

**EARLY - TARDY SCHEDULING PROBLEM WITH
ENERGY CONSUMPTION CONSIDERATION**

BY

HUDAIFAH ABDULLAH SYAWAL

A Thesis Presented to the
DEANSHIP OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

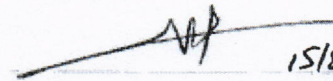
In

INDUSTRIAL & SYSTEM ENGINEERING

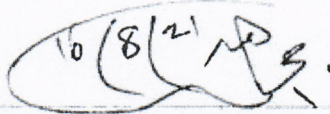
APRIL, 2021

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS
DHAHRAN- 31261, SAUDI ARABIA
DEANSHIP OF GRADUATE STUDIES

This thesis, written by **HUDAIFAH ABDULLAH SYAWAL** under the direction of his thesis advisor and approved by his thesis committee, has been presented and accepted by the Dean of Graduate Studies, in partial fulfillment of the requirements for the degree of **MASTER OF SCIENCE IN INDUSTRIAL & SYSTEMS ENGINEERING.**

 15/8/21

Dr. Umar M. Al-Turki
(Advisor)



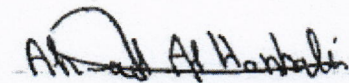
Dr. Mujahed AlDhaifallah
Department Chairman

 15/8/2021

Dr. Khaled Al-Shareef
(Member)



Dr. Suliman Saleh Al-Homidan
Dean of Graduate Studies

 (15/8/21)

Dr. Ahmad Al-Hanbali
(Member)

Date _____



© Hudaifah Abdullah Syawal

2021

I dedicate this thesis to my beloved;

Parents, Brothers, Sisters, and Teachers

For their support and encouragement through my life so far.

I also dedicate this thesis to my;

Wife and Son,

For their love, patient, support, and understanding |

ACKNOWLEDGMENTS

In the name of Allah, the Entirely Merciful, the Especially Merciful

All praises be to Allah, I am very grateful to Allah for the mercy and protection for me to complete this work as well as my master's program successfully. Peace and bless to the Prophet Muhammad, his family, his companions, and his followers.

I would like to express my gratitude to my thesis advisor, Professor Umar Al-Turki for his guidance, advice, and encouragement during my study. I extend my gratitude to my thesis committee members, Dr. Khaled Al-Shareef and Dr. Ahmed Al-Hanbali for their cooperation, interest, and valuable advice.

Special thanks to my parents, my wife, my son and my family for their motivation, support and patients. Extended thanks to all my friends, Brothers Tamim and Mehmed, and all Indonesian student who have been part of my life in KFUPM.

Finally, I thank the Government of the Kingdom of Saudi Arabia and King Fahd University of Petroleum & Minerals (KFUPM) for granting me scholarship and an opportunity to complete my graduate study.

TABLE OF CONTENTS

ACKNOWLEDGMENTS	V
TABLE OF CONTENTS	VI
LIST OF TABLES.....	IX
LIST OF FIGURES.....	X
LIST OF ABBREVIATIONS.....	XI
ABSTRACT	XII
ملخص الرسالة	XIII
 1 CHAPTER 1 INTRODUCTION	
1.1 Background	1
1.2 The E/T Scheduling Problem	3
1.2.1 Theorems for E/T Scheduling with Common Due Dates	6
1.2.2 Construction Approach for E/T with Distinct Due Dates	7
1.3 Energy Efficiency Strategies in Scheduling	8
1.4 Scope.....	10
1.5 Problem Definition	10
1.5.1 Problem Statement.....	11
1.5.2 Assumptions	12
 2 CHAPTER 2 LITERATURE REVIEW	
2.1 Research on E/T Scheduling Problem.....	13
2.2 MILP Model for E/T Scheduling Problem	14

2.3	Energy Consideration in Scheduling Problem	15
2.4	Integrated vs. hierarchical solution approach	17
2.5	Controlled Processing Time	18
2.6	Lexicographic Optimization	20
2.7	Augmented ε -Constraint Optimization	20
3	CHAPTER 3 ENERGY CONSIDERATION IN THE E/T SCHEDULING PROBLEM	
3.1	Energy Consumption on Basic E/T scheduling problem	24
3.2	Energy Consumption Ratio	27
3.3	Properties of the optimal schedule	30
4	CHAPTER 4 MIXED INTEGER LINEAR PROGRAMMING MODEL	
4.1	Formulation	39
4.2	Integrated objective functions	42
4.2.1	Weighted Approach	43
4.2.2	AUGMECON Approach	47
5	CHAPTER 5 THE ASCENDING SPEED LEVEL (ASL) HEURISTIC	
5.1	Notations	51
5.2	Procedure of ASL Method	52
6	CHAPTER 6 RESULTS AND CONCLUSION	
6.1	Design of Experiments	64
6.2	Comparison between The AUGMECON and The Weighted Method	66
6.2.1	Number of Points in the Pareto Solution Result	67
6.2.2	Computational Time to Produce Pareto Optimum	70
6.2.3	Testing the hypothesis	71
6.3	Comparison between the MILP Weighted Method and the ASL Heuristic	73

6.3.1	Results	73
6.3.2	Computational time to Produce Pareto Optimum	77
6.4	Conclusions.....	79
6.5	Recommendation	80
APPENDIX 1		82
APPENDIX 2		84
APPENDIX 3		87
APPENDIX 4		90
REFERENCES.....		93
VITAE		96

LIST OF TABLES

Table 1 Processing Time of Jobs at the lowest speed	25
Table 2. Three Different Speed Levels and Its Power Consumption	25
Table 3 Performance Measure of Three Different Scheduling Scenarios	26
Table 4. Power Consumption Data of 3 Different Machines.....	27
Table 5. Energy Consumption in Different Speed Processing	28
Table 6. Job Processing Time	44
Table 7. Speed level and power consumption	44
Table 8. Resulting Pareto solutions for example 4 using the weighted method	45
Table 9 Result of Example 4 using AUGMECON Method	49
Table 10 Job Processing Time of Example 5.....	60
Table 11. Speed Level and Its Power Consumption of Example 5	60
Table 12 Result of Example 5 using ASL Method	63
Table 13 The Example Data of Five Possible Speed Levels	65
Table 14 Example Data of The Experiment	66
Table 15 The Avg. Number of Optimum Point in The Pareto solution with 11 iterations	69
Table 16 The Average Computational Time of AUGMECON and Weighted Method (in second).....	70
Table 17 The Performance of ASL Optimization Method	74
Table 18 The Comparison of Different Speed Level Configuration	75
Table 19 The Avg. Computational time of Weighted Method and ASL Method (in Seconds).....	78

LIST OF FIGURES

Figure 1 Flowchart of the AUGMECON method [33]	23
Figure 2 Three Different Scheduling Scenarios.....	26
Figure 3 Pareto optimum of Example 4 Using Weighted Method	44
Figure 4 The Algorithm of the Weighted Method	46
Figure 5 Pareto optimum of Example 4 using AUGMECON Method.....	48
Figure 6 Mechanism of the AUGMECON Method.....	50
Figure 7 Flowchart of ASL Method.....	54
Figure 8 The Detail Iterations of Example 5 with $W=0.6$	61
Figure 9 Pareto optimum of Example 5 Using ASL Method	62
Figure 10 The Pareto Optimum for Dataset 21	68
Figure 11 The Average Computational time	72
Figure 12 Pareto Optimum of Experiment 112.....	76
Figure 13 The Avg. Computational Time of The Weighted Method	78

LIST OF ABBREVIATIONS

E/T	:	Earliness/Tardiness
MILP	:	Mixed Integer Linear Programming
TCT	:	Total Completion Time
TCTU	:	Total Completion Time Utopian
TCTN	:	Total Completion Time Nadir
TEC	:	Total Energy Consumption
TECU	:	Total Energy Consumption Utopian
TECN	:	Total Energy Consumption Nadir
TET	:	Total Earliness Tardiness
TETU	:	Total Earliness Tardiness Utopian
TETN	:	Total Earliness Tardiness Nadir
AUGMECON:		Augmented ϵ -Constrain
MOMP	:	Multi-Objective Mathematical Programming
RHS	:	Right Hand Side
ECR	:	Energy Consumption Ratio
SET A	:	Set of jobs that complete after the due date
SET B	:	Set of jobs that complete before or at the due date

ABSTRACT

Full Name : Hudaifah Abdullah Syawal
Thesis Title : The Early -Tardy Scheduling Problem with Energy Consumption Consideration
Major Field : Industrial and System Engineering
Date of Degree : June 2021

Managing a manufacturing process in effective, efficient and ecofriendly manner is the most concern of modern manufacturers. Just-In-Time production system is used widely in order to achieve an effective and efficient production by eliminating waste. It also improves customer's satisfaction by delivering the customer order on time. The Early / Tardy (E/T) scheduling problem is the straight forward implementation of JIT in the production plan. On the other side, reducing energy consumption is addressed to achieve eco-friendly manufacturing environment. Several strategies have been introduced in the literature such as turn on/off strategy, speed scaling, peak power management and so on. However, the literature about E/T scheduling problem with energy consideration is very limited. In this study, we developed a mathematical model and proposed three methodologies to optimize the scheduling problem. Pareto optimization is utilized in order to provide the decision maker with several alternative solutions. Based on the experiments, each method has advantages and disadvantages in terms of the optimality of the solution, the computational time, and the number of optimum points in the Pareto solutions. Also, several properties of the optimum solution are identified in this study.

ملخص الرسالة

الاسم الكامل: حذيفة عبد الله شوال

عنوان الرسالة: جدولة العمليات مع مراعاة التقديم والتأخير عن الموعد واستهلاك الطاقة

التخصص: هندسة النظم والصناعية

تاريخ الدرجة العلمية: رمضان 1442

تعد إدارة عملية التصنيع بكفاءة وفعالية وصديقة للبيئة هي الشغل الشاغل للشركات المصنعة الحالية. يتم استخدام نظام الإنتاج Just-In-Time على نطاق واسع من أجل تحقيق إنتاج فعال من خلال تقليص النفقات. كما أنه يحسن رضا العملاء من خلال تسليم طلب العميل في الوقت المحدد. إن مشكلة الجدولة المبكرة / المتأخرة (E/T) هي التنفيذ المباشر لـ JIT في خطة الإنتاج. وعلى الجانب الآخر، يتم تناول تقليل استهلاك الطاقة لتحقيق تصنيع صديق للبيئة. ولذلك تم تقديم العديد من الاستراتيجيات مثل استراتيجيات التشغيل / الإيقاف، وتغيير السرعة، وإدارة الطاقة القصوى وما إلى ذلك. ومع ذلك، فإن الأبحاث حول مسألة جدولة E / T مع مراعاة الطاقة محدودة للغاية. في هذه الدراسة، اقترحنا ثلاث طرق لمعالجة تلك المسألة كما تم استخدام طريقة باريتو من أجل تزويد صانع القرار بالعديد من الحلول البديلة ليتم اختيار المناسب منها. وبناءً على الاختبارات التي تم عملها لمقارنة الطرق الثلاث، تبين أن كل طريقة لها مزايا وعيوب من حيث أمثلية الحل، والوقت المطلوب للحل، وعدد نقاط باريتو الناتجة. وبالإضافة لكل ذلك تم استخلاص العديد من خصائص الحلول المثلى للمسألة .

