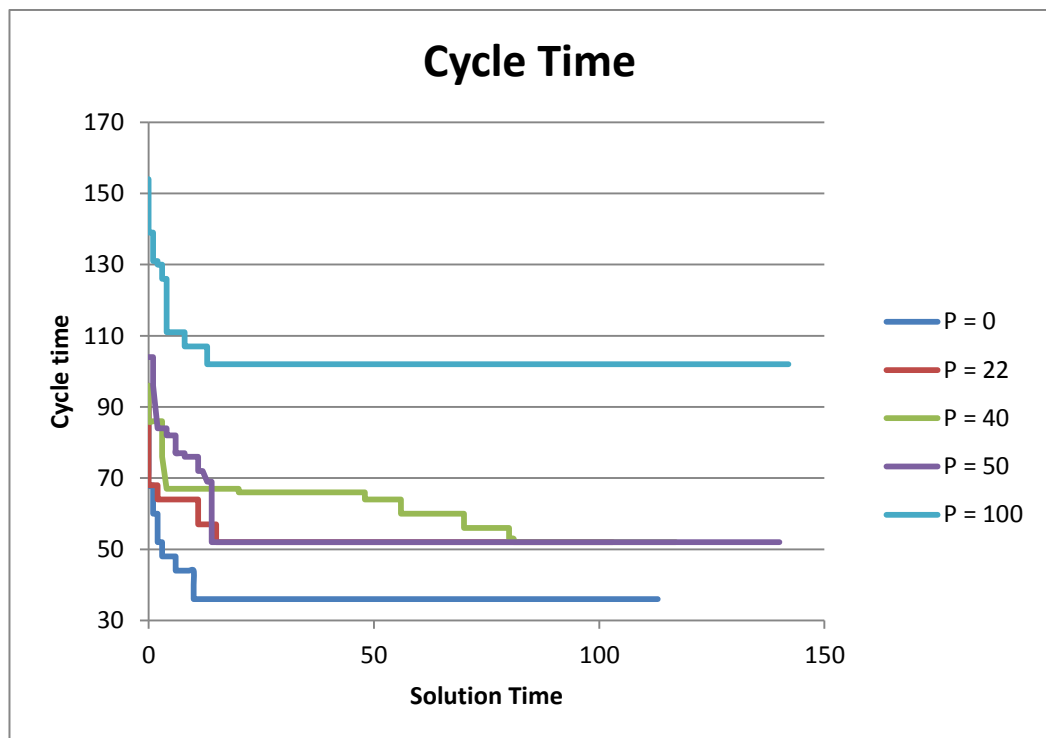


Figure A.14: Cycle Time vs Solution Time Convergence Graph for  $m = 4$ ,  $K = 4$

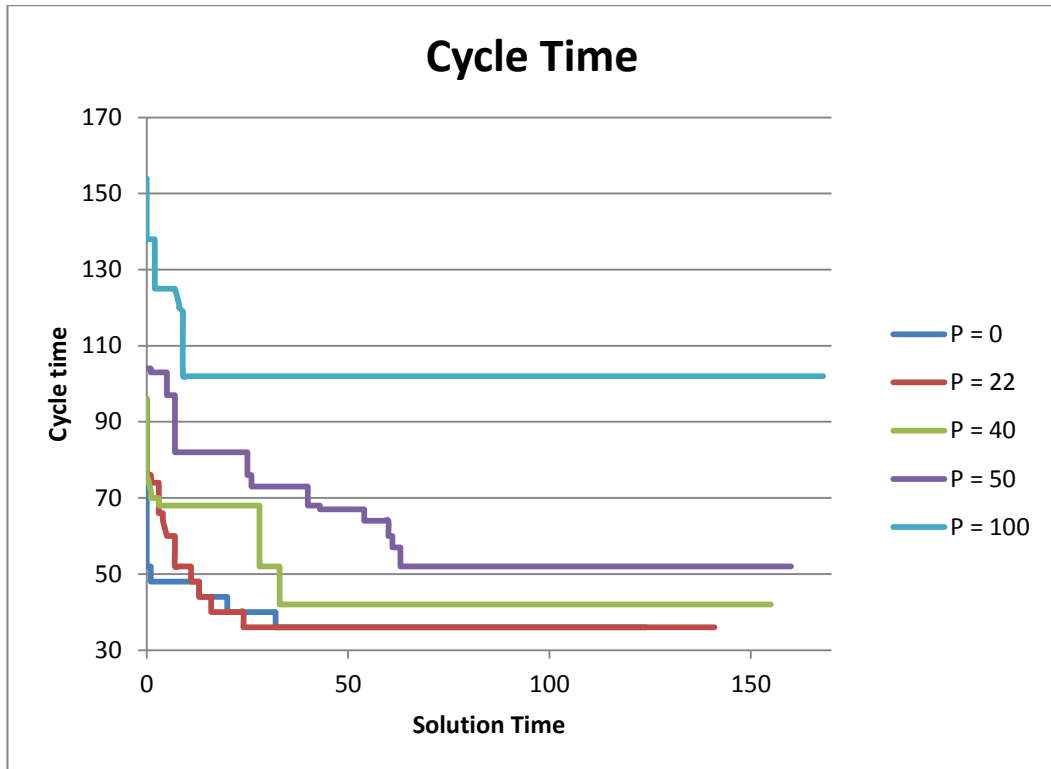


Figure A.15: Cycle Time vs Solution Time Convergence Graph for  $m = 4$ ,  $K = 5$

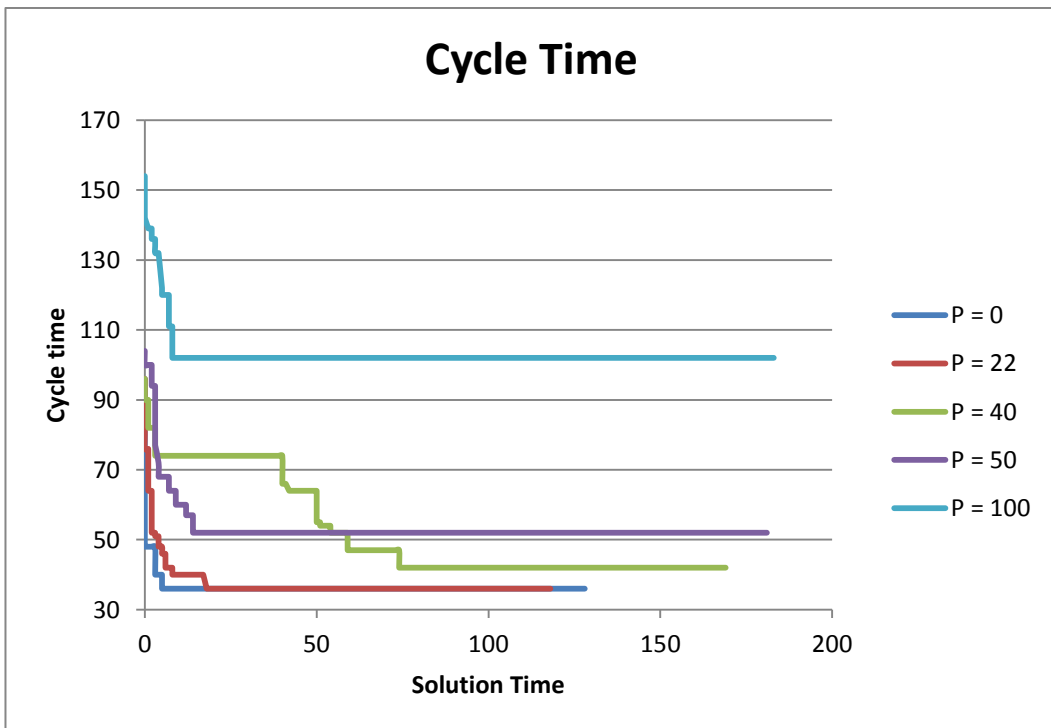


Figure A.16: Cycle Time vs Solution Time Convergence Graph for  $m = 4$ ,  $K = 6$



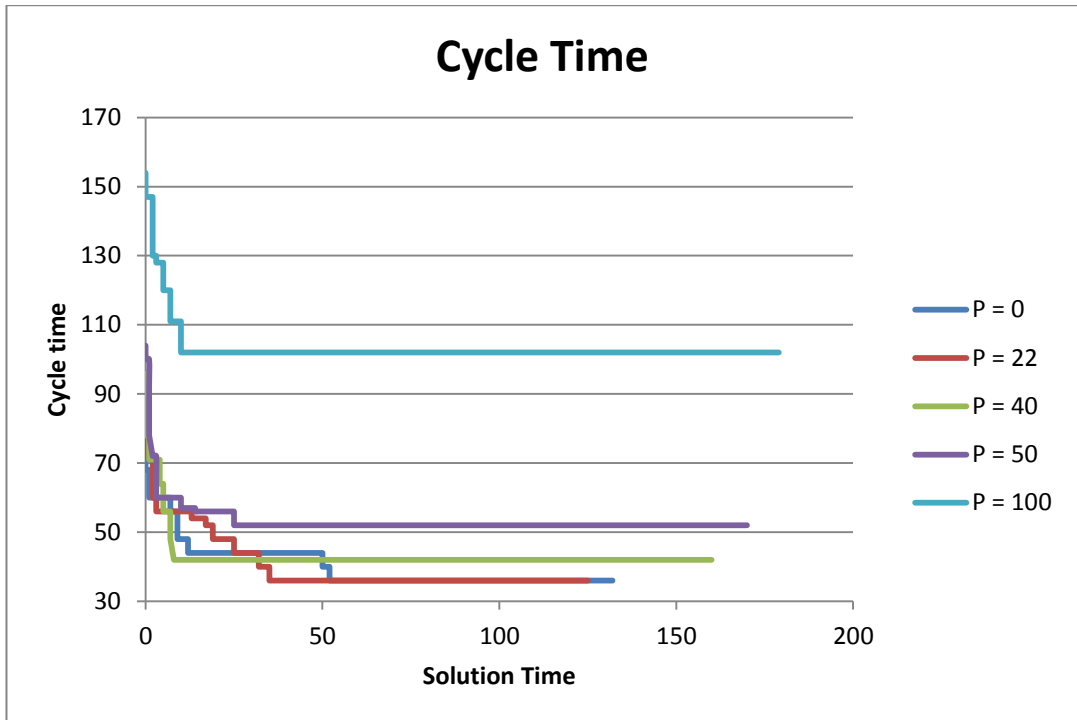


Figure A.17: Cycle Time vs Solution Time Convergence Graph for  $m = 4$ ,  $K = 7$

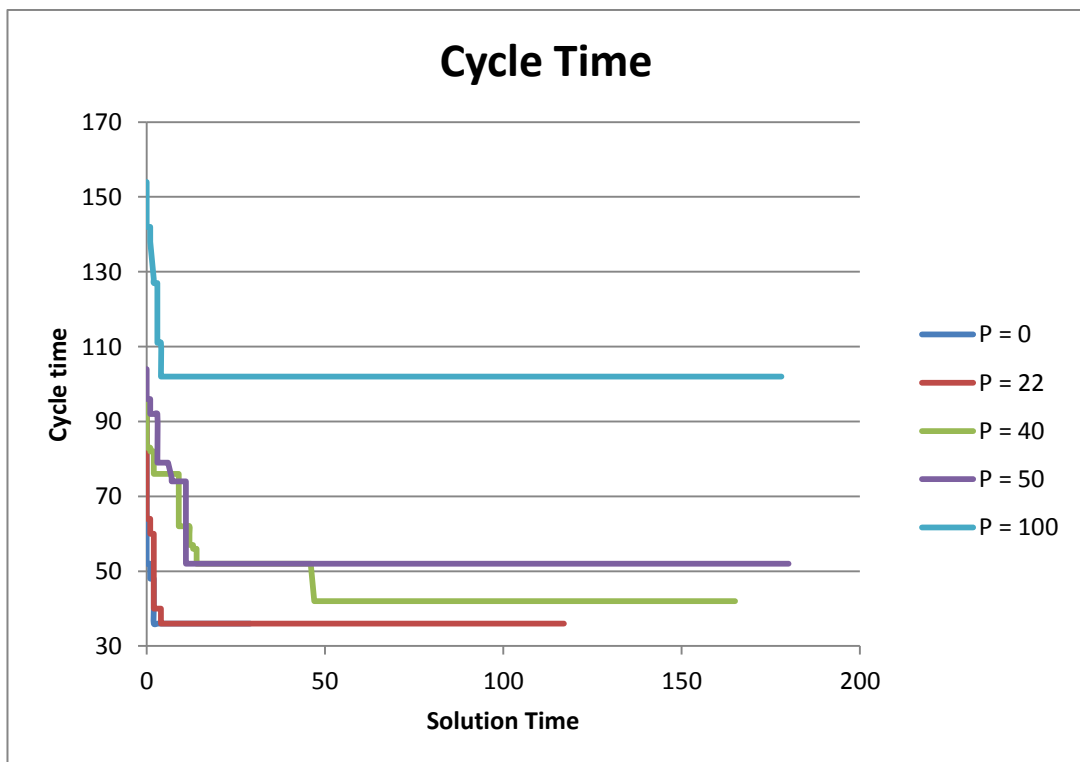


Figure A.18: Cycle Time vs Solution Time Convergence Graph for  $m = 4$ ,  $K = 8$