CHAPTER 1

INTRODUCTION

In this chapter, we introduce the problem addressed in this thesis. The state of the current research in E/T scheduling problem with energy consideration is presented. The problem is stated and research goals are defined for filling the gaps in the literature of the problem.

1.1 Background

Lean manufacturing is a production methodology introduced by Toyota company [1]. The principle of this methodology is to minimize waste and eliminate non added value activities to decrease the production cost, increase the profit margin, and run manufacturing operations effectively and efficiently. One of the wastes come from inventory management [2]. Products that are finished earlier than the due date are stored until they are shipped to customers. It raises up the inventory cost by keeping finished products in the warehouse. In contrast, if the products are not ready by the due date and gets late in delivery to the customer, the manufacturer gets penalized for this lateness. Just-In-Time production was introduced to handle this problem. Integrated information technology in manufacturing can easily respond to each arrival order from the customer into the production line. However, the scheduling problem arises while the manufacturer tries to schedule all jobs to finish on-time.

Early/Tardy scheduling is the most suitable scheduling strategy for Just-In-Time production systems [3]. Here, we combine two scheduling performance measures. Both calculate the difference between the due date of a job and its completion time. Earliness happens when a job finishes before its due date and tardiness appears when the job completion time exceeds its due date. Clearly, the combination of these two performance measures is calculated by the total deviation of each job's completion time from its due date. E/T problem has several models [4]. The due date can be either common due date (similar for all jobs) or distinct due date (each job has its due date). The cost of being early may be equal to the cost of being tardy, or different but has common value, or totally different for each job. Dispatching rules and heuristic approaches for each E/T problem variation has been studied in the literature to minimize both earliness and tardiness of all jobs.

Another issue that should be considered by manufacturers while managing their production is the environment issue. Since some traditional manufacturing processes may harm the environment and the awareness of people on sustainability arise, many companies try to apply green production to achieve an environment-friendly manufacturing. Recycling, reusing and remanufacturing the product for second life cycle, reducing emission to the air, developing a new ecofriendly equipment with less energy consumption, developing green culture in the company, and other approaches are adopted. Also considering energy consumption while constructing production schedules can be done for this purpose.

Several strategies were introduced for energy consumption such as; on-off strategy of turning off the machine at each idle period of production [5], speed scaling strategy of varying the speed of the machine to process jobs in different conditions [6], avoiding peak

time power consumption to distribute energy consumption during a day [7], and combination of those strategies. However, including energy consideration into the scheduling objective is not straight forward since minimizing energy consumption is mostly contradicting with traditional performance measures in scheduling theory. Several studies were conducted to consider these objectives but very few consider the E/T scheduling objective. In this study, we consider the speed scaling strategy for the basic E/T scheduling problem. We assume that the machine can run on several speed levels. The speed level for each job is determined in a way that minimizes power consumption.

1.2 The E/T Scheduling Problem

The goal of E/T scheduling problem as it is applied in manufacturing is to finish all jobs exactly on their due dates. Costs incurs for any deviation between job completion time and its due date. Here, the objective is to minimize the total penalty costs. The notation used throughout this thesis is as follows [4]:

$n = \text{total nuber of jobs}$

$B = \text{set of jobs completing before or on the due date}$

$b = \text{number of job in } B$

$A = \text{set of jobs completing after the due date}$

$a = \text{number of job in } A$

$E_j = \text{earliness of job } j$

$T_j = \text{tardiness of job } j$

$P_j = \text{Processing time of job } j$

$d_j = \text{Due date of job } j$

$C_j = \text{Completion time of job } j$

$\alpha_j = \text{unit earliness cost of job } j$

$\beta_j = \text{unit tardiness cost of job } j$

$f(S) = \text{objective function for a schedule } S$

The E/T scheduling problem is translated into a mathematical model in order to conduct the optimization process. The earliness and tardiness for any job j is defined as follows:

$$E_j = \max \{0, d_j - C_j\} = (d_j - C_j)^+ \quad (1.1)$$

$$T_j = \max \{0, C_j - d_j\} = (C_j - d_j)^+ \quad (1.2)$$

The E/T objective function of a schedule S , while the cost $\alpha_j > 0$ and $\beta_j > 0$, can be defined mathematically as follows:

$$f(S) = \sum_{j=1}^n (\alpha_j E_j + \beta_j T_j) \quad (1.3)$$

$$f(S) = \sum_{j=1}^n [\alpha_j (d_j - C_j)^+ + \beta_j (C_j - d_j)^+] \quad (1.4)$$

In some E/T scheduling problems, due dates can be treated as decision variables where the objective is not only to optimize the job sequence but also to optimize the due date for each job. If due dates are given and all jobs have similar due dates ($d_j = d$), the objective is to

minimize the total deviation from a single common due date. In particular, the objective function can be formulated as follows:

$$f(S) = \sum_{j=1}^n [\alpha_j (d - C_j)^+ + \beta_j (C_j - d)^+] \quad (1.5)$$

If due dates are given and the due date is distinct for each job, equation (1.4) can be used to optimize the job sequence.

Costs due to earliness and tardiness can be similar when $\alpha_j = \beta_j = 1$, i.e., the model has symmetric E/T cost. In this case, the model becomes easier to solve. Mathematically, the objective function can be written as,

$$f(S) = \sum_{j=1}^n (E_j + T_j) = \sum_{j=1}^n |C_j - d_j| \quad (1.6)$$

For another case, earliness and tardiness are penalized with different costs since α represent the inventory cost and β represents the penalty for tardiness. Equations (1.4) or (1.5) are then used to represent this model.

The basic E/T scheduling problem is a special case in the group of E/T problem defined by Baker [4]. He studied the characteristics of this unique case where all jobs have a common due date ($d_j = d$) and equal costs of earliness and tardiness for all jobs ($\alpha_j = \beta_j = 1$). Here, the objective of minimizing the sum of earliness and tardiness of all jobs is written as:

$$f(S) = \sum_{j=1}^n (E_j + T_j) = \sum_{j=1}^n |C_j - d| \quad (1.7)$$

The literature considers two different cases when the due date is given (not a decision variable): restricted and unrestricted cases. The restricted case occurs when the due date d

is less than the completion of all early jobs ($\sum_j P_j$, $j \in \text{set } B$) while jobs are arranged in a V-shape order. Contrary, the unrestricted version occurs when the due date is large enough to accommodate jobs in set B. The focus of this thesis is on the basic E/T problem and the due date is considered to be unrestrictive.

1.2.1 Theorems for E/T Scheduling with Common Due Dates

Based on the literature [4], there are theorems that are used to find a solution for the early/tardy scheduling problem with equal weights and a common due date. Any scheduling problem satisfying this condition is called the basic E/T problem. For the unrestricted version (where the due date is either a decision variable or given a value that is large enough to accommodate the optimum schedule), the following theorems are proven to be true.

*“**Theorem 1:** Schedules without inserted idle times between successive jobs constitute a dominant set.”*

*“**Theorem 2:** An optimal schedule exists in which Jobs that complete on or before the due date are sequenced in LPT order, and jobs that start on or after due date are sequenced in SPT order.”*

*“**Theorem 3:** An optimal schedule exists in which some job completes exactly at the due date.”*

*“**Theorem 4:** An optimal schedule exists in which the b^{th} job completes at time d , where $b = \lceil n/2 \rceil$ ”*

Based on the theorems above, the algorithm to solve the basic E/T problem with unconstrained due date is as follows:

Step 1. Assign the longest job to set B

Step 2. Find the next two longest jobs, assign one to B and one to A

Step 3. Repeat step 2 until there are no jobs left, or until there is one job left, in which case it is assigned to either set A or B.

Step 4. Finally, order jobs in B in LPT and jobs in A in SPT order.

1.2.2 Construction Approach for E/T with Distinct Due Dates

In the case where due dates are distinct, inserted idle times may be desirable. Finding an optimal schedule can be broken down into two sub problems, determining the job sequence and scheduling inserted idle times [4]. A schedule can be partitioned into blocks, each block consists of contiguous jobs in the schedule. Insertion of idle times can be done between blocks, not within blocks. An optimal schedule holds the following condition: the first job in a block cannot be tardy unless it starts at time zero and the last job in any block cannot be early. The solution procedure begins by ordering all jobs in EDD sequence. Then, if possible scheduling the first job to be on time at its due date in the first block, otherwise, it starts at time zero. The next job j in the list is scheduled to start exactly at the completion of the previous job in the block. If that job is early when appended to the current block, a new block is created by shifting job j to complete on its due date. The last block is then

analyzed for any possible shifting in the block to minimize the cost. These steps are repeated until all jobs are assigned.

1.3 Energy Efficiency Strategies in Scheduling

Scheduling with energy consideration has attracted many researchers for developing a methodology for “green scheduling”. Many approaches were proposed to minimize the makespan and energy consecutively, especially for job shop and flow shop scheduling problems. The three main strategies to minimize energy consumption in production scheduling are; on-off strategy, speed scaling, and time of use electricity price [8].

The idea of on-off strategy is to reduce energy consumption of the production machine by turning it off on the idle periods of production. The machine will turn off when the idle time exceeds a certain threshold and turn on again when the next job starts processing [5]. Two important things should be considered on this strategy, determining the optimum threshold for turning off the machine and determining the maximum on-off cycle. Number of on-offs should be limited since this action will affect the lifetime of the machine.

Speed scaling strategy allows the job to be processed in different speed levels [6]. Higher machining speed will shorten the processing time but consume more energy. Conversely, lower machining speed will reduce the energy consumption but extend the processing time of the job. Processing some jobs with high-speed level might be necessarily for reducing the total tardiness of the schedule. Decreasing the speed level of some jobs can reduce the energy consumption of the schedule. Two decisions should be made, the sequence of the

jobs and the speed level for each job for minimizing earliness, tardiness and energy simultaneously.

Typically, daily electricity consumption has peak time on specific duration periods during the day. Power plants generate electricity based on the highest expected demand since they cannot increase their production capacity easily. Hence, there is a potential electricity loss during off-peak period since the electricity generated exceed the demand. The Idea of saving energy by shifting production to off-peak times and avoid peak-times [9]. In some countries, governments apply different electricity charges to encourage manufactures to run their production on off-peak times. So, this strategy can increase the utilization of the generated electricity and reduce electricity cost as well as reduce pollution from power plants.

Several studies about green scheduling were conducted by many researches. Mohktari and Hasani [10] designed an energy efficiency scheduling algorithm under flexible job shop environment. An enhanced evolutionary algorithm introduced to find global optimum in a multi-objective optimization model to solve three objectives; maximizing the total availability of the system, minimizing the total energy cost and minimizing the total makespan. Wu and Sun [8] studied a flexible job shop scheduling problem and proposed a NSGA method (Non-Dominated Sorted Genetic Algorithm) for energy saving. Particularly, it optimizes the makespan, the number of turning on/off machines and the energy consumption simultaneously. Zandi, et al [11] introduced a bi-objective mathematical formulation for green scheduling of parallel machines with total makespan and total energy consumption as objectives. The exact Pareto frontier for these two objectives obtained using a heuristic algorithm.

1.4 Scope

The problem in this thesis is scheduling jobs on a single machine with the objective of minimizing the deviation of job completion time from a common due date and minimizing energy consumption. The machine can process jobs at several speed levels which will affect the processing time and the amount of energy consumed. Due to machine efficiency issues, reducing job processing time will increase energy consumption exponentially per unit time.

1.5 Problem Definition

The Early/Tardy Scheduling problem with energy consumption is addressed in this thesis. The focus is to integrate the application of speed scaling on the basic E/T scheduling problem. Speed scaling is a strategy where a job can be processed in different levels of speed with different levels of power consumption. The power consumption is assumed to be an exponential function of the speed. The basic E/T scheduling problem is a special case of the general E/T problem as defined by Baker and Trescent [4] having a common due date ($d_j=d$) and equal penalties of earliness and tardiness for all jobs ($\alpha_j=\beta_j=1$). Moreover, the basic E/T problem is a single machine problem in which all jobs are ready at time zero to be scheduled with no interruptions during processing.

Speed scaling strategy has the benefit of reducing energy consumption to process a set of jobs. Processing a job in low-speed machining will reduce energy consumption while setting high-speed machining to process a job will increase the energy consumption. This

condition arises when the energy consumption rate increases significantly as the processing speed increases. On the other hand, processing jobs in low-speed results in a longer processing time so that total earliness and tardiness of the schedule may increase.

Clearly, it is not easy to construct a schedule that minimizes total energy consumption and total earliness and tardiness simultaneously since both objectives are contradicting in nature. To achieve a minimum of total energy consumption, the speed level for all jobs should be set on the lowest machining speed level but this setting has worse E/T cost. Contrarily, in order to achieve minimum E/T cost, all jobs should be set on the highest speed level but this setting has the largest total energy consumption. To compromise both objectives, the speed level for each individual job should be selected properly in order to achieve a reduction in total energy consumption without sacrificing too much E/T performance.

1.5.1 Problem Statement

Given a set of n jobs, $j_1, \dots, j_n$ are to be scheduled for processing on a single machine. Each job has a processing time P_j at the lowest machining speed level v_1 , and can be processed at s different machining speed levels, $v_1, \dots, v_s$. Each machining speed level consumes different power energy π_i . Both v and π are increasing with s . All jobs have a common due date, $d_j = d$.

The size of the set of feasible solutions for this problem with n -jobs and s -speed levels is $n! * s^n$. The aim of this study is to develop a method for constructing a schedule that optimally minimize both objectives.

1.5.2 Assumptions

In order to study the problem mentioned above and define the scope of research, several common assumptions in scheduling are applied as follows:

- No preemption is allowed while processing a job
- No cancellation when the job starts.
- The Machine is always available to process a job
- Processing time is known in advance
- All jobs are ready at time zero
- The machine has a feature to be operated at several discrete speed levels
- A Job can be processed in only one of the possible speed levels
- Once a schedule and speed levels are set, it can't be changed while processing.
- The time to change the speed level of the machine is relatively small compared to the processing time hence are neglected

CHAPTER 2

LITERATURE REVIEW

In this chapter, a review of the literature is conducted in order to construct a solid understanding and knowledge regarding the problem addressed in this thesis, and identify gaps in the current research on this topic.

2.1 Research on E/T Scheduling Problem

Rocholl and Monch [12] utilize Mixed Integer Linear Programming to solve the E/T scheduling problem on single machine with multiple orders per job and unrestricted common due date. Simple heuristic based on dispatching rules and Genetic Algorithm was hybridized to solve a scheduling problem in semiconductor wafer fabrication facility. Syahriari, M., et al. [13] developed a bi-objective mixed integer programming model on JIT single machine scheduling problem with a periodic preventive maintenance. They implemented multi-objective particle swarm optimization (MOPSO) to minimize total makespan and earliness and tardiness simultaneously.

Corominas, A., et al. [14] proposed a mathematical model and a heuristic approach based on multistage graph-based procedure for solving a JIT scheduling problem. They considered a flexible job shop with machine and time-dependent processing costs. Allaoua H., et al. [15] proposed mathematically enhanced genetic algorithm (MEGA) to solve the basic E/T problem with common due date. They identified several new mathematical

properties of single machine E/T problem and combine it with genetic algorithm to enhance the performance of the searching process and the quality of the solution.

Radhakrishnan S, et al. [16] conducted a study on parallel machines E/T scheduling problem with distinct due dates considering sequence dependent setup times. The Meta-heuristic, Simulated Annealing is used to minimizing the total absolute deviation of completion times from their due dates. Khader M. [17] used Tabu Search method to solve the single machine E/T problem with distinct due dates and compared it with a greedy approach. He considered different penalty costs for earliness and tardiness.

Rabadi, et al. [18] introduced SAPT method (Shortest Adjusted Processing Time First) to find the near optimal solution for the single machine E/T scheduling problem with unrestricted common due date with sequence dependent setup times. A Hybrid Genetic Algorithm was proposed by Valente, et al. [19] on single machine E/T problem to improve an initial schedule and compare the new approach with dispatching rules and local search methods. Hepdogan, et al. [20] studied the single machine E/T problem with common due date and sequence dependent setup times and proposed a solution method, namely Meta-RaPS (Meta-heuristic for Randomized Priority Search). The proposed method was compared with another method for minimizing the sum of earliness and tardiness.

2.2 MILP Model for E/T Scheduling Problem

An alternative methodology to solve the E/T scheduling problem is using mathematical models or mixed-integer linear programming. The MILP formulation can be based on the

time index of processing period decision, time index of starting times decision, job sequence position decision, or precedence job decision [3]. Ronconi [21] proposed a mixed-integer programming models to minimize the total earliness and tardiness under the flow shop environment. Two families of the MILP model are introduced for the flow shop scheduling with unlimited buffer and zero buffer. Low, et al [3] proposed a MILP model for the single machine E/T scheduling with a common due date and unavailability interval. For the problem addressed in his research, formulating MILP using the time index of the starting time of jobs was faster than others.

2.3 Energy Consideration in Scheduling Problem

Energy consumption during processing jobs can be split into several types. Corominas [14] use the different cost of energy at peak time and at valley time. Electricity cost at peak hours is charged higher than valley hours. Two integrated solution approaches namely HeurJSP and Binary Integer Programming are used to minimize the total cost to process all jobs. Fang [6] breakdown energy consumption into basic energy, idle energy, and cutting energy. A multi-objective mixed integer programming was utilized as an integrated solution approach to minimize completion time, peak total power consumption, and carbon footprint in a flow shop environment. Cumulative energy to process a job is calculated by multiplying each power energy type with its related duration as shown in the equation below:

$$E_{total} = P_{basic} \cdot t_{basic} + P_{idle} \cdot t_{idle} + P_{cutting} \cdot t_{cutting} \quad (2.1)$$

Zandi [11] split power consumption into two types, power consumption during idle time and during processing time. An integrated solution approach using NBI, Normal boundary intersection, was utilized to minimize the total completion time as well as the total energy consumption. Mouzou & Yildirim [5] divide energy consumption into power idle, power processing, and energy setup (energy to turn on and off the machine) while minimizing tardiness and energy consumption on a single machine with a unique release date for each job. Li, et al [22] consider machining energy E_m and non-machining energy (stand-by energy) E_s as shown in equation (2.2) to calculate the energy consumed by machine tools in a flexible job shop.

$$E = E_m + E_s \quad (2.2)$$

Hence, for scheduling with contiguous jobs, standby energy or idle energy can be eliminated while calculating total energy consumption since there is no idle time arising in these schedule types.

In some applications, a job can be processed in several machining speed levels. Here, machining or processing energy is varying depend on the speed level. Speed level not only has an effect on machining power energy consumption but also affect the processing time of the job. In a basic single machine scheduling with multiple speed levels, the energy consumption of processing a job is calculated as follows:

$$EC_j = \pi_i \cdot p_{ji} = \pi_i \cdot \frac{P_1}{v_i} \quad (2.3)$$

Where v_i is the speed at level i , where $v_i \geq 1$ and lowest speed at $i=1$, π_i is the power consumption of processing job at speed level i , p_{ji} is processing time of job j if processed with speed level i , and P_{j1} is processing time of a job at the lowest speed level.

2.4 Integrated vs. hierarchical solution approach

Considering energy consumption in scheduling problems is not straight forward. Two solution approaches used by researchers; integrated solution approach or hierarchical solution approach.

Integrated solution approach tackles both minimizing energy consumption and maximizing scheduling performance simultaneously. Researchers express energy consumption with unit cost and integrate it with penalty costs arising by common scheduling performance measures. If the cost of energy consumption and common scheduling performance cannot be integrated directly, both objectives then are normalized with weights for each objective and solved as a multi-objective problem.

On the other hand, the hierarchical solution approach tackles the two objectives separately in two stages. In the first stage, jobs are sequenced to optimize a given scheduling performance. Then the proposed sequence is treated to minimize energy consumption. This approach is commonly applied in scheduling with dynamic release dates problems and/or under flow shop or job shop environment [23].