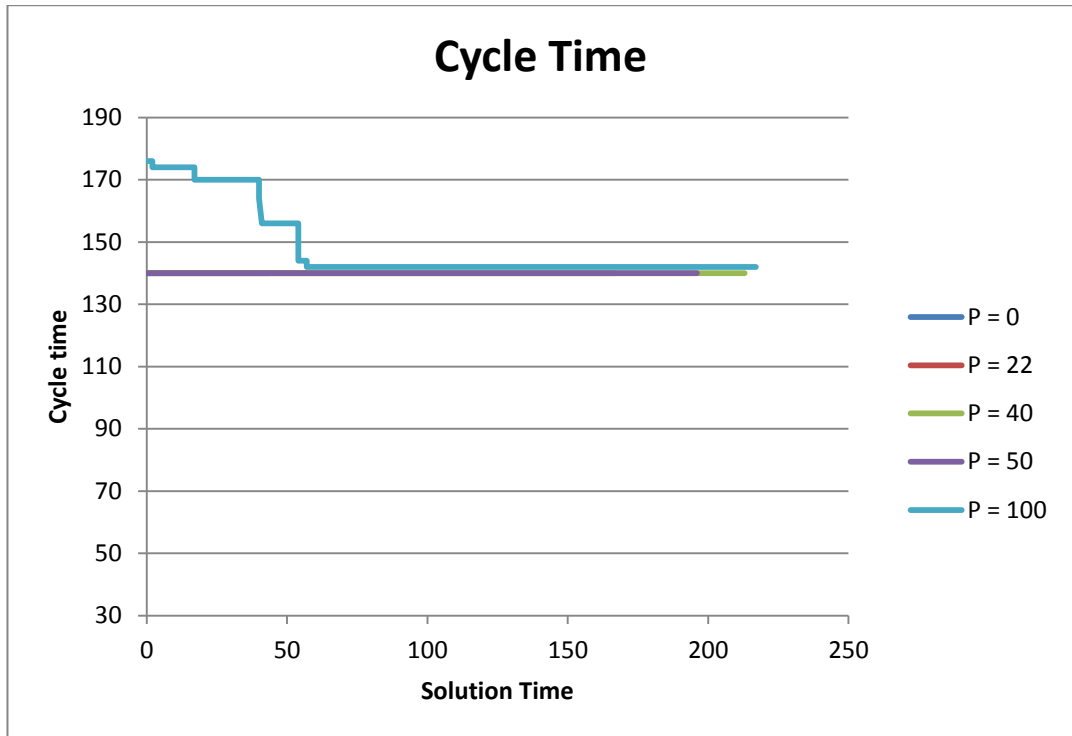


Figure A.19: Cycle Time vs Solution Time Convergence Graph for  $m = 5$ ,  $K = 1$

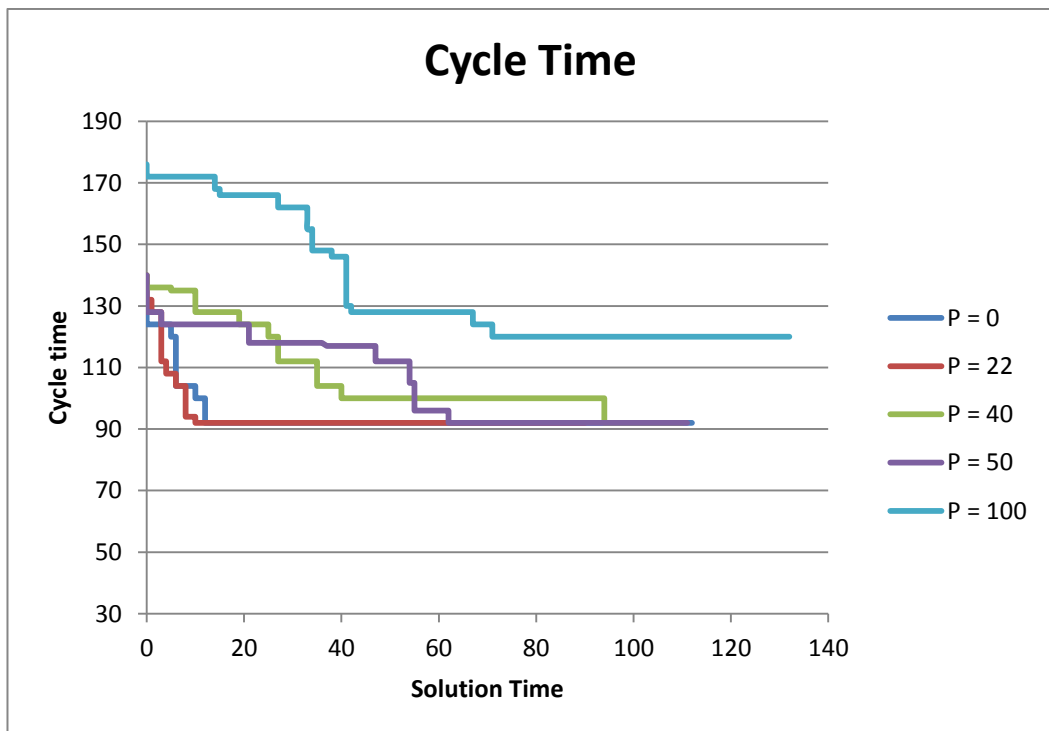


Figure A.20: Cycle Time vs Solution Time Convergence Graph for  $m = 5$ ,  $K = 2$

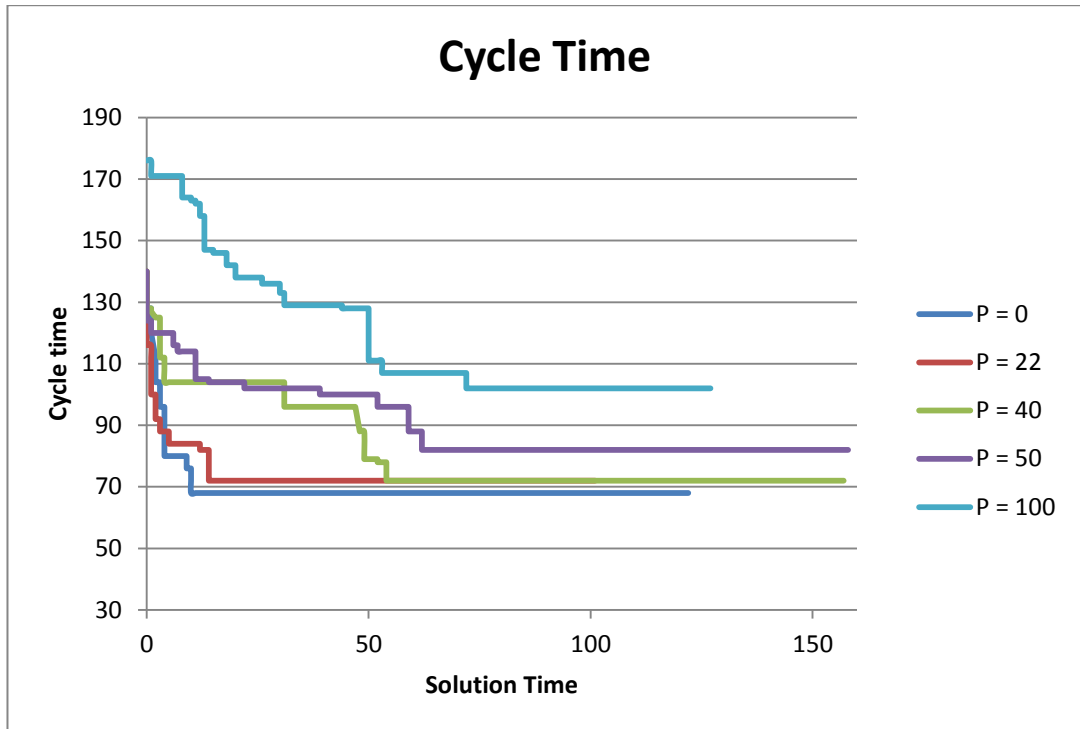


Figure A.21: Cycle Time vs Solution Time Convergence Graph for  $m = 5$ ,  $K = 3$

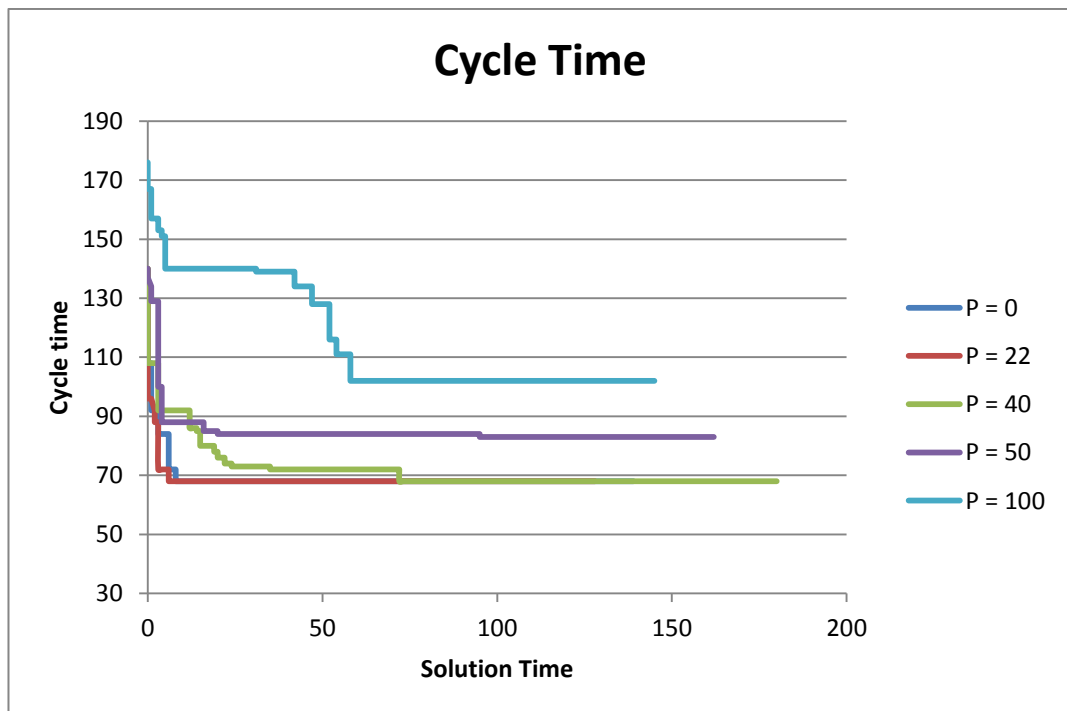


Figure A.22: Cycle Time vs Solution Time Convergence Graph for  $m = 5$ ,  $K = 4$

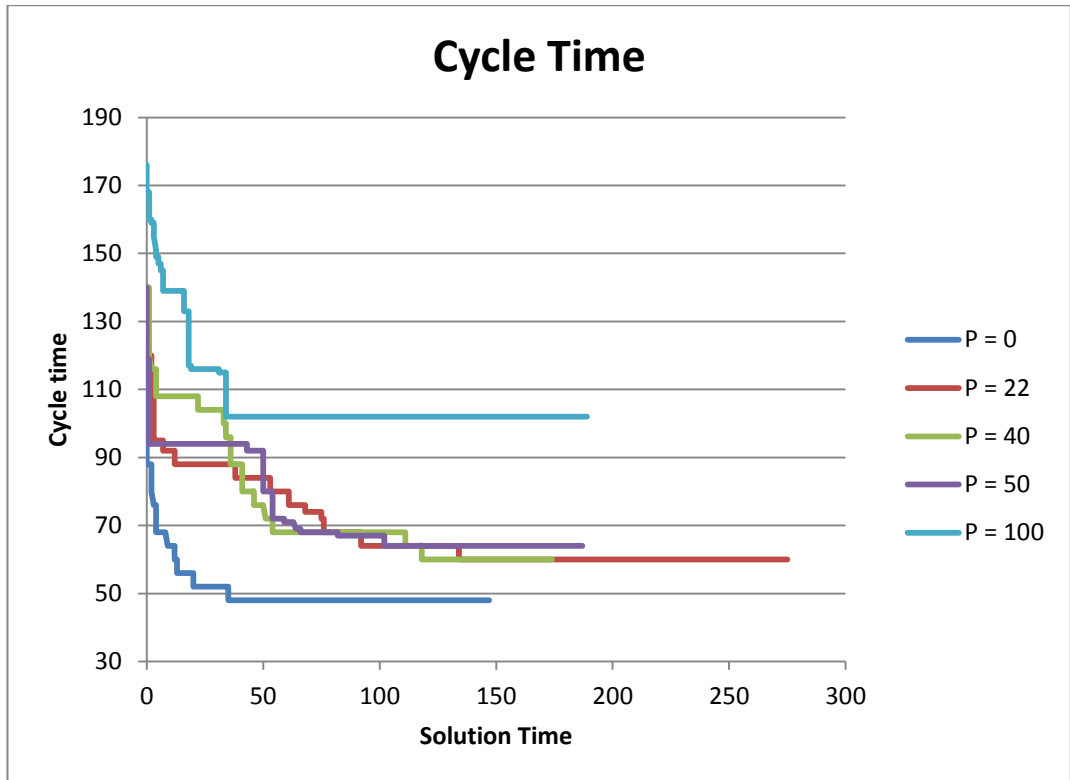


Figure A.23: Cycle Time vs Solution Time Convergence Graph for  $m = 5$ ,  $K = 5$

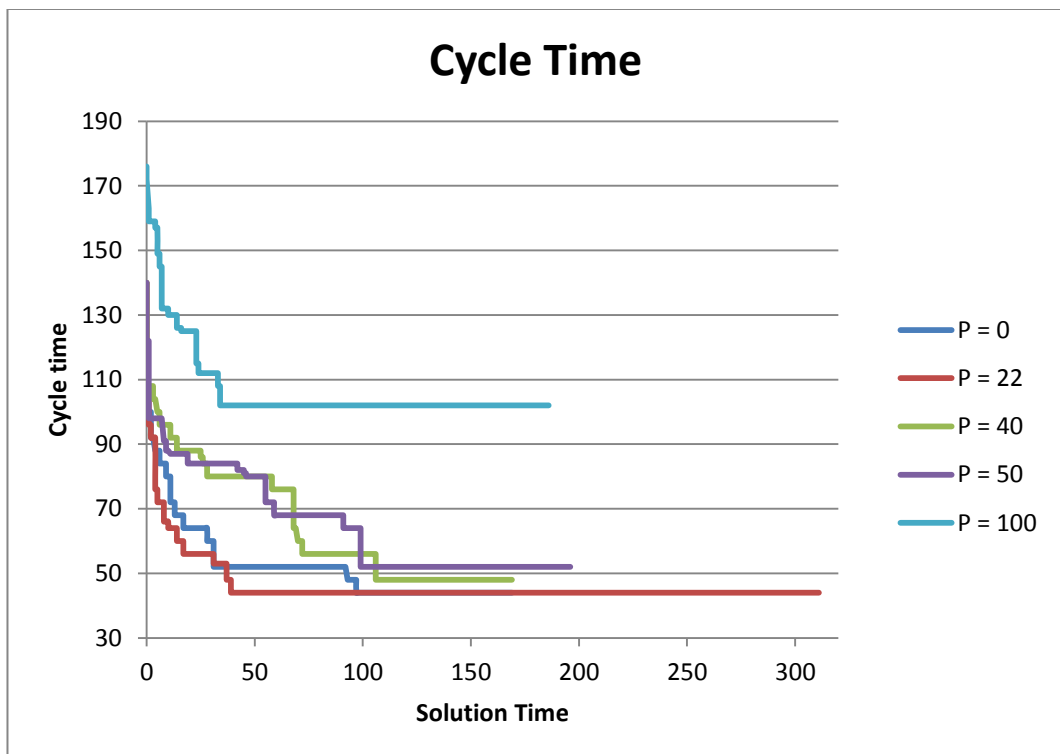


Figure A.24: Cycle Time vs Solution Time Convergence Graph for  $m = 5$ ,  $K = 6$

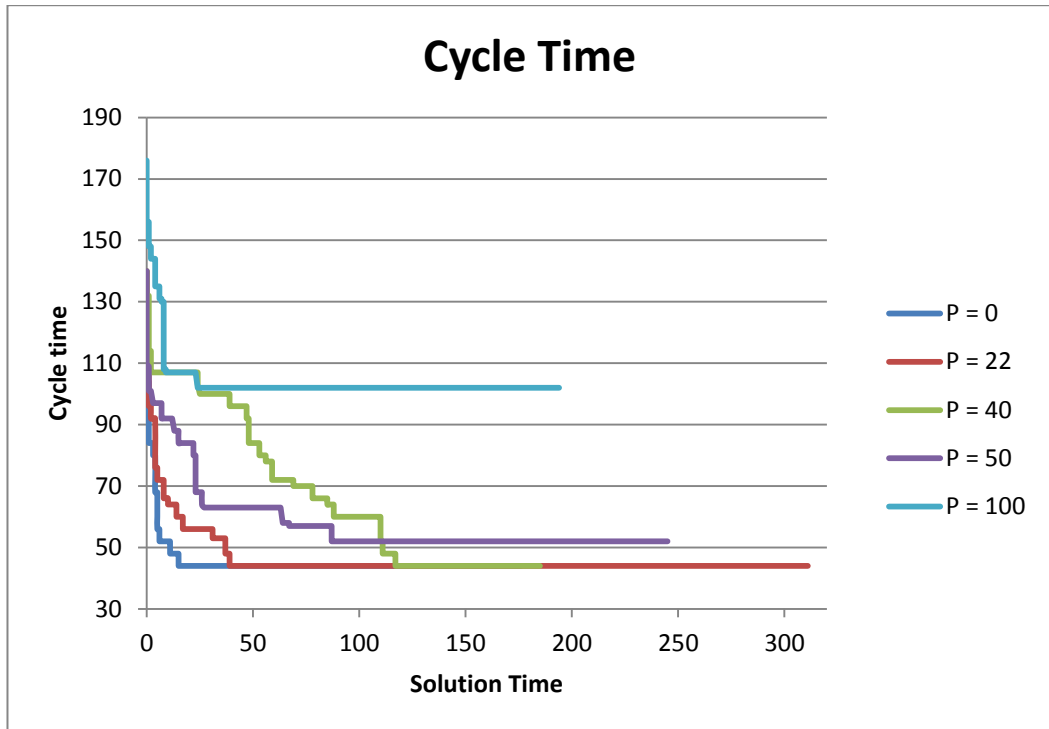


Figure A.25: Cycle Time vs Solution Time Convergence Graph for  $m = 5$ ,  $K = 7$

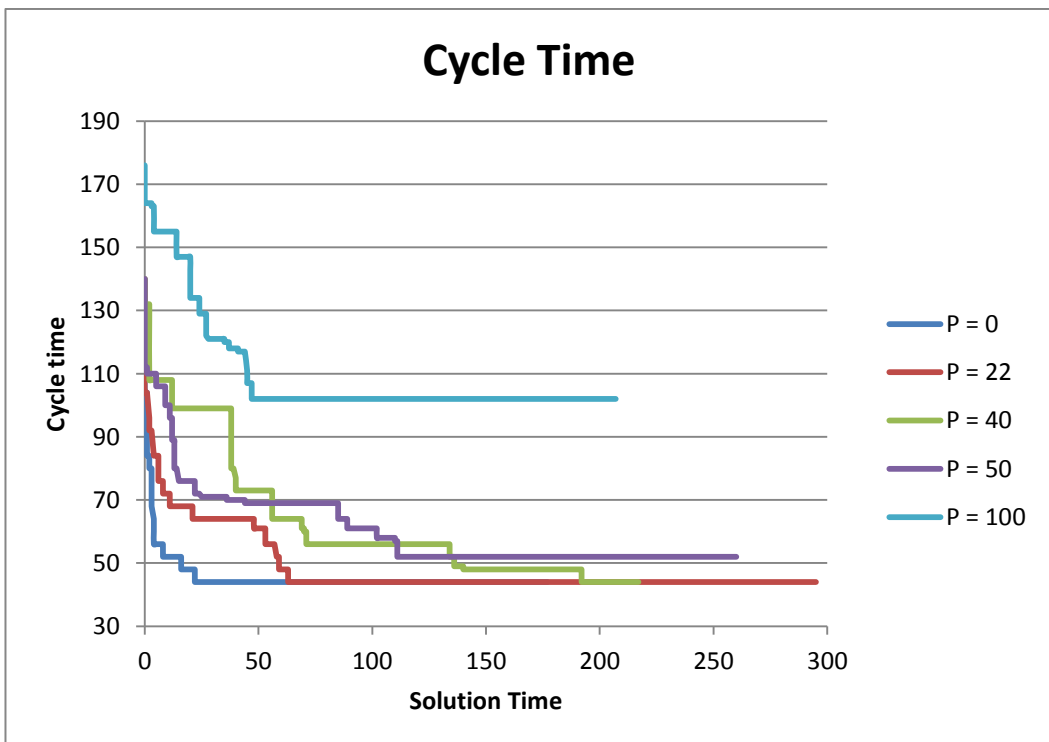


Figure A.26: Cycle Time vs Solution Time Convergence Graph for  $m = 5$ ,  $K = 8$

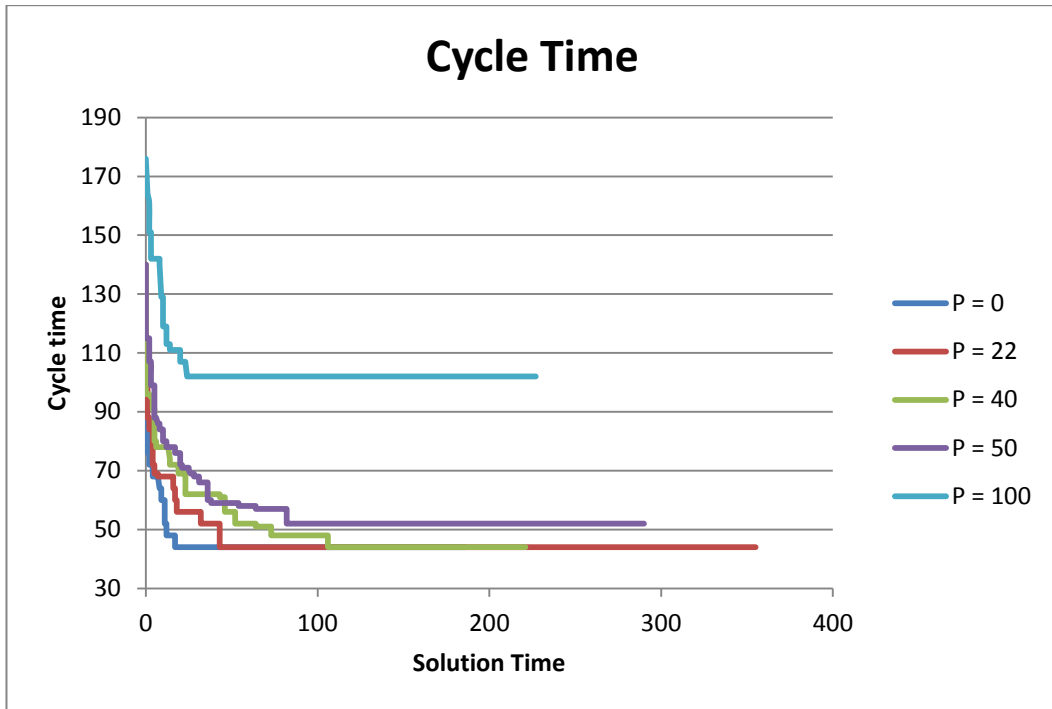


Figure A.27: Cycle Time vs Solution Time Convergence Graph for  $m = 5$ ,  $K = 9$

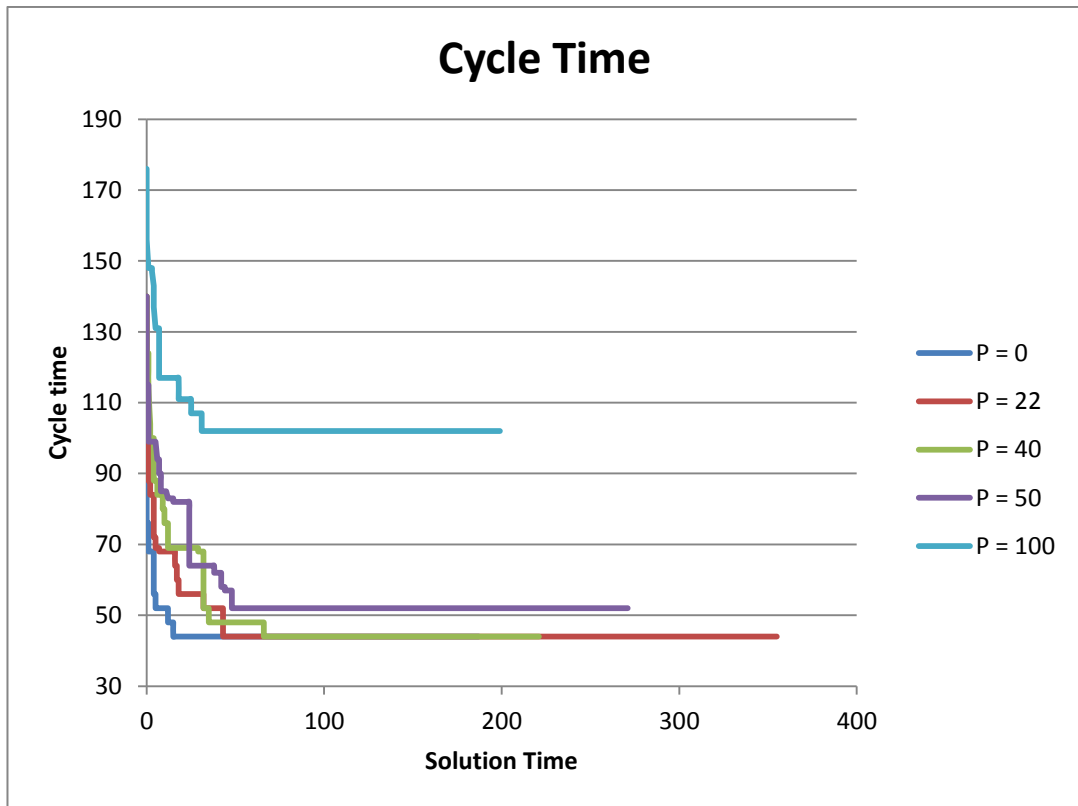


Figure A.28: Cycle Time vs Solution Time Convergence Graph for  $m = 5$ ,  $K = 10$