Corominas [14] used cost function approach to solve JIT scheduling in a flexible job shop with machine and time-dependent processing cost as shown in equation (2.4) below:

$$Min. [z] = \sum_{j=1}^n [C_E(E_j) + C_T(T_j) + U_j] \quad (2.4)$$

where : $C_E(E_j)$ is the earliness cost of job j, $C_T(T_j)$ is the tardiness cost of job j, and U_j is the cost of energy required to process job j.

In some cases, cost of energy consumption cannot be integrated directly with the cost arisen by traditional scheduling performance measures, i.e., energy consumption cost is not comparable to the cost of other scheduling performance measures, or too high compared to the cost of traditional performance measures. Here, researchers tend to use another approach. For example, Das & Denis [24] introduced the Normal Boundary Intersection (NBI) approach to combine two objectives that are normalized by the Nadir point as the lower limit, and the Utopian point as the upper limit, in order to integrate total completion time (TCT) with total energy consumption (TEC) as seen in equation (2.5) below:

$$z = w \frac{TCT - TCT^U}{TCT^N - TCT^U} + (1 - w) \frac{TEC - TEC^U}{TEC^N - TEC^U} \quad (2.5)$$

where: TCT^U is the Utopian point or optimal value of each objective, and TEC^N is the Nadir point or worst value of each objective, w and $(1 - w)$ are weighted value of both objectives with $0 \leq w \leq 1$. This approach is common in multi-objective optimization as “Upper Bound” and “Lower Bound”.

2.5 Controlled Processing Time

Commonly, job processing times are known in advance and are fixed. However, in many cases, processing time of job can be accelerated by adding more workers or running the

machine in higher speeds. This feature is considered by many researchers while constructing optimum schedules. Several terms have been introduced by the researchers for this issue such as “compressible processing time”, “controllable processing time”, and “speed level strategy”.

Cheng TCE, et al, [25] introduced the term of “compressible processing time” for jobs that can be speeded up with equal unit compression penalty per time unit for the single machine scheduling problem. The objective is to minimize the total penalty cost of earliness, tardiness, due date and compression job. The term “controllable processing time” is used by Vicson R.G. [26], Panwalker & Rajagopalan [27], Chen, et al [28] and Shabtay D. & Steiner G. [29]. In either “compressible processing time” or “controllable processing time”, the reduction of job’s processing time, linear cost function are considered. Hence, for E/T performance measure, there is a property for optimum condition in which job are not partially compressed, either fully compressed or not compressed.

Other researchers use the term of “speed scaling strategy” to study the scheduling problem under adjustable job processing times. This is originally introduced by Yao, et al [30]. Kwon and Kim [31] considered speed scaling with continuous speed level while Bansal, et al [32] and Fang, et al [33] considered discrete speed levels. Alternatively, “speed scaling strategy” is assumed that the cost of the power consumption is not linear function of the speed but increasing exponentially. Hence, the job may be compressed partially on the optimum solution.

2.6 Lexicographic Optimization

Lexicographic optimization is a method for solving a multi-objective mathematical model. It starts by solving the problem with respect to the first objective, then solving the problem with respect to the second objective with the solution of the first objective as a constraint. Subsequently, the problem is solved with respect to the third objective with the solution of the previous two objectives as constraints. This continues for the remaining objectives in the same manner. Lexicographic optimization was introduced by Mavrotas [34] to construct the efficient payoff table with only Pareto optimal solution for each objective function in multi-objective mathematical programming (MOMP) problems.

The payoff table is the most common approach to calculate the possible range of values of objective functions. Consisting of best value (utopian point) and worst value (nadir point) as the result of a series of n individual objective functions. Utopian value can be calculated as the optimal value of individual optimization while nadir value is usually approximated with the minimum value in the corresponding column related to its objective function (see e. g. [35], [36]).

2.7 Augmented ϵ -Constraint Optimization

The ϵ -constraint method or epsilon constraint method is one of the widely used methodologies to address the MOMP problem even for relatively large size problems. This method is a family of solution generating methods (or posterior method) that provide the

decision-maker a set of Pareto optimum solutions [34]. Mathematically, for MOMP problem below:

$$\text{Min } (f_1(x), f_2(x), \dots, f_n(x))$$

s. t.

$$x \in S$$

Where $f_1(x), f_2(x), \dots, f_n(x)$ are the n objective functions, x is the vector of decision variables, and S is the feasible region. The ε -constraint method of the above problem can be written as:

$$\text{Min } f_1(x)$$

s. t.

$$f_2(x) \leq e_2,$$

$$f_3(x) \leq e_3,$$

...

$$f_n(x) \leq e_n,$$

$$x \in S$$

The Pareto solution can be obtained by varying the RHS of the ε –constrained related to objective function i (e_i) in the range of its possible value.

Considering several drawbacks of the conventional ε -constraint method, a novel version of this method was proposed by Mavrotas [34] called Augmented ε -constraint (AUGMECON). This is done to avoid the generation of weak Pareto optimal solutions

generated by implementing lexicographic optimization, guarantee efficiency of the obtained solutions, and accelerate the whole process with an early exit from the loops procedure and avoid redundant iterations. Hence, The AUGMECON model for MOMP mentioned above is modified as:

$$\text{Min } (f_1(x) + \text{eps} \times (s_2/r_2 + s_3/r_3 + \dots + s_n/r_n))$$

s. t.

$$f_2(x) - s_2 = e_2,$$

$$f_3(x) - s_3 = e_3,$$

...

$$f_n(x) - s_n = e_n,$$

$$x \in S \text{ and } s_i \in R^+$$

Where *eps* is a relatively small number (between 10^{-3} and 10^{-6}), s_i is the slack value of the constrained objective function i , r_i is the range of the i th objective function as calculated from the payoff table (see [33]).

The procedure to implement the AUGMECON is shown in figure 1 below:

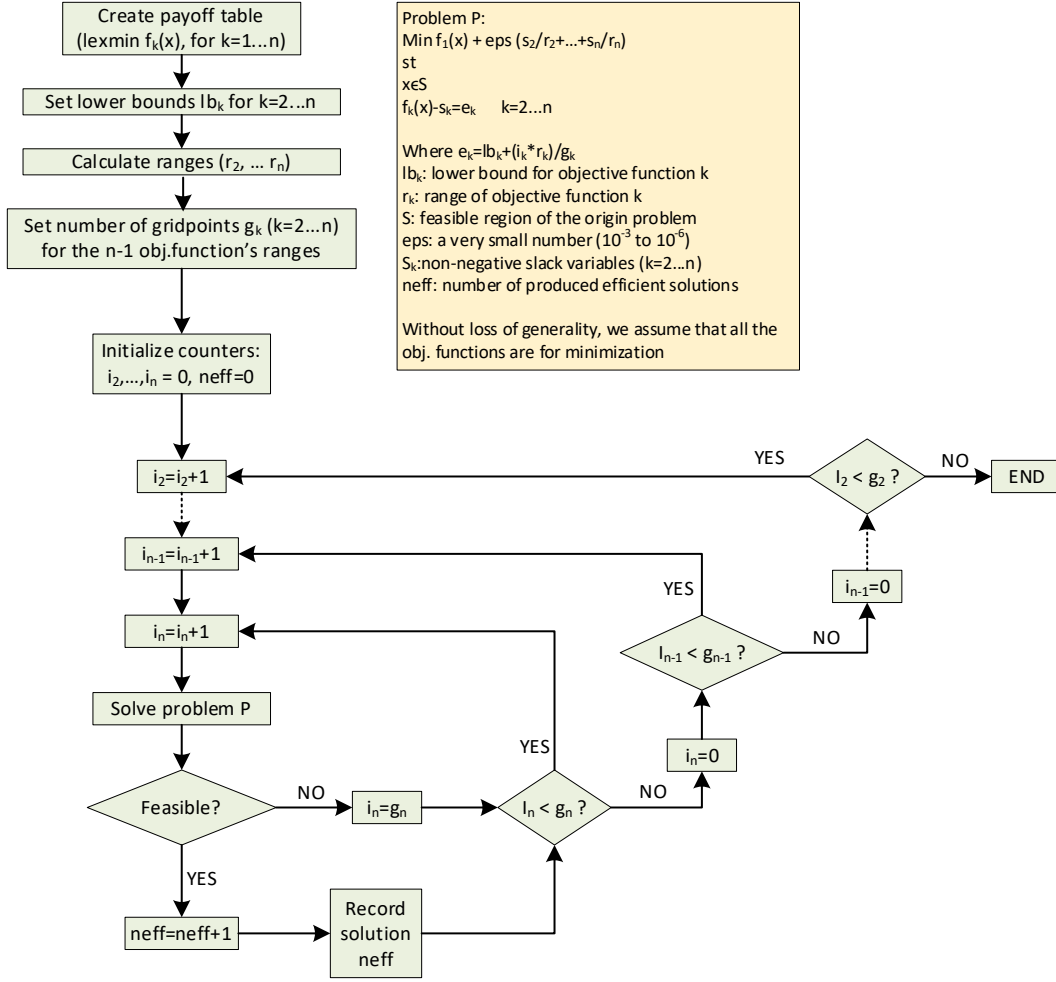


Figure 1 Flowchart of the AUGMECON method [33]

In conclusion, the research of E/T scheduling problem with energy consideration is limited. The current literature on this area with adjustable processing times assume that the cost of reducing processing time of jobs is linear. On the other hand, some researchers use “speed scaling strategy” to solve several scheduling problems but not for E/T performance measures.

CHAPTER 3

ENERGY CONSIDERATION IN THE E/T SCHEDULING

PROBLEM

This chapter discusses scheduling problems with total earliness and tardiness performance combined with energy consumption related performance measures. The effect of speed machining level on job processing time and energy consumption rate is then examined. Several assumptions such as; no preemptions are allowed, no cancellation when the job starts, the machine is always available, and other assumptions mentioned in chapter 1 are applied. At the end of this chapter, several properties for the addressed problem are proposed to find optimum scheduling conditions.

3.1 Energy Consumption on Basic E/T scheduling problem

The Basic E/T scheduling problem with speed scaling in a single machine can be divided into two sub-problems, sequencing the jobs and determining the speed of processing each job. Two solution approaches can be used. If the integrated solution approach is selected, the process of sequencing the jobs and determining the speed level for each job are done simultaneously. Whereas in the hierarchical solution approaches, the sequence of the job is determined first followed by determining the speed level for each job as a second step, or vice versa.