## Appendix B: Simulated Annealing Solution Time Convergence

Graphs for Case when  $\delta = 2$ ,  $\varepsilon = 1$ ,  $P = 5000$

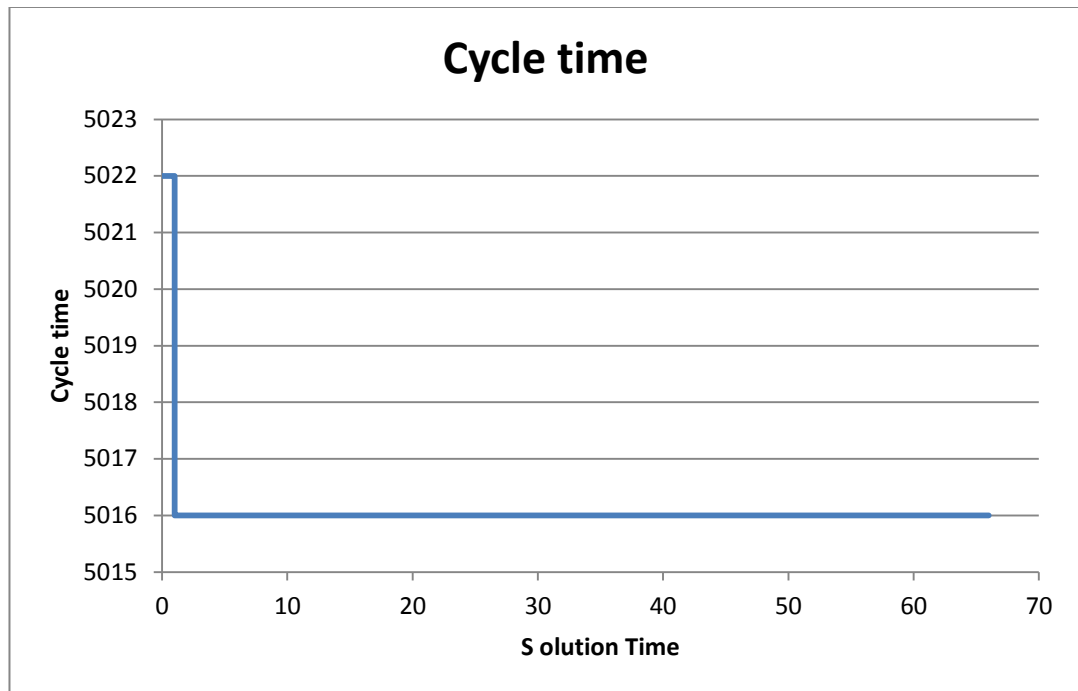


Figure B.1: Cycle Time vs Solution Time Convergence Graph for  $m = 2$ ,  $K = 1$

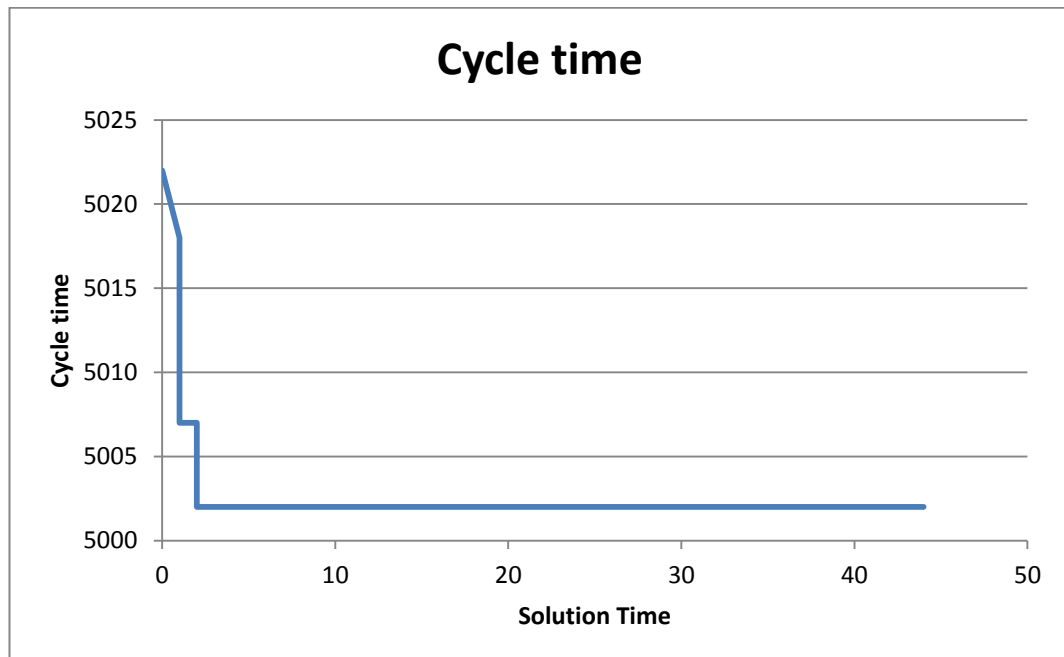


Figure B.2: Cycle Time vs Solution Time Convergence Graph for  $m = 2$ ,  $K = 2$

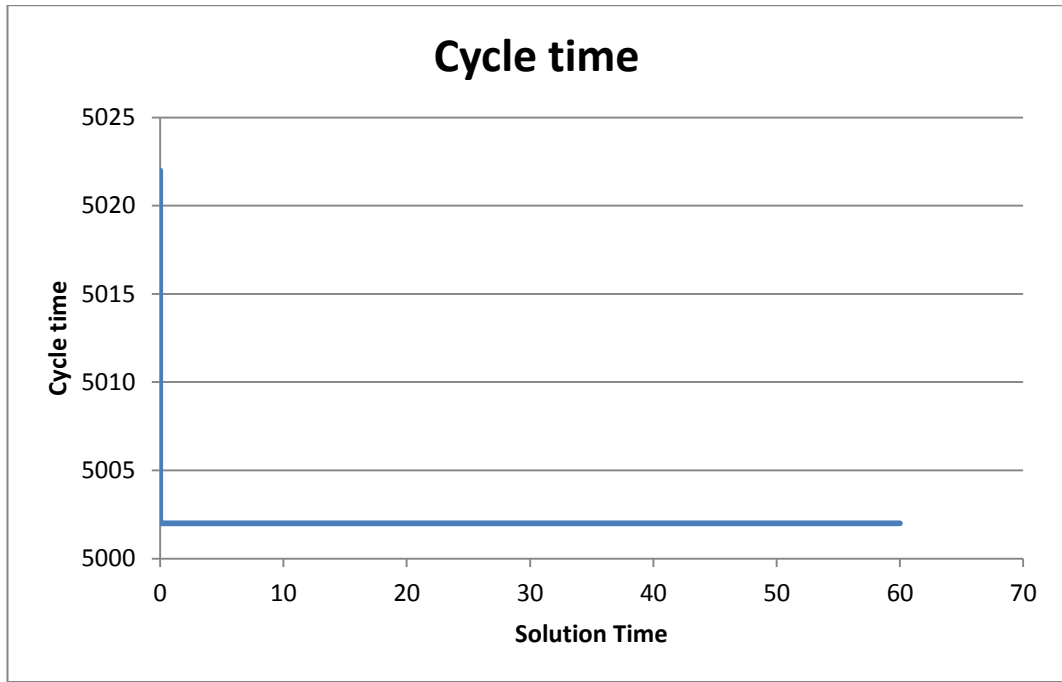


Figure B.3: Cycle Time vs Solution Time Convergence Graph for  $m = 2$ ,  $K = 3$

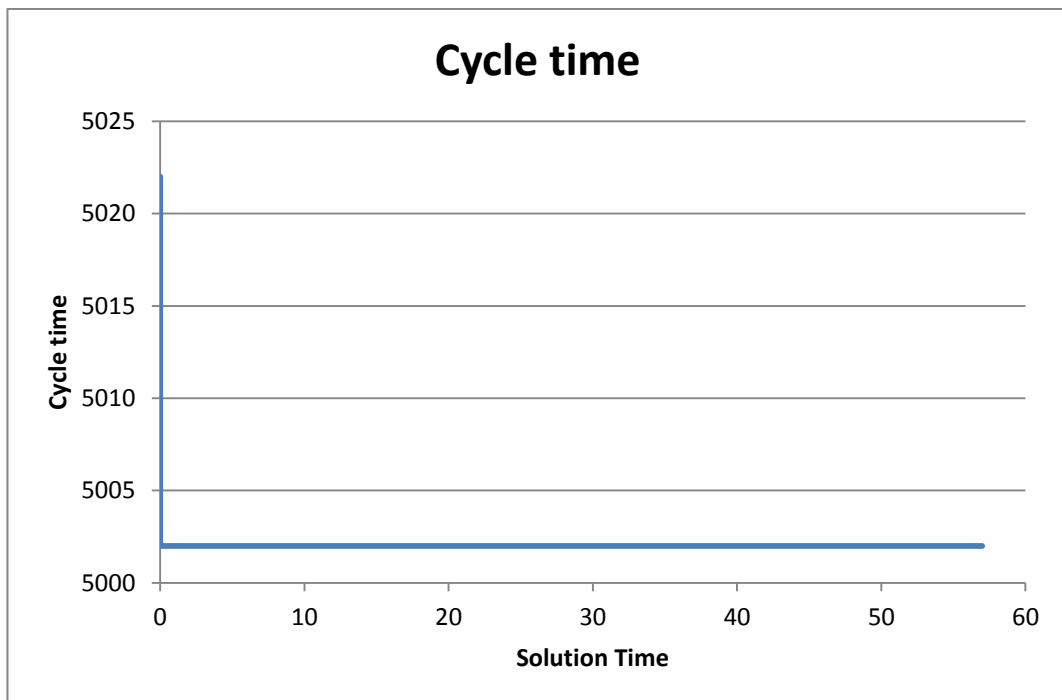


Figure B.4: Cycle Time vs Solution Time Convergence Graph for  $m = 2$ ,  $K = 4$



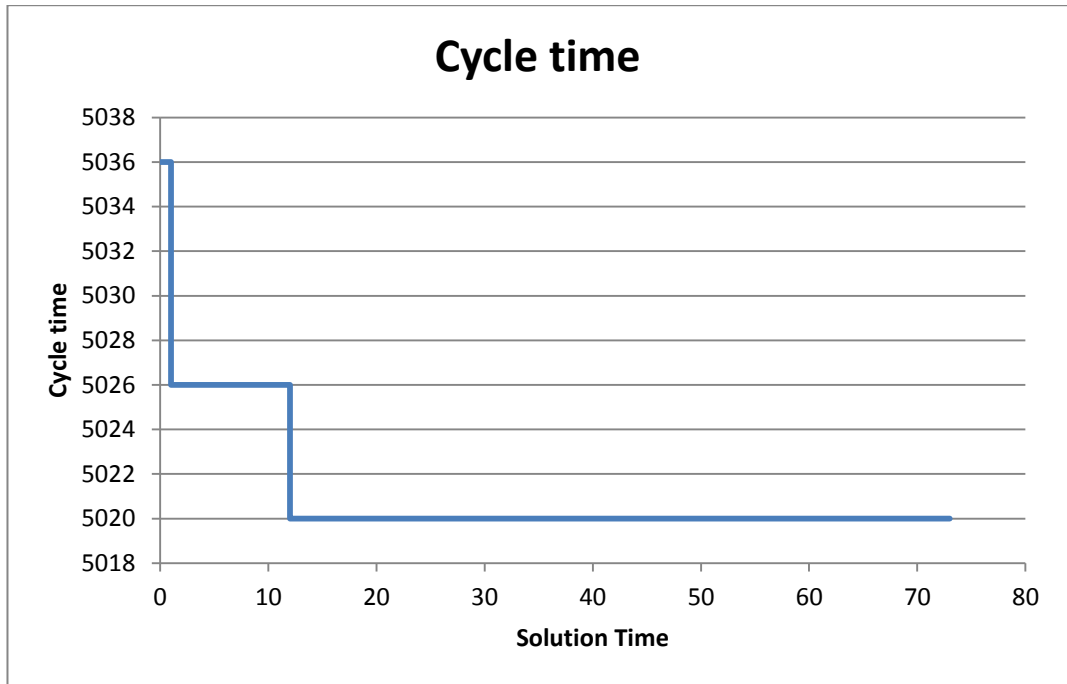


Figure B.5: Cycle Time vs Solution Time Convergence Graph for  $m = 3$ ,  $K = 1$

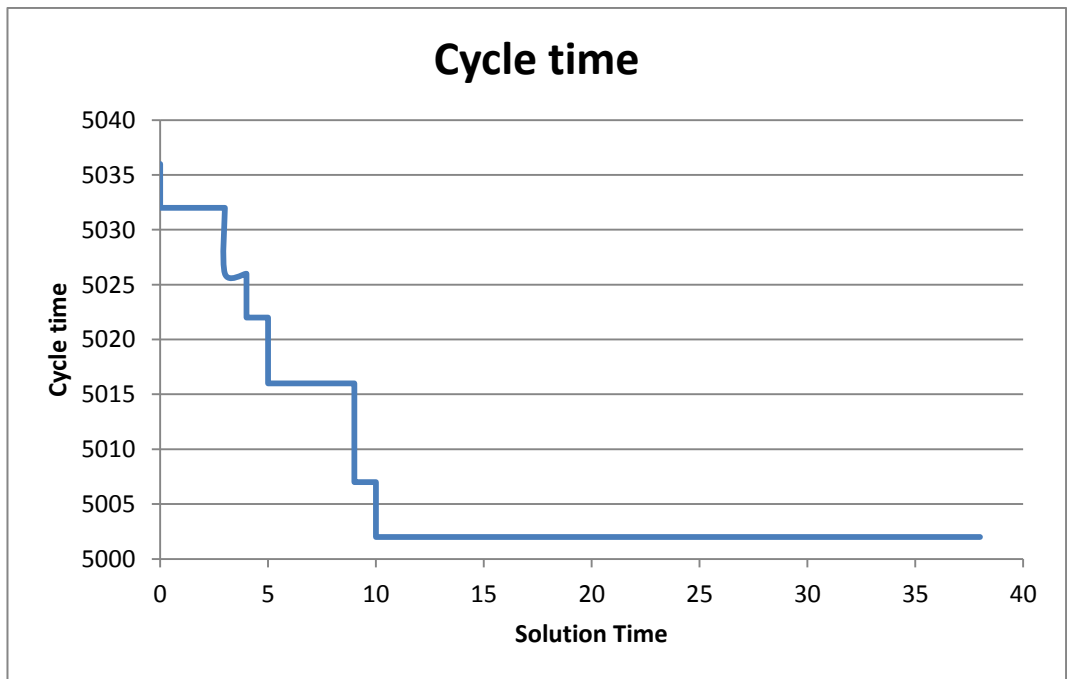


Figure B.6: Cycle Time vs Solution Time Convergence Graph for  $m = 3$ ,  $K = 2$