There are two contradicting objectives, first is minimizing the total deviation of completion time of jobs from the common due date and minimizing energy consumption as a second objective. A simple example is used to demonstrate the conflict between the two objectives:

Example 1. Consider four jobs that need to be scheduled on a single machine with processing times as shown in Table 1. Each job can be processed in three different speeds. Let v_i denote the machining speed of level i and π_i denote the power energy consumption at speed level i . Note that v_1 and π_1 are the lowest speed level and its corresponding energy consumption. The common due date d is assumed to be large enough to be unrestrictive. Table 2 shows the power energy consumed in processing a job at different machining speeds. Figure 2 shows the Gantt chart for three different scheduling scenarios.

Table 1 Processing Time of Jobs at the lowest speed

Jobs	J ₁	J ₂	J ₃	J ₄
P _j (hour)	6	8	12	16

Table 2. Three Different Speed Levels and Its Power Consumption

Speed level index (i)	v_i	π_i (kW per hour)
1	1	3
2	1.5	7
3	2	13

The processing time for job j depend on the speed level i in which the job is processed. The power consumed for processing job j at level i is $\left(\frac{P_j}{v_i}\right) \pi_i$. When all jobs are processed at the lowest level (v_1), processing times are 6, 8, 12, 16 and the optimum E/T solution is given in Figure 2(a). When all jobs are processed at the highest speed level, processing

times are 3, 4, 6, 8 and the optimum is given in Figure 2(b). Hence, to achieve the minimum energy consumption, all jobs should be processed at the lowest possible speed, i.e. v_1 (Figure 2(a)). This scenario leads to a high value of total earliness and tardiness of the schedule (32 hours) with minimum energy consumption (126 kWh). On the other hand, if the objective is to minimize the earliness and tardiness jobs, all jobs should be processed using the fastest possible speed, i.e. v_3 (Figure 2(b)). This scenario leads to the minimum value of total E/T (16 hours) with maximum of energy consumption (273 kWh). Noted that all schedules in Figure 2 are V-shaped optimal for a given speed level.

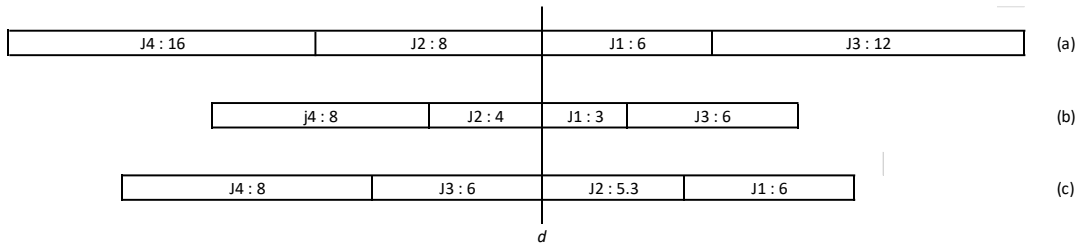


Figure 2 Three Different Scheduling Scenarios

In scenario (c), the speed level of each job is varying i.e., job 1 processes with v_1 , job 2 with v_2 , job 3 and 4 with v_3 (see Figure 2(c)) resulting processing time ; 6, 5.3, 6, 8. Table 3 shows the detail of the performance measure of those three scenarios.

Table 3 Performance Measure of Three Different Scheduling Scenarios

Scenario	Earliness (E)	Tardiness (T)	Total E/T	Total Processing time	Total Energy Consumption
(a)	8	24	32	42	126.0
(b)	4	12	16	21	273.0
(c)	6	16.6	22.6	25.3	237.1

3.2 Energy Consumption Ratio

While a job can be processed at different speed level, we need to investigate the effect of the speed level on the total energy consumption for processing the job. Intuitively, increasing the speed of the machine will decrease the processing time and increase the power consumption simultaneously. Here, changing the speed level has several possibilities in terms of power consumption. Consider the following example.

Example 2. Assume that there is a job j with processing time equal to 5 hours (if processes at speed level 1). There are three different machines with different characteristics as shown in Table 4.

Table 4. Power Consumption Data of 3 Different Machines

Speed level	Power Consumption (KW per hour)		
	Machine 1	Machine 2	Machine 3
$v_1 = 1$	5	5	5
$v_2 = 2$	8	10	12

Using equation 2.3 we can calculate the energy consumption in processing job j at speed levels 1 and 2 in each machine, as shown in Table 5.

Table 5. Energy Consumption in Different Speed Processing

Type	Energy Consumption (kWh)		Comment
	$v_1 = 1$	$v_2 = 2$	
Machine 1	40	32	Increasing the speed level will decrease the energy consumption
Machine 2	40	40	Increasing the speed level has no effect on the amount of energy consumption
Machine 3	40	48	Increasing the speed level will increase the energy consumption

Hence, changing the speed level has 3 possible outcomes. First, increasing the speed level will reduce processing time and energy consumption simultaneously as shown in Machine 1. Second, increasing the speed level will decrease the processing time but the process will consume the same amount of energy as shown in Machine 2. Third, increasing the speed level will decrease the processing time but increasing the energy consumption to finish the job as shown in Machine 3.

Mathematically, in order to determine the effect of changing the speed level on energy consumption, we can calculate the Energy Consumption Ratio (ECR) of speed level i as follows,

$$ECR_i = \frac{(\pi_i - \pi_1)/(v_i - v_1)}{\pi_1}, \text{ for } i > 1 \quad (3.1)$$

The ECR is unit less since the unit of the numerator is cancelled by the unit of denominator. Practically, for scheduling with earliness, tardiness, and energy consumption as performance measures, the value of ECR_i can be interpreted as follows:

- $ECR_i < 1$, means that processing a job with speed level v_i is preferable to v_1 since it will decrease total E/T and energy consumption simultaneously. Hence, processing jobs with a higher speed level, i , will benefit both objectives. Therefore, processing jobs with speed level v_i which has the lowest ECR_i value will provide an optimum solution for energy objective. For special case where the maximum possible speed level has minimum ECR with value less than 1, the optimal solution for both objectives can be found by setting the machine to process all jobs in maximum speed level, and then the algorithm for solving the basic E/T scheduling problem can be applied.
- $ECR_i = 1$, means that processing a job with speed level v_i consumes a similar amount of energy of speed v_1 . Hence, increasing the speed level to v_i benefits only the E/T performance since it will decrease processing time without additional energy consumption. In other words, processing a job with v_i dominates v_1 . Hence, the option of process jobs with v_1 can be eliminated. If all speed levels have $ECR_i = 1$, then this problem can be treated as a basic E/T scheduling problem and optimal scheduling can be achieved by setting the speed at the highest level.
- $ECR_i > 1$, means that increasing the processing speed reduces processing time as well as earliness and tardiness, but requires more energy consumption. Hence, selecting a higher speed level benefits E/T objective but harms the energy objective.

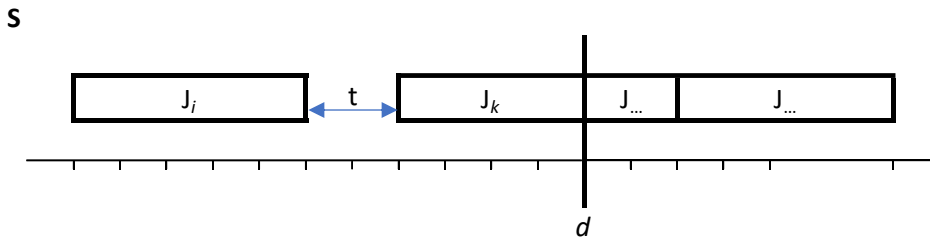
In this case, the optimization procedure is needed to compromise both objectives as shown in Example 1.

3.3 Properties of the optimal schedule

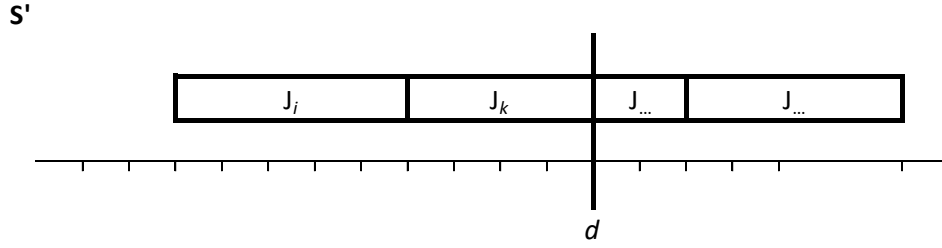
In this section, several properties are identified for the optimal solution for the basic E/T scheduling problem with energy consumption consideration. Six properties of the optimum solution are identified. These properties hold for a given set v_i and processing time P_j .

Property 1: For the basic single machine E/T scheduling with energy consideration problem, schedule with no inserted idle time dominate schedules with inserted idle times.

Proof. This property is in line with Theorem 1 proposed by Baker and Trietsch [4]. Let S be an optimal schedule with inserted idle time between jobs i and k and both jobs are early as shown below:

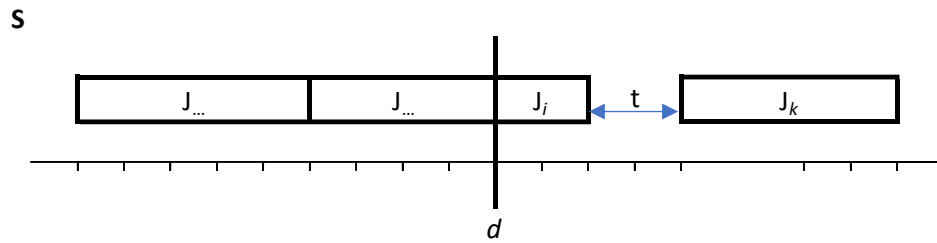


Let S' be a schedule obtained by shifting job i with all its predecessors to the right completing at the start of job k as shown below:

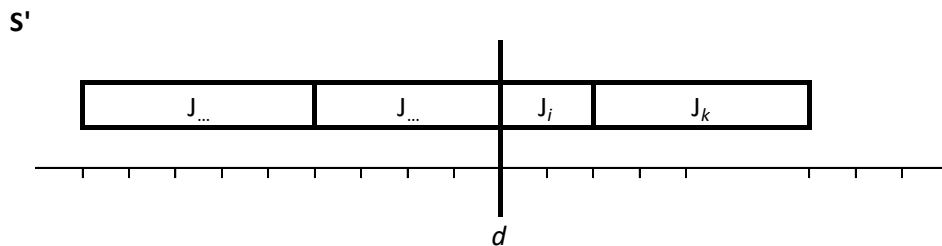


Clearly, the earliness of schedule S' is less than schedule S, $\{ E(S') < E(S) \}$, tardiness of schedule S' are equal with schedule S, $\{ T(S') = T(S) \}$, and energy consumption in schedule S' will decrease by φt , $\{ TEC(S') = TEC(S) - \varphi t \}$, where φ = power consumption during idle time.