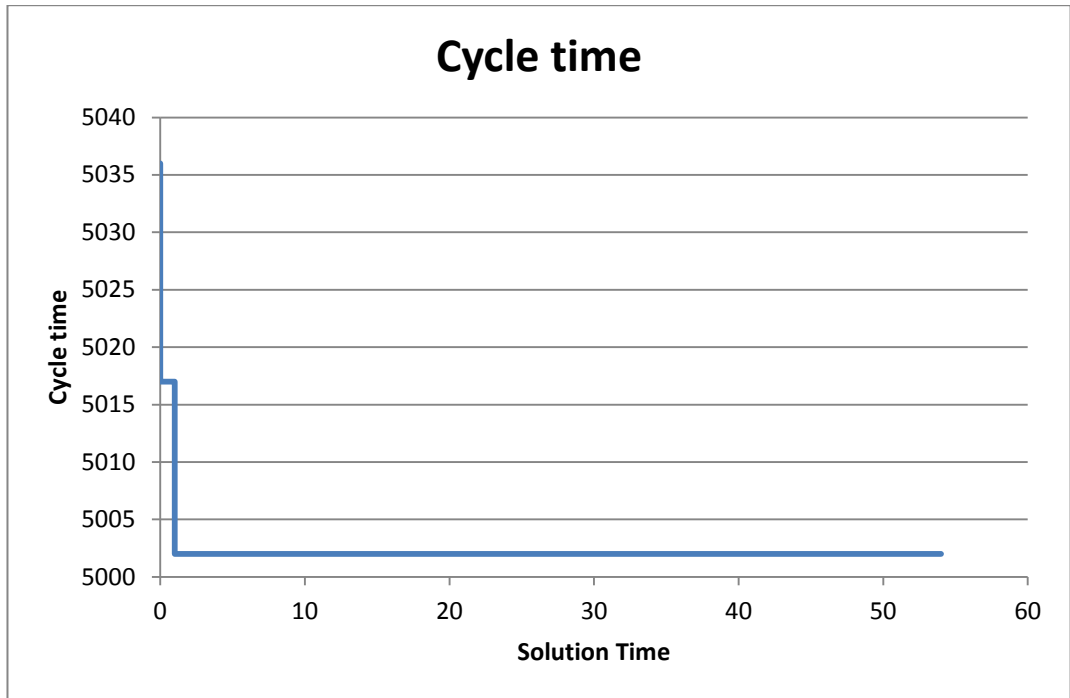


Figure B.7: Cycle Time vs Solution Time Convergence Graph for  $m = 3$ ,  $K = 3$

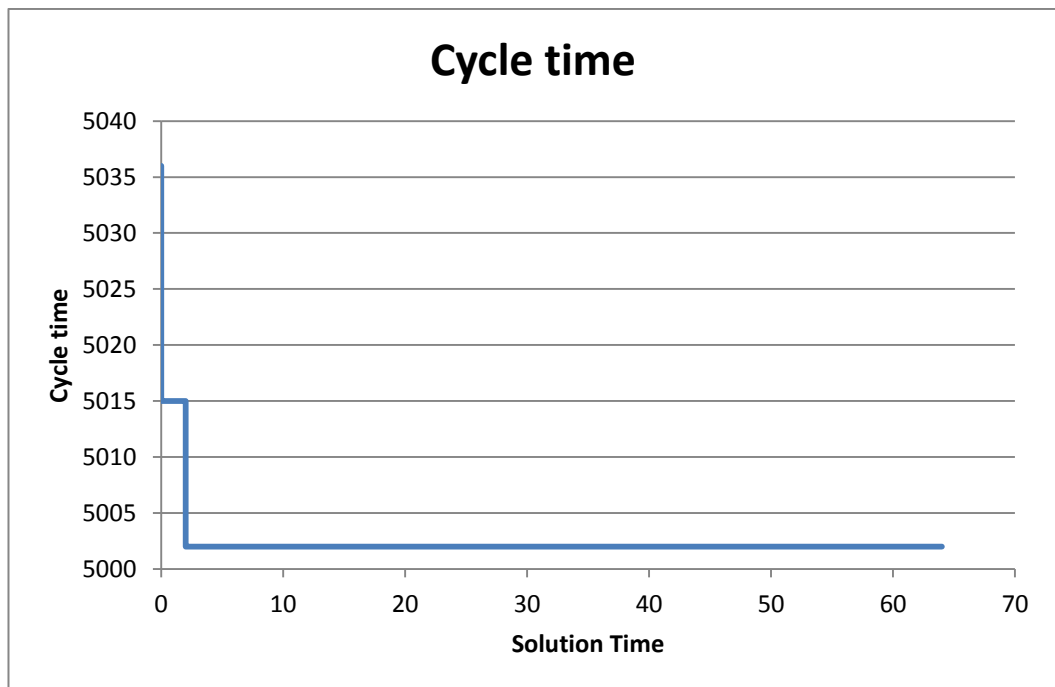


Figure B.8: Cycle Time vs Solution Time Convergence Graph for  $m = 3$ ,  $K = 4$

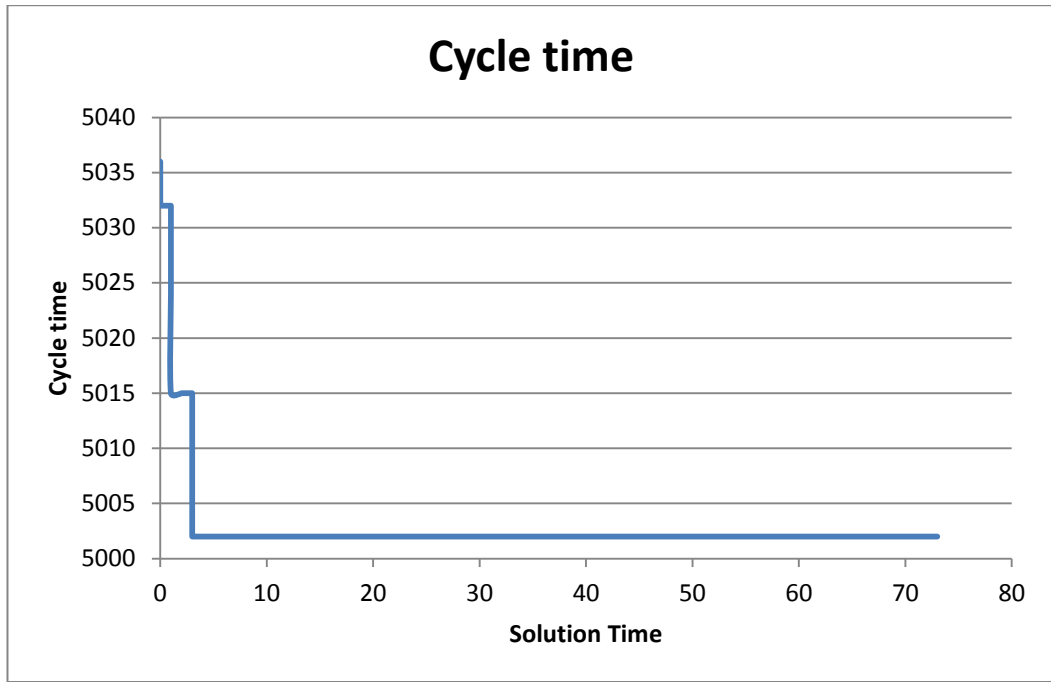


Figure B.9: Cycle Time vs Solution Time Convergence Graph for  $m = 3$ ,  $K = 5$

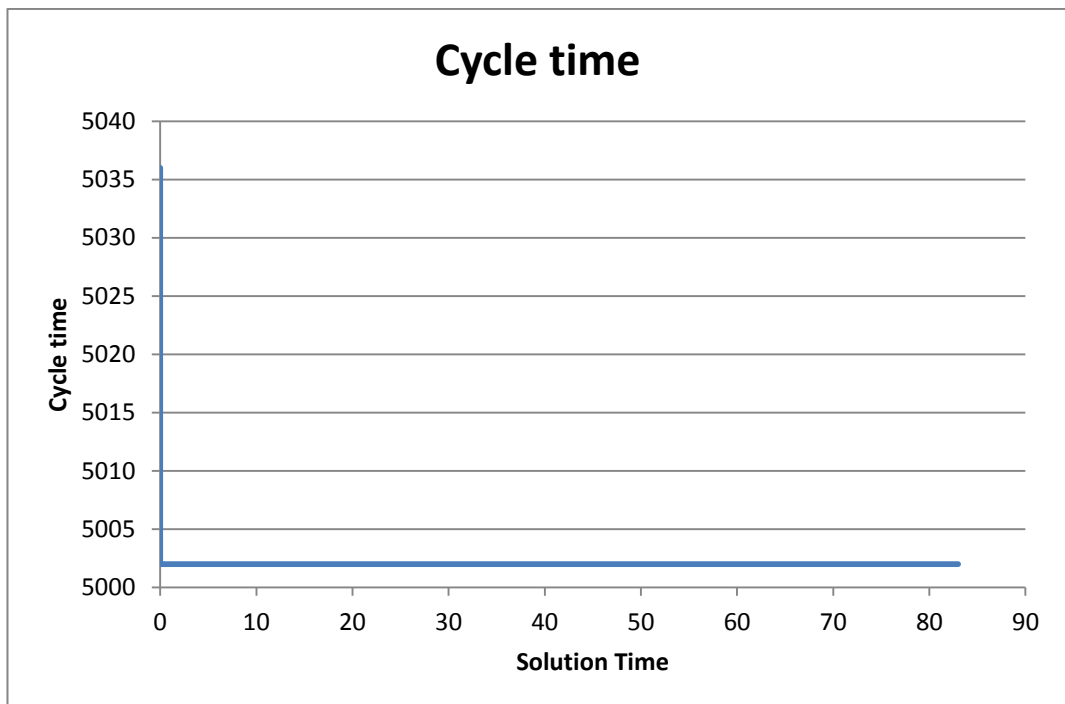


Figure B.10: Cycle Time vs Solution Time Convergence Graph for  $m = 3$ ,  $K = 6$

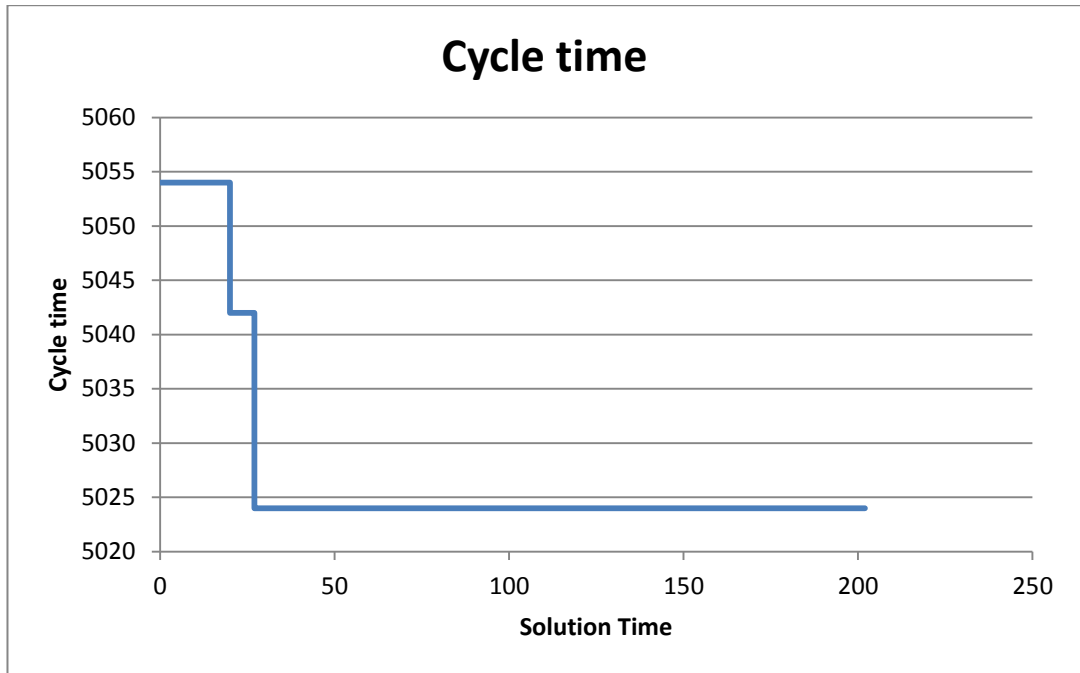


Figure B.11: Cycle Time vs Solution Time Convergence Graph for  $m = 4$ ,  $K = 1$