Similarly, for the case of idle time in set A where both jobs are tardy, the property is still valid. Let S be an optimal schedule with inserted idle time as shown below:



Let S' be a schedule obtained by shifting job J_k along with all its successors to the left as shown below:



Clearly, the tardiness of schedule S' is less than schedule S , $\{ T(S') < T(S) \}$, earliness of schedule S' and schedule S are equal, $\{ E(S') = E(S) \}$, and energy consumption in schedule S' will decrease by φt , $\{ TEC(S') = TEC(S) - \varphi \Delta t \}$ since the schedule will finish early.

□

Property 1 allows us to eliminate any idle (waste) energy and consider only processing energy while constructing a schedule for basic E/T scheduling problems with energy consideration. Also, only scheduling with contiguous jobs are considered in the solution set. This reduces the size of the set of schedules potentially optimum.

Property 2: In the basic E/T scheduling with energy consideration problem, schedule with jobs in set A are in SPT order and jobs in set B are in LPT order with respect to $\frac{p_j}{v_i}$ dominates other schedules.

Proof. let S be an optimal schedule in which some adjacent pair of jobs in set A are not in SPT order. Since jobs in Set A are jobs that start on or after the due date, then a pairwise interchange of these two jobs will reduce the total tardiness without affecting the total earliness and energy consumption. Similarly, if S is an optimal schedule containing an adjacent pair of jobs in set B that are not in LPT order. Since jobs in set B consist of job that complete on or before the due date, then interchange these two jobs will reduce the total earliness without affecting the total tardiness and total energy consumption. In other words, schedule S cannot be an optimal schedule.

□

Property 2 has no impact on energy consumption objective since interchanging job position do not change the energy consumption. However, this property guides the scheduler to sequence optimally the jobs in terms of E/T performance objective that eventually provides optimal scheduling for this integrated problem.

Property 3: In basic E/T scheduling with energy consideration problems, an optimal schedule contains a job with completion time coinciding with the common due date for any given set v_i .

Proof. The proof of this property in terms of E/T performance measure can be seen at Baker & Trietsch [4]. In terms of energy consumption performance, since shifting a set of jobs early or tardy to make one jobs finish exactly on the due date does not change the amount of energy required to process those jobs. Hence this property still holds under energy consumption objective. $\square$

Property 3 is important since it can reduce the number of potential schedules for optimality.

Property 4: In the basic E/T scheduling with energy consideration problem, an optimal schedule exists in which job J_b completes exactly at the due date where b is the smallest integer equal to or greater than $n/2$.

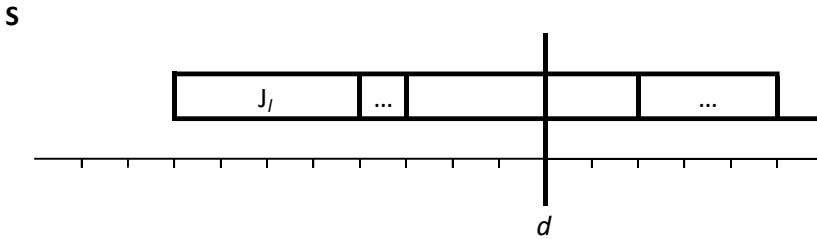
Proof. The proof of this property in terms of E/T performance measure can be seen at Baker & Trietsch [4]. Similar to Property 3, this property is not affected by the speed of

jobs since shifting some job early or tardy will not change the amount of energy required to process those jobs. □

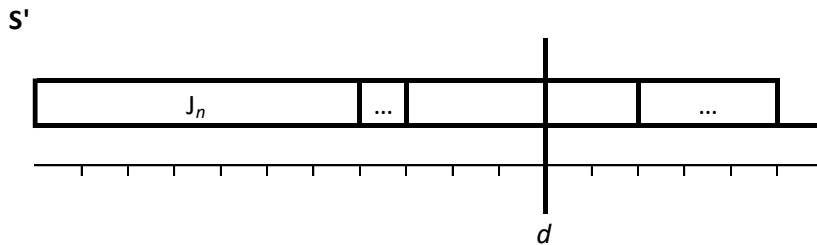
Property 4 results a V-shape schedule and help in determining the job completes exactly at the due date in an optimum schedule.

Property 5: In the basic E/T scheduling with energy consideration problem, the schedule in which the job with the longest processing time is processed first with lowest speed level constitute a dominant set if $ECR > 1$.

Proof. let S be an optimal schedule with the longest job J_l proceed first with maximum speed level at the first position in the sequence as shown below:



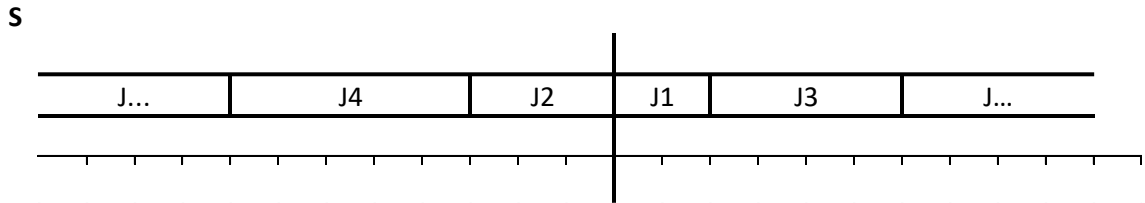
Let S' denotes schedule in which job J_l processed in lowest speed level starting on an earlier time with the same completion time as in S . (we assume that there is enough time before d to do so since d is large).



Clearly, the E/T cost for both schedules, S, S' are equal but total energy consumption in schedule S' is lower than schedule S. It means that increasing processing time at job J1 does not have an impact on E/T cost of the schedule S'. Since schedule S' has lower total energy consumption, schedule S cannot be optimum schedule. $\square$

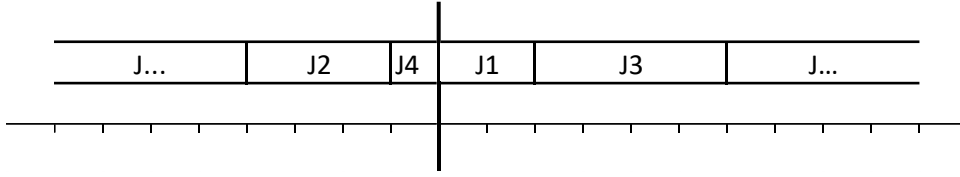
Property 6: In the basic E/T scheduling with energy consideration problem, a schedule in which prioritizing jobs with lower processing time to be speed up constitute a dominant set when the objective function is define using both objectives.

Proof. It is obvious that increasing the speed level for jobs closes to the due date will give more advantages to reduce the earliness and tardiness performance. Contrary, processing jobs that are far from the due date at a low-speed level has less impact on the earliness tardiness performance. Based on Property 2, the jobs that assign near the due date are job that has shorter processing time. Suppose S denotes schedule (optimal in term of E/T performance) in which all jobs are processed at lowest speed level.



Without loss of generality, Let $P_{J_4} = P_{J_1} + P_{J_2}$ and $\frac{P_{J_4}}{v_k} \leq P_{J_1} \leq P_{J_2}$. Suppose a schedule S' in which job J_4 is processed with speed level k arranged optimally in terms of E/T performance. Hence, the total energy consumption increases by $\left(\frac{P_{J_4}}{v_k} * \pi_k - P_{J_4} * \pi_1 \right)$.

S'



The change in E/T of Schedule S' is as follows:

$$E = \frac{P_{J_4}}{v_k} * (a - 1) + P_{J_2} * (a - 2) + \dots$$

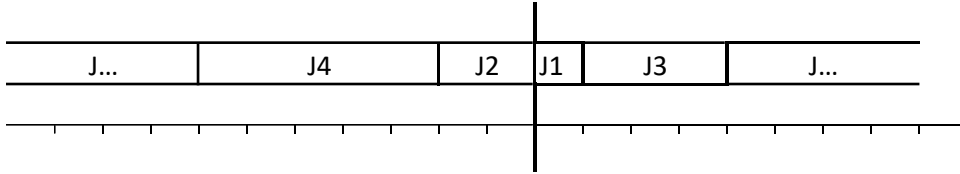
$$T = P_{J_1} * b + P_{J_3} * (b - 1) + \dots$$

Suppose another Schedule S'' in which job J_1 and J_2 is processed with speed level k arranged optimally in terms of E/T performance. Since $P_{J_1} + P_{J_2} = P_{J_4}$, the total energy

consumption increases by $\left(\frac{P_{J_1} + P_{J_2}}{v_k} * \pi_k - P_{J_1} + P_{J_2} * \pi_1 \right) = \left(\frac{P_{J_4}}{v_k} * \pi_k - P_{J_4} * \pi_1 \right)$.

Hence, in terms of energy consumption, Schedule S' and Schedule S'' has equal impact on increasing energy consumption.

S''



The change of E/T of Schedule S'' is as follows:

$$E = \frac{P_{J_2}}{v_k} * (a - 1) + P_{J_4} * (a - 2) + \dots$$

$$T = \frac{P_{J_1}}{v_k} * b + P_{J_3} * (b - 1) + \dots$$

Mathematically, comparing the E/T performance of Schedule S' and Schedule S'' is follows: (E/T for Schedule S' vs: E/T for Schedule S'')

$$\frac{P_{J_4}}{v_k} * (a - 1) + P_{J_2} * (a - 2) + P_{J_1} * b \quad \text{vs:} \quad \frac{P_{J_2}}{v_k} * (a - 1) + P_{J_4} * (a - 2) + \frac{P_{J_1}}{v_k} * b$$

if $a = b$, then:

$$\frac{P_{J_4}}{v_k} * (a - 1) + P_{J_2} * (a - 2) + P_{J_1} * a \quad \text{vs:} \quad \frac{P_{J_2}}{v_k} * (a - 1) + P_{J_4} * (a - 2) + \frac{P_{J_1}}{v_k} * a$$

$$\frac{P_{J_4}}{v_k} * a + P_{J_2} * a + P_{J_1} * a - \frac{P_{J_4}}{v_k} - 2(P_{J_2}) \quad \text{vs:} \quad \frac{P_{J_2}}{v_k} * a + P_{J_4} * a + \frac{P_{J_1}}{v_k} * a - \frac{P_{J_2}}{v_k} - 2(P_{J_4})$$

since $P_{J_4} = P_{J_1} + P_{J_2}$, then

$$\frac{P_{J_4}}{v_k} * a + P_{J_4} * a - \frac{P_{J_4}}{v_k} - 2(P_{J_2}) \quad \text{vs:} \quad \frac{P_{J_4}}{v_k} * a + P_{J_4} * a - \frac{P_{J_2}}{v_k} - 2(P_{J_4})$$