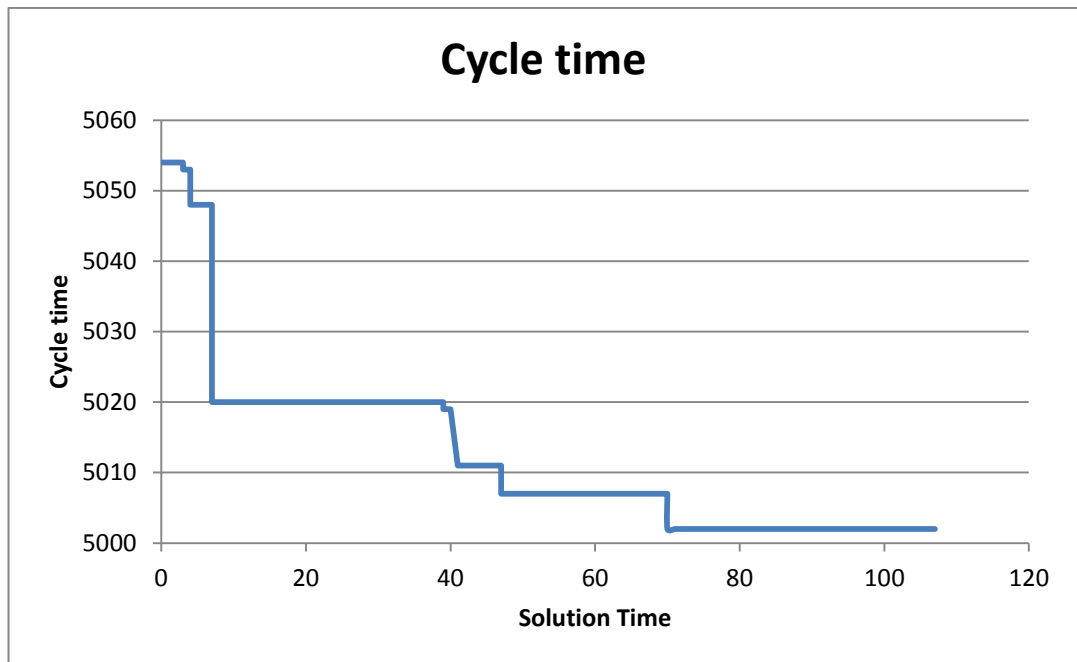


Figure B.12: Cycle Time vs Solution Time Convergence Graph for  $m = 4$ ,  $K = 2$

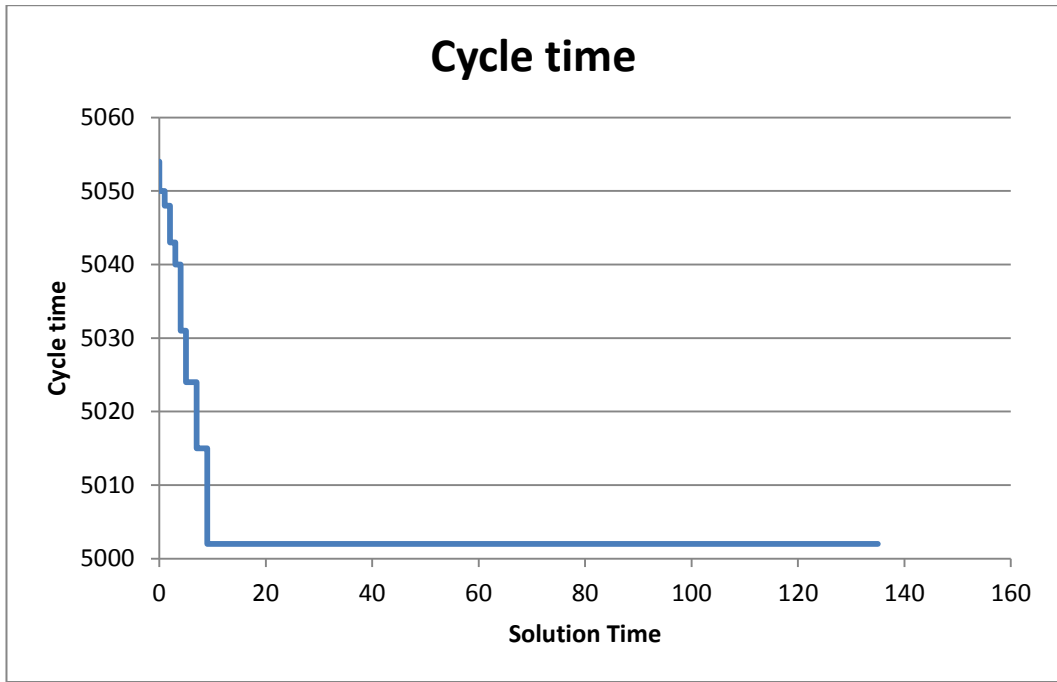


Figure B.13: Cycle Time vs Solution Time Convergence Graph for  $m = 4, K = 3$

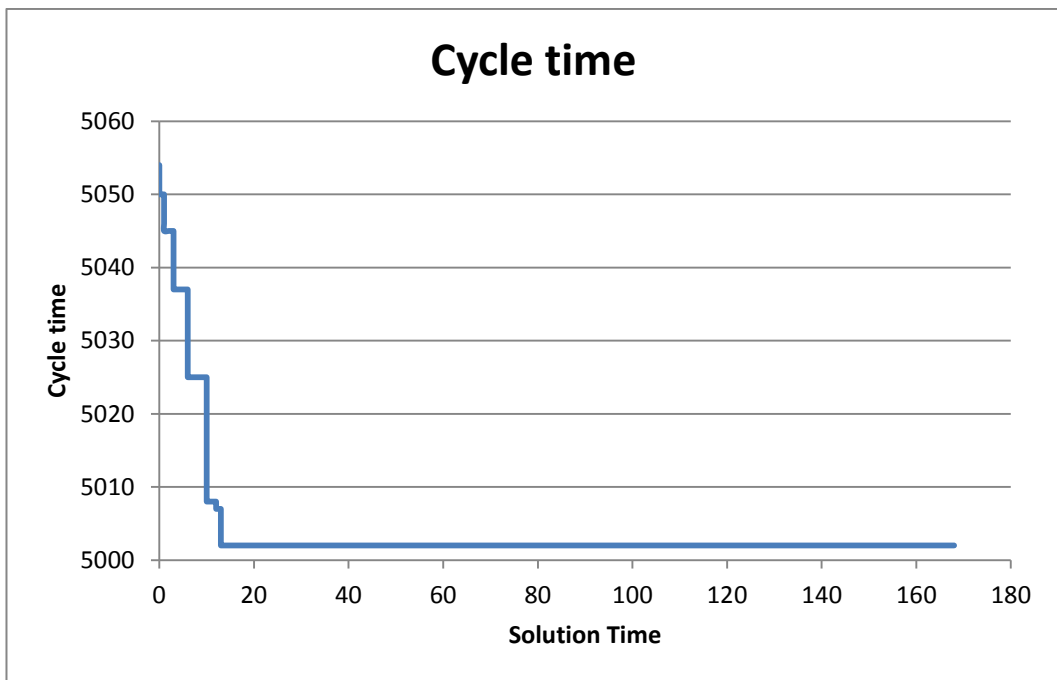


Figure B.14: Cycle Time vs Solution Time Convergence Graph for  $m = 4, K = 4$

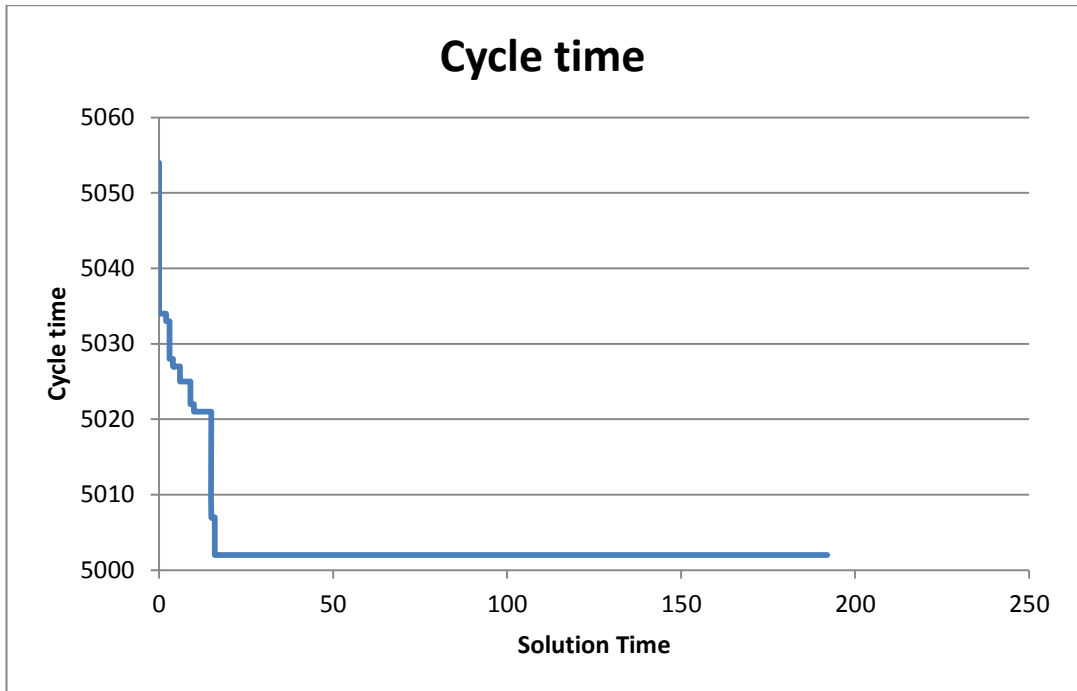


Figure B.15: Cycle Time vs Solution Time Convergence Graph for  $m = 4$ ,  $K = 5$

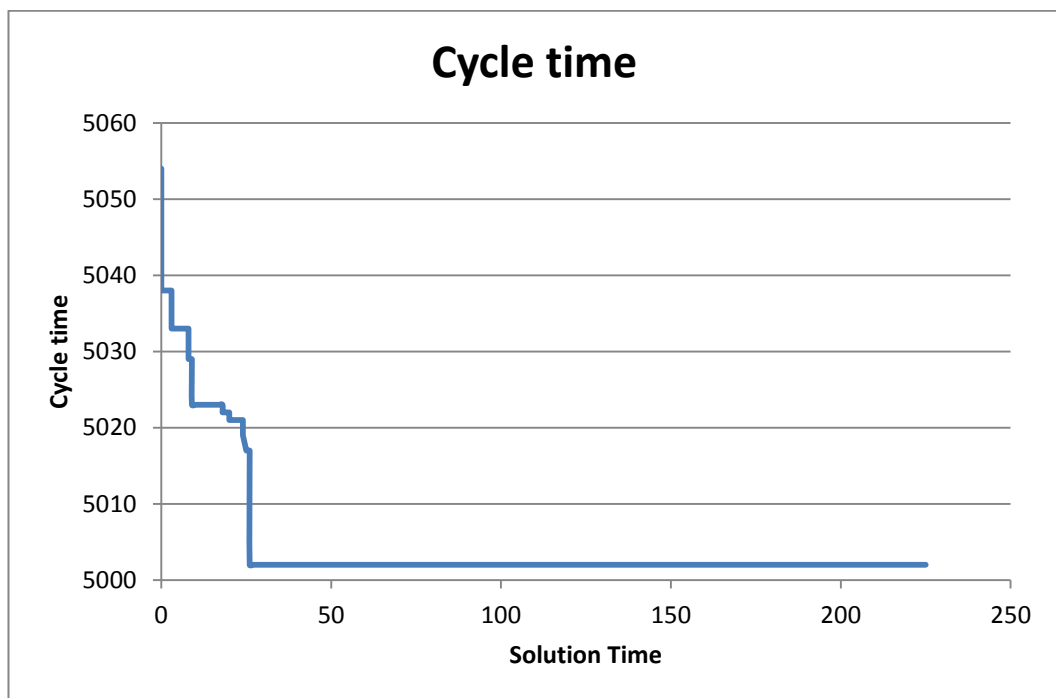


Figure B.16: Cycle Time vs Solution Time Convergence Graph for  $m = 4$ ,  $K = 6$

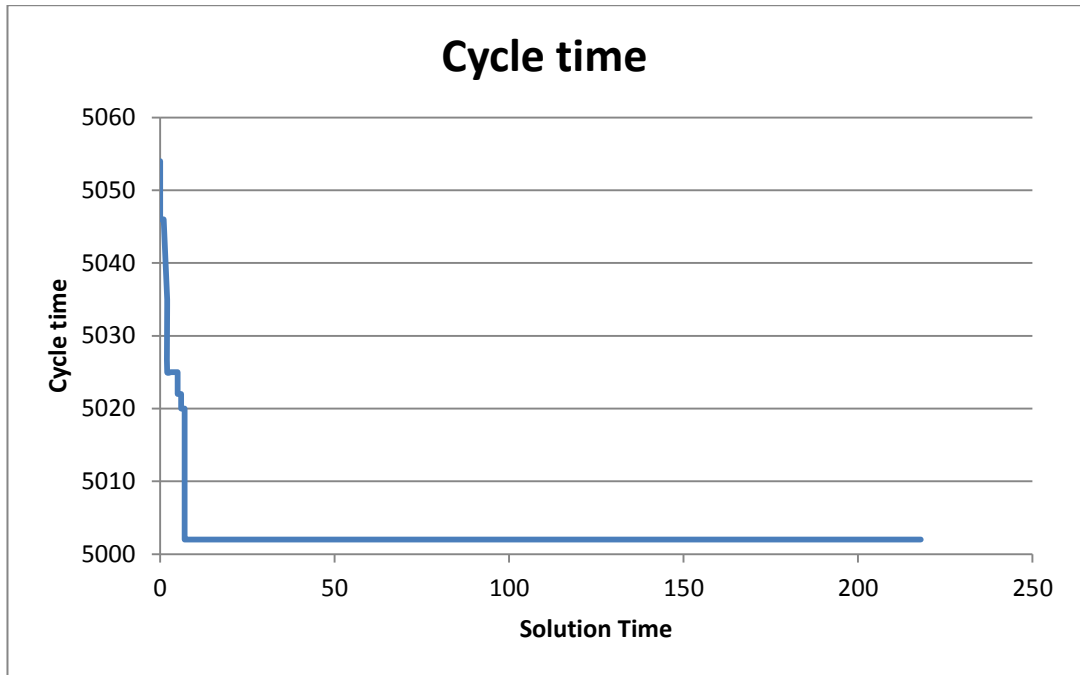


Figure B.17: Cycle Time vs Solution Time Convergence Graph for  $m = 4$ ,  $K = 7$