$$-\frac{P_{J_4}}{v_k} - 2(P_{J_2}) > -\frac{P_{J_2}}{v_k} - 2(P_{J_4})$$

if $a = b + 1$, then:

$$\frac{P_{J_4}}{v_k} * (a - 1) + P_{J_2} * (a - 2) + P_{J_1} * (a - 1) \quad \text{vs:} \quad \frac{P_{J_2}}{v_k} * (a - 1) + P_{J_4} * (a - 2) + \frac{P_{J_1}}{v_k} * (a - 1)$$

$$\frac{P_{J_4}}{v_k} * a + P_{J_2} * a + P_{J_1} * a - \frac{P_{J_4}}{v_k} - 2(P_{J_2}) - P_{J_1} \quad \text{vs:} \quad \frac{P_{J_2}}{v_k} * a + P_{J_4} * a + \frac{P_{J_1}}{v_k} * a - \frac{P_{J_2}}{v_k} - 2(P_{J_4}) - \frac{P_{J_1}}{v_k}$$

$$\frac{P_{J_4}}{v_k} * a + P_{J_4} * a - \frac{P_{J_4}}{v_k} - 2(P_{J_2}) - P_{J_1} \quad \text{vs:} \quad \frac{P_{J_4}}{v_k} * a + P_{J_4} * a - \frac{P_{J_4}}{v_k} - 2(P_{J_4})$$

$$-2(P_{J_2}) - P_{J_1} > -2(P_{J_4})$$

The reduction of E/T performance of Schedule S'' is greater than Schedule S' . Hence, Schedule S'' dominates Schedule S' . □

Property 6 result a reverse-V-shape of speed-level configuration and give a guidance to search optimum schedule with optimum speed-level configuration. On the other words, the optimum schedule has speed level configuration in non-increasing order while jobs arrange in SPT order.

CHAPTER 4

MIXED INTEGER LINEAR PROGRAMMING MODEL

In this chapter, a mixed integer linear programming model is proposed for the scheduling problem with total earliness - tardiness and energy consumption performance measure. Two methods are used to solve the addressed problem. Several assumptions are applied such as no preemption is allowed while processing a job, no cancellation when the job starts, the machine is always available to process a job, processing time is known in advance. All jobs are ready at time zero. The machine has a feature to be operated at several discrete speed levels. A job can be processed in only one of the possible speed levels. Once a schedule and speed levels are set, it can't be changed while processing. Also, the time to change the speed level of the machine is relatively small compare to the processing time hence are neglected the following formulation is based on the E/T formulation introduced by Low [3].

4.1 Formulation

Sets

$J = \text{Set of } n \text{ jobs,} \quad j = 1, \dots, n$

$V = \text{Set of } s \text{ possible machining speed level,} \quad v_i, i = 1, \dots, s$

Parameters

P_j = Processing time of job j , $j=1, \dots, n$

v_i = machining speed at level i , $i=1, \dots, s$

π_i = Power consumption of machine at speed level i $i=1, \dots, s$

d = common due date (Unrestrictive large)

M = An appropriately large positive number

Decision variables

S_j = Starting time of job j , $j=1, \dots, n$

C_j = Completion time of job j , $j=1, \dots, n$

E_j = Earliness of job j , $j=1, \dots, n$

T_j = Tardiness of job j , $j=1, \dots, n$

TET = Total Earliness and Tardiness

TEC = Total Energy Consumption

X_{ji} = Binary variable, 1 if job j is processed with speed v_i , 0 otherwise.

$j=1, \dots, n$; $i=1, \dots, s$

Y_{jk} = Binary variable, 1 if job j processes before job k , 0 otherwise.

$j, k=1, \dots, n$; $j \neq k$

The MILP models (Multi Objectives Mixed Integer Linear programming)

Objectives:

$$\text{MIN } \sum_j (E_j + T_j) \quad (4.1)$$

$$\text{Min. } \sum_j \sum_i \left(\frac{P_j * \pi_i}{v_i} * X_{ji} \right) \quad (4.2)$$

subject to :

$$\sum_{i=1}^s X_{ji} = 1 \quad \text{for } j \in J \quad (4.3)$$

$$C_j = S_j + \sum_{i=1}^s \frac{P_j}{v_i} * X_{ji} \quad \text{for } j \in J \quad (4.4)$$

$$C_j + E_j - T_j = d \quad \text{for } j \in J \quad (4.5)$$

$$S_k \geq C_j - M(1 - Y_{jk}) \quad \text{for } j, k \in J ; j \neq k \quad (4.6)$$

$$Y_{jk} + Y_{kj} = 1 \quad \text{for } j, k \in J ; j < k \quad (4.7)$$

$$X_{ji} \in \{0,1\}, \quad Y_{jk} \in \{0,1\} \quad \text{for } j, k \in J ; i \in V \quad (4.8)$$

$$S_j \geq 0, \quad C_j \geq 0, \quad E_j \geq 0, \quad T_j \geq 0 \quad \text{for } j \in J \quad (4.9)$$

Objective function (4.1) defines the primary objective of minimizing the total earliness and tardiness of jobs. Objective function (4.2) defines the secondary objective of minimizing energy consumption. Constraint (4.3) ensures that any job has to be processed exactly with one speed level. Constraint (4.4) calculates the completion time of job j . Constraint (4.5) calculates the earliness or tardiness for job j . Constraint (4.6) is to ensure that a job may not start before the completion of all previous scheduled jobs. Constraint (4.7) defines the precedence relationship between pairs of jobs.

Example 3: Consider a simple case of scheduling two jobs, j_1 and j_2 with two possible speed levels, v_1 and v_2 . The MILP model for this problem is as follows,

Objectives :

$$\text{Min. } (E_1 + T_1) + (E_2 + T_2)$$

$$\text{Min. } \left(\frac{P_1 * \pi_1}{v_1} * X_{11} \right) + \left(\frac{P_1 * \pi_2}{v_2} * X_{12} \right) + \left(\frac{P_2 * \pi_1}{v_1} * X_{21} \right) + \left(\frac{P_2 * \pi_2}{v_2} * X_{22} \right)$$

subject to :

$$X_{11} + X_{12} = 1$$

$$X_{21} + X_{22} = 1$$

$$C_1 = S_1 + \frac{P_1}{v_1} * X_{11} + \frac{P_1}{v_2} * X_{12}$$

$$C_2 = S_2 + \frac{P_2}{v_1} * X_{21} + \frac{P_2}{v_2} * X_{22}$$

$$C_1 + E_1 - T_1 = d$$

$$C_2 + E_2 - T_2 = d$$

$$S_1 \geq C_2 - M(1 - Y_{21})$$

$$S_2 \geq C_1 - M(1 - Y_{12})$$

$$Y_{12} + Y_{21} = 1$$

$$X_{11}, X_{12}, X_{21}, X_{22}, Y_{11}, Y_{12}, Y_{21}, Y_{22} \in \{0,1\}$$

$$S_1, S_2, C_1, C_2, E_1, E_2, T_1, T_2 \geq 0$$

4.2 Integrated objective functions

Different from the single objective mathematical model, most of MOMP problem do not have a single solution that simultaneously optimizes all objective functions. Hence, the optimality concept in MOMP problems is replaced by “Pareto efficient” or “Pareto optimal” solution set, which consists of a set of non-dominated solutions. Providing a Pareto set gives the decision-maker an option to select “the most preferable” optimum solution.

4.2.1 Weighted Approach

The integration of objective functions in E/T scheduling problem with energy consideration may be done by a “weighted approach”. In this study, the Primary objective function is the Total Earliness-Tardiness (TET), and the secondary objective is the Total Energy Consumption (TEC). Let w be the subjective weight of the primary objective and $(1 - w)$ be the weight of the secondary objective where the value of w is between zero and one ($0 \leq w \leq 1$). Since the units of both objectives are different, we need to normalize both values in the integrated objective function as shown in equation 4.10 below [24]:

$$z = w \frac{TET - TETU}{TETN - TETU} + (1 - w) \frac{TEC - TECU}{TECN - TECU} \quad (4.10)$$

where TETU and TECU are the “utopian point” of TET and TEC, consecutively. TETN and TECN are the “nadir point” of TET and TEC, consecutively. Utopian point refers to the best possible value of the objective function and Nadir point refers to the worst non-dominated value [24]. Hence, objective functions 4.1 and 4.2 can be replaced by equation 4.10.

In order to generate Pareto solutions, the value of w is generated uniformly with increments of 0.1 starting from 0 to 1 ($w \in \{0, 0.1, \dots, 1\}$) resulting in at most 11 points of the non-dominated solutions. The procedure can be described as follows. First, Lexicographic optimization [34] is utilized to find the Utopian and the Nadir point for both objective functions. Then, the problem is solved for each value of w . Constraint 4.11, 4.12, and 4.13 are added to the MILP model to ensure Pareto efficient result for w equals 0 or 1.

$$TEC \leq TECN \quad (4.11)$$

$$TET \leq TETN \quad (4.12)$$

$$TEC \geq 0, TET \geq 0 \quad (4.13)$$

A simple example is presented to demonstrate the implementation of this procedure.

Example 4: Consider 10 jobs to be scheduled on a single machine. All jobs have a common due date, $d = 80$ hours. Table 6 and 7 presents the example's data.

Table 6. Job Processing Time

Jobs	J1	J2	J3	J4	J5	J6	J7	J8	J9	J10
P _j (hr)	4	5	7	8	10	11	12	15	18	20

Table 7. Speed level and power consumption

Speed level	1	2	3
V _i	1	1.5	2
Π _i (kW)	3	6	13

The MILP weighted approach for this problem is coded in Python 3.0 using PULP library and run on an Intel Core i5 2.4GHz Memory 8GB. Figure 3 shows the resulting Pareto solutions. See Appendix 1 for the details of the code.