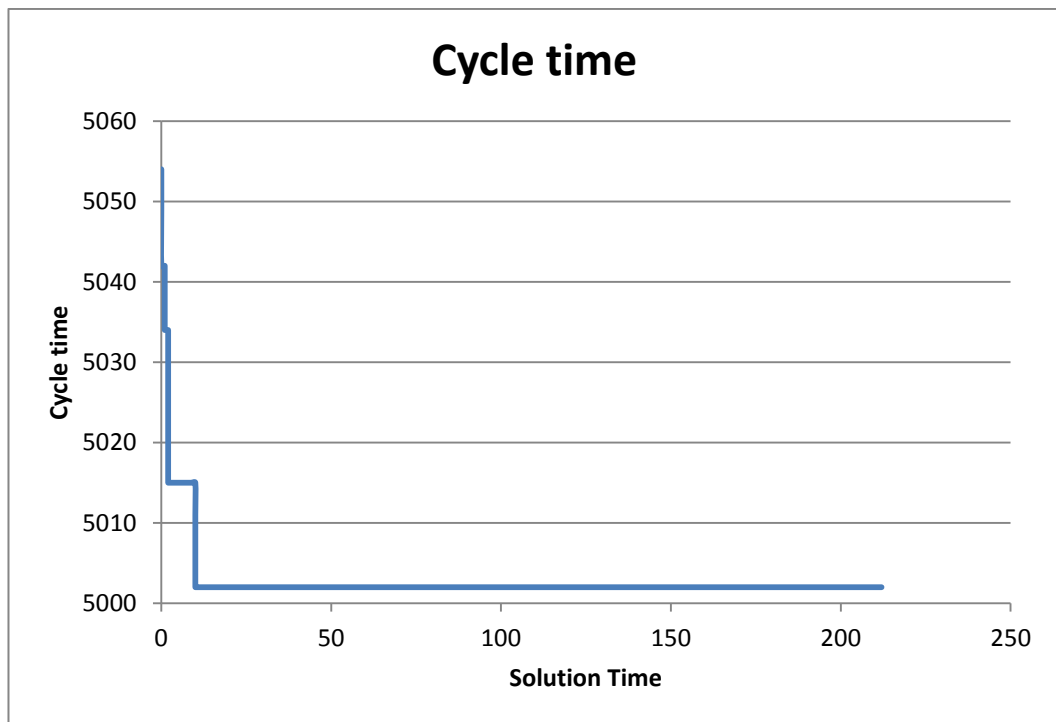


Figure B.18: Cycle Time vs Solution Time Convergence Graph for  $m = 4$ ,  $K = 8$

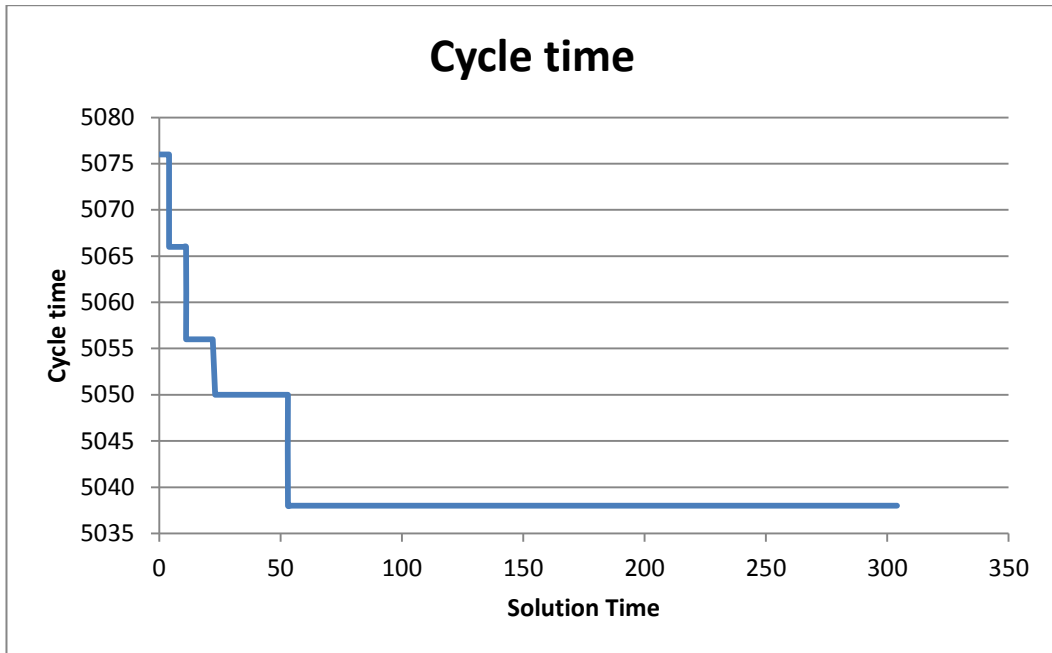


Figure B.19: Cycle Time vs Solution Time Convergence Graph for  $m = 5$ ,  $K = 1$

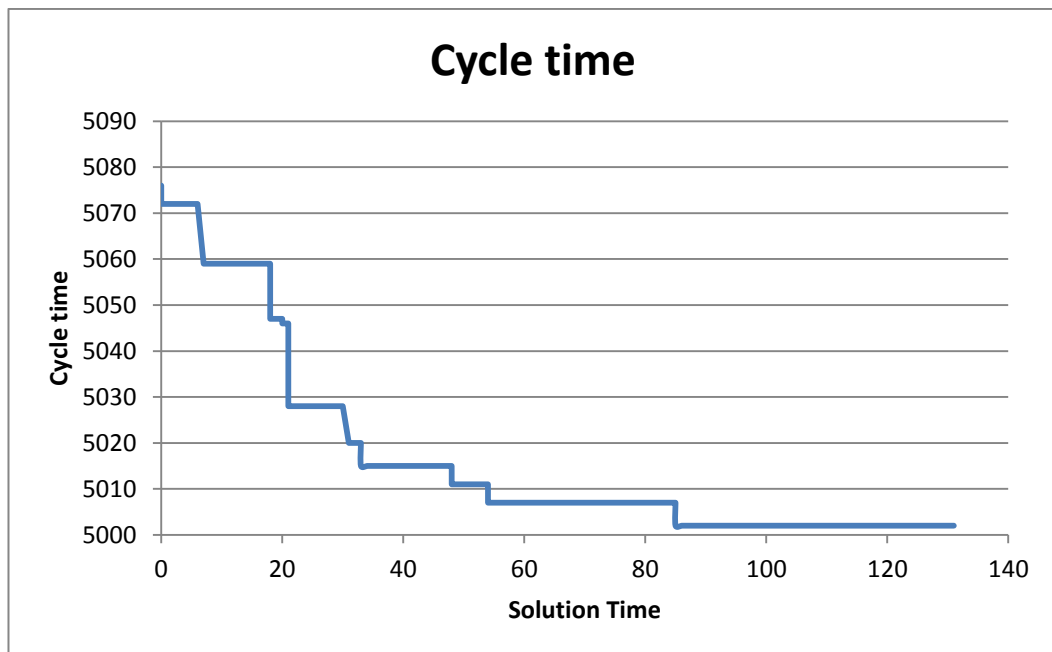


Figure B.20: Cycle Time vs Solution Time Convergence Graph for  $m = 5$ ,  $K = 2$



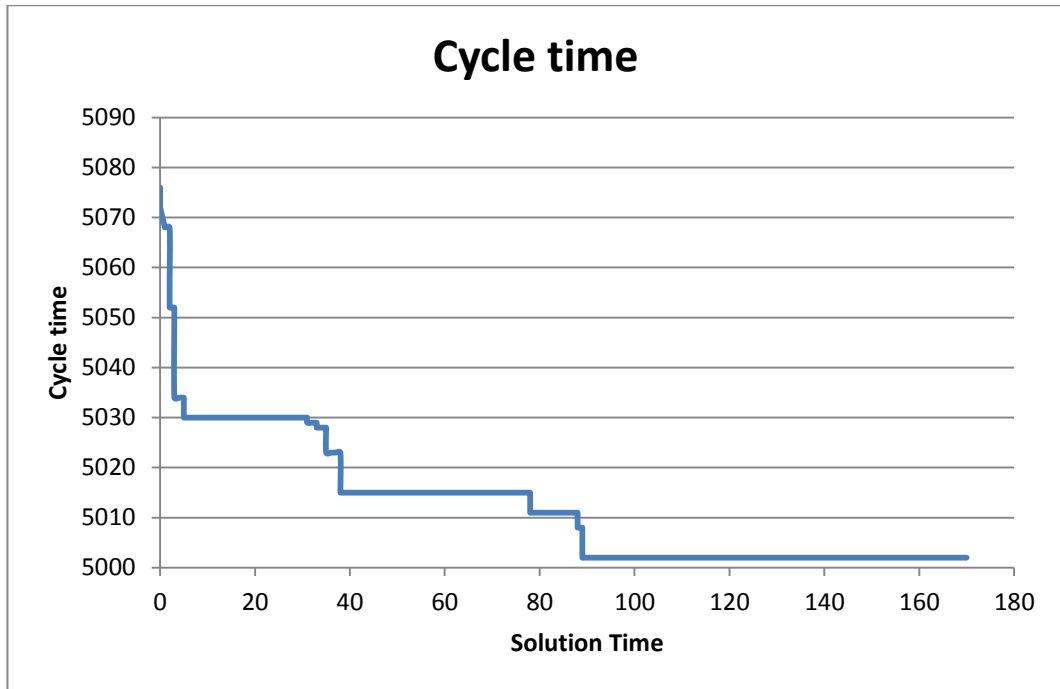


Figure B.21: Cycle Time vs Solution Time Convergence Graph for  $m = 5$ ,  $K = 3$

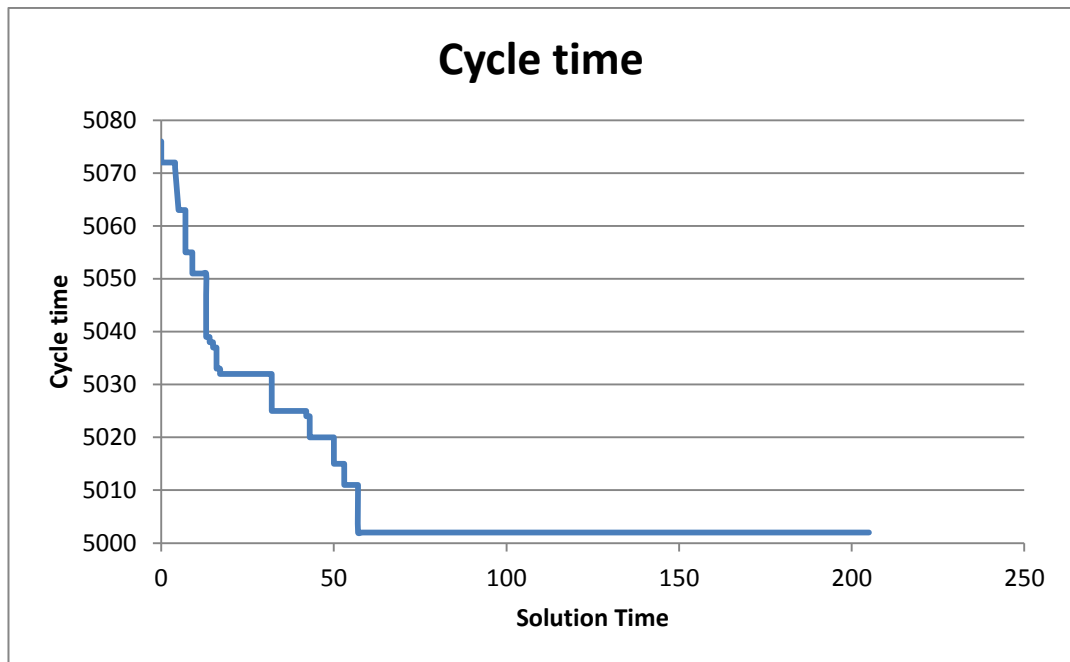


Figure B.22: Cycle Time vs Solution Time Convergence Graph for  $m = 5$ ,  $K = 4$

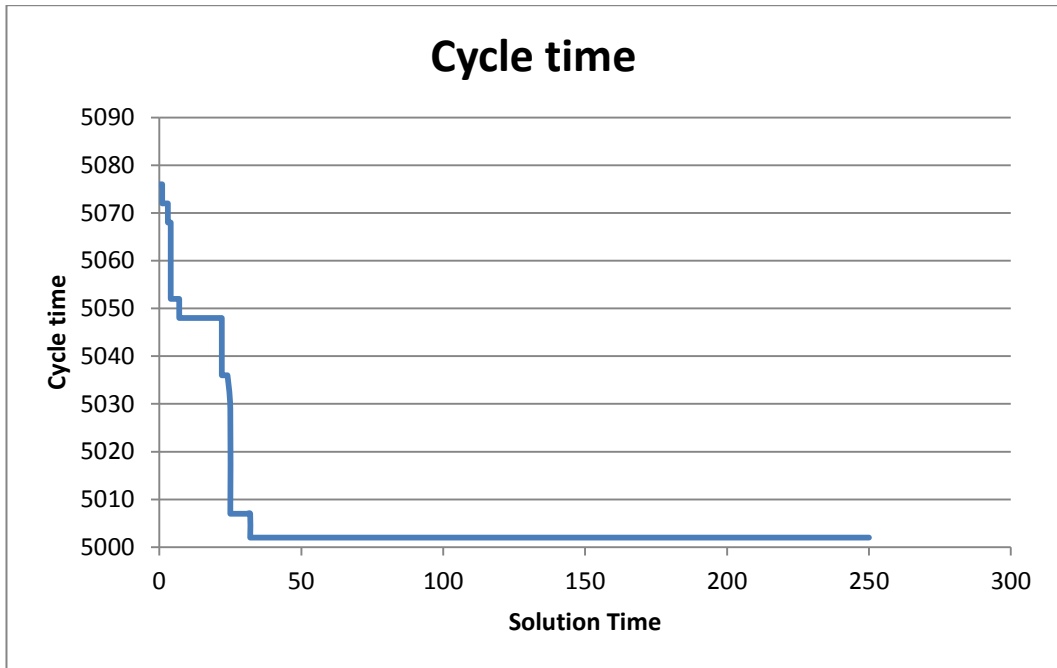


Figure B.23: Cycle Time vs Solution Time Convergence Graph for  $m = 5$ ,  $K = 5$

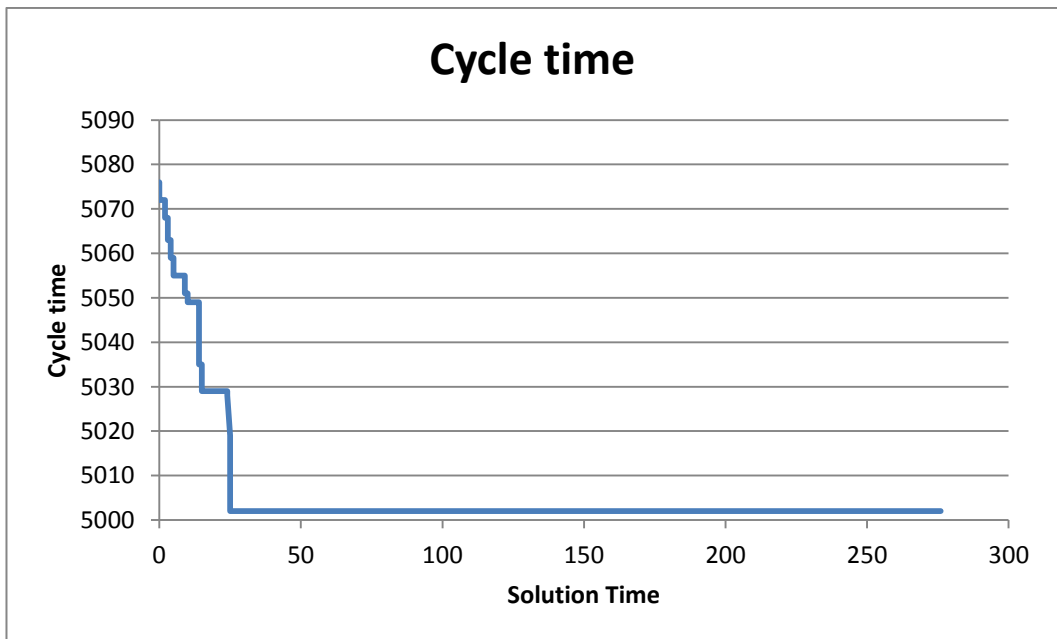


Figure B.24: Cycle Time vs Solution Time Convergence Graph for  $m = 5$ ,  $K = 6$

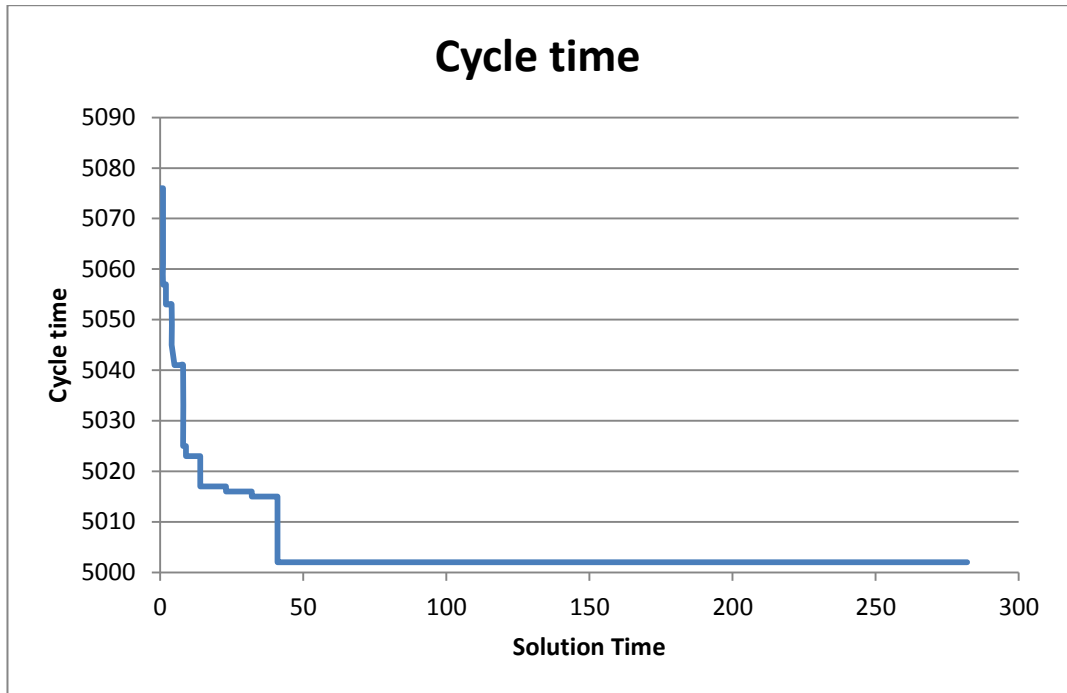


Figure B.25: Cycle Time vs Solution Time Convergence Graph for  $m = 5$ ,  $K = 7$

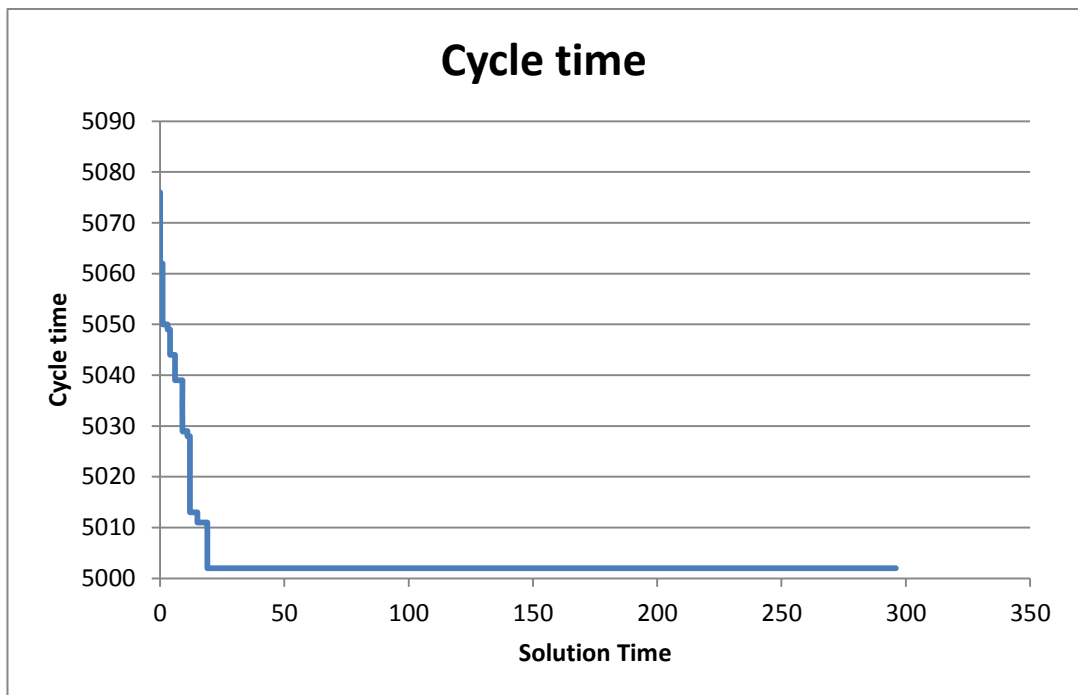


Figure B.26: Cycle Time vs Solution Time Convergence Graph for  $m = 5$ ,  $K = 8$

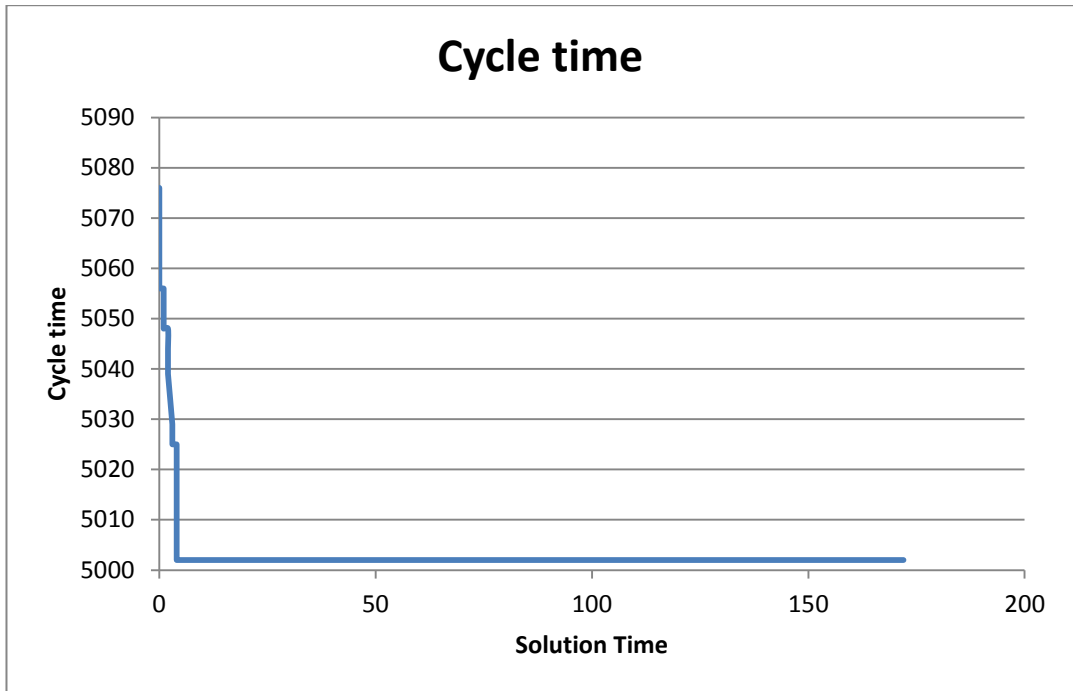


Figure B.27: Cycle Time vs Solution Time Convergence Graph for  $m = 5$ ,  $K = 9$

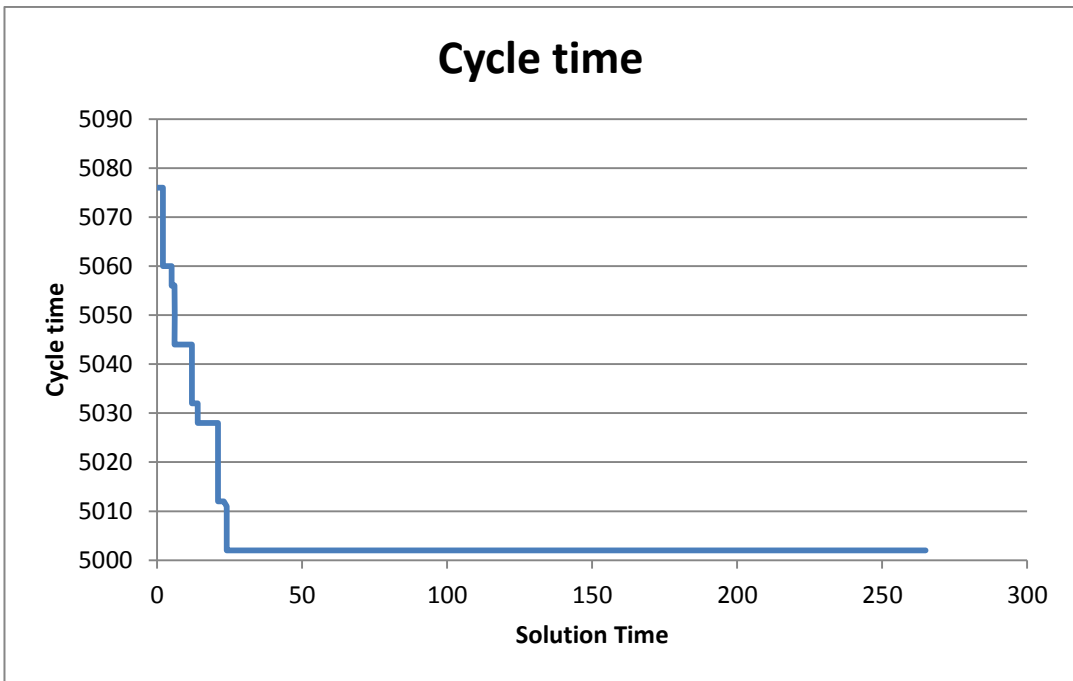


Figure B.28: Cycle Time vs Solution Time Convergence Graph for  $m = 5$ ,  $K = 10$