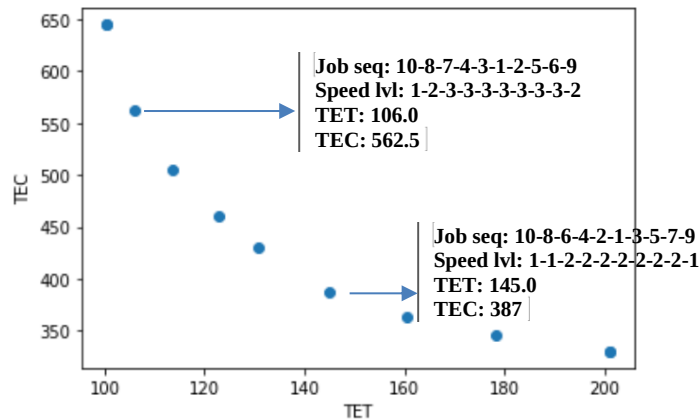


Figure 3 Pareto optimum of Example 4 Using Weighted Method

Note that the first pareto point (at the top left of Figure 3) is the result of the optimization for two values of w ; 1 and 0.9 having the values of TET = 105.5 and TEC = 645. Hence, those two values of w produced the same pareto points. The next point is for $w = 0.8$ with TET=106 and TEC= 562.5. For this point, the resulting job sequence is 10-8-7-4-3-1-2-5-6-9 with speed levels 1-2-3-3-3-3-3-3-2, consecutively. The detailed information related to all values of w is given in Table 8.

Table 8. Resulting Pareto solutions for example 4 using the weighted method

Point	w	TET	TEC	Integrated objective	Run time (s)	Sequence	Speed
1	1	100.5	645	0	253.73	10-9-6-4-2-1-3-5-7-8	1-3-3-3-3-3-3-3-3
2	0.9	100.5	645	0.1	119.79	10-8-6-5-3-1-2-4-7-9	1-3-3-3-3-3-3-3-3
3	0.8	106	562.5	0.191	107.29	10-8-7-4-3-1-2-5-6-9	1-2-3-3-3-3-3-3-2
4	0.7	113.7	505	0.258	116.61	10-9-7-4-3-1-2-5-6-8	1-2-2-3-3-3-3-3-2-2
5	0.6	122.7	460	0.297	136.76	10-9-6-4-2-1-3-5-7-8	1-2-2-2-3-3-3-2-2-2
6	0.5	130.7	430	0.309	167.21	10-8-7-4-3-1-2-5-6-9	1-2-2-2-2-3-2-2-2-2
7	0.4	145	378	0.286	152.39	10-8-6-4-2-1-3-5-7-9	1-1-2-2-2-2-2-2-2-1
8	0.3	160.3	364	0.254	100.45	10-8-6-4-2-1-3-5-7-9	1-1-1-2-2-2-2-2-1-1
9	0.2	178.3	346	0.196	113.42	10-9-7-4-3-1-2-5-6-8	1-1-1-1-2-2-2-1-1-1
10	0.1	201	330	0.099	112.66	10-9-7-5-2-1-3-4-6-8	1-1-1-1-1-1-1-1-1-1
11	0	201	330	0	6.03	10-9-6-4-2-1-3-5-7-8	1-1-1-1-1-1-1-1-1-1

In Table 8, $w=1$ means that the optimization process only considers TET objective performance because the weight of TET is one and the weight of TEC is zero. So, the value

of $TET=100.5$ is the TETU and $TEC=645$ is the TECN. Conversely, $w=0$ means the optimization process only considers the TEC objective hence $TEC=330$ is the TECU and $TET=201$ is the TETN. Note that in column “Sequence”, job J_{10} (job with the longest processing time) is always located in the first position with the lowest speed level as shown under column “speed”. This condition is in line with Property 5 that is derived in Chapter 3.

The mechanism of the Weighed Method to generate the Pareto efficient solutions can be seen in Figure 4. The mechanism starts by creating the payoff table. This table consists of the Utopian and Nadir values of the TET and TEC objective functions. Then it follows several steps until the algorithm successfully generate all Pareto efficient solutions and stops.

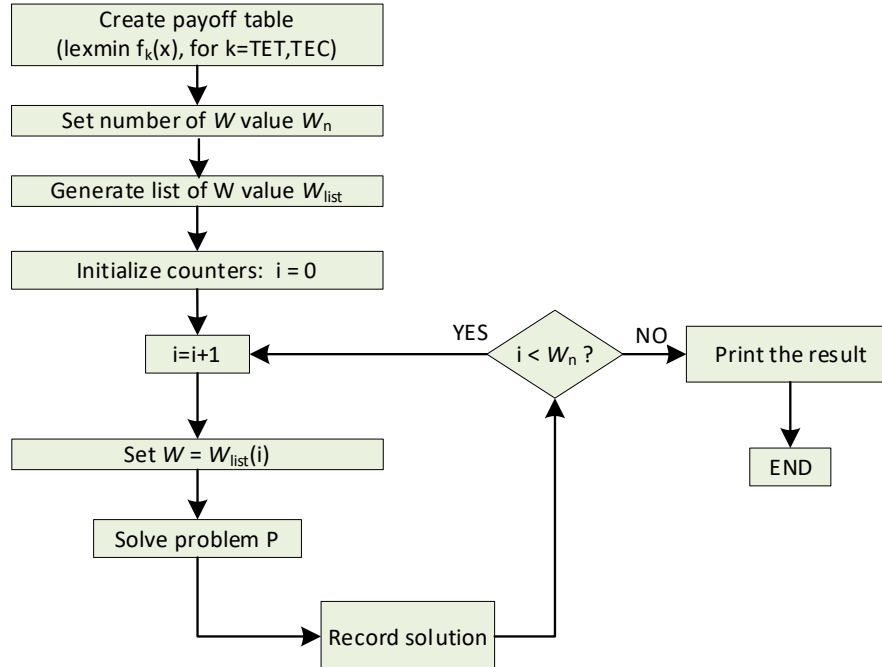


Figure 4 The Algorithm of the Weighed Method

4.2.2 AUGMECON Approach

An alternative approach for dealing with multi-objective functions is applying the Augmented ϵ -constraint (AUGMECON) method. Here, only one objective is selected to define the minimization function while other objectives are treated as constraints. In order to implemented AUGMECON approach to solve the E/T scheduling problem with energy consideration, the total Earliness and Tardiness was selected to be the objective function of the model and the Total Energy Consumption as the constraint.

Initially, Lexicographic optimization is utilized to find the lower bound (TECN) and to calculate the range value of the TEC objective function [34]. Afterward, the number of grids of TEC is set to generate a list of RHS_{TEC} . Equation 4.14 is used as an objective function where s is a slack variable and r_{TEC} is the range of the TEC objective function. Constraint 4.15 and 4.16 are added to the model to represent the ϵ -constraint for TEC objective function where the epsilon value $e \in RHS_{TEC}$. Then, the model is solved iteratively. The iteration procedure will be stopped if there is no optimum solution in the current iteration or the iteration reaches the end of the grid point.

$$Min.: \sum_j (E_j + T_j) - eps * (s/r_{TEC}) \quad (4.14)$$

$$\sum_j \sum_i \frac{P_j * \pi_i}{v_i} * X_{ji} + s = e \quad (4.15)$$

$$s \geq 0 \quad (4.16)$$

The result of the implementation of AUGMECON for example 4 is presented in figure 5. The value of $M = 2000$, $eps = 10^{-3}$, number of grid points (g_{TEC}) = 11 are used to solve the problem. The algorithm was coded in Python 3.0 using PULP library and run on an Intel

Core i5 2.4GHz Memory 8GB. See Appendix 2 for the detailed code. The value of g_{TEC} can be adjusted based on the need of the decision maker.

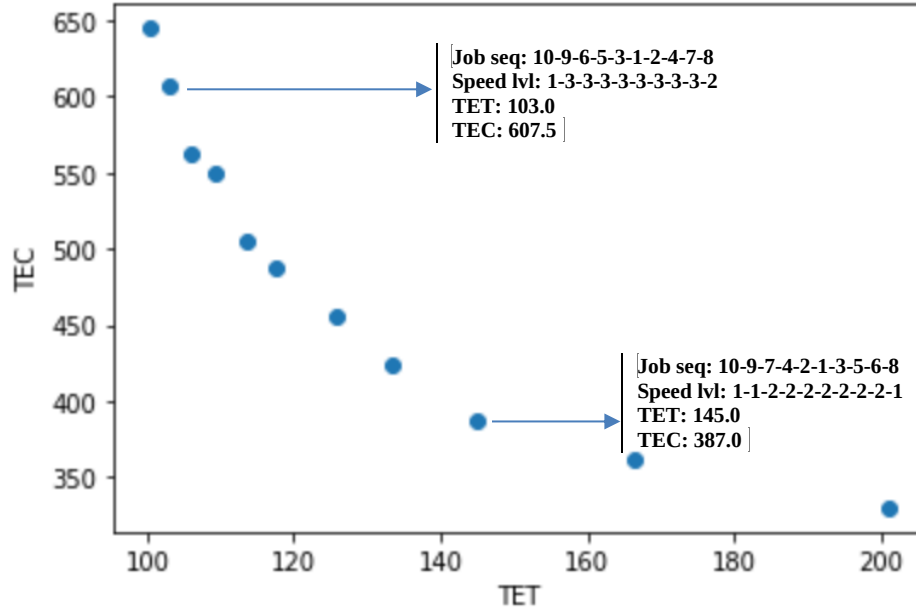


Figure 5 Pareto optimum of Example 4 using AUGMECON Method

The first effecient point (in the top left) on Figure 5 is the result of the obtained for $e = 645$ which is the TETN. The next point is for $e = 613.5$ with TET equal to 103 and TEC equal to 607.5. For this point, the job sequence obtained is 10-9-6-5-3-1-2-4-7-8 with speed levels 1-3-3-3-3-3-3-3-2, consecutively. The detailed results obtained are given in Table 9.

Table 9 Result of Example 4 using AUGMECON Method

Point	e	TET	TEC	Run time (s)	sequence	speed
1	645	100.5	645	150.13	10-8-7-5-2-1-3-4-6-9	1-3-3-3-3-3-3-3-3-3
2	613.5	103	607.5	231.76	10-9-6-5-3-1-2-4-7-8	1-3-3-3-3-3-3-3-2
3	582	106	562.5	360.45	10-9-7-5-2-1-3-4-6-8	1-2-3-3-3-3-3-3-2
4	550.5	109.3	550	250.67	10-8-6-4-3-1-2-5-7-9	1-2-3-3-3-3-2-3-3-2
5	519	113.7	505	471.91	10-8-7-5-3-1-2-4-6-9	1-2-2-3-3-3-3-3-2-2
6	487.5	117.7	487.5	184.34	10-8-7-3-4-1-2-5-6-9	1-2-2-2-3-3-3-3-2-2
7	456	125.7	455	345.8	10-8-7-5-2-1-3-4-6-9	1-2-2-3-2-3-2-2-2-2
8	424.5	133.3	424.5	1142.87	10-8-6-5-2-1-3-4-7-9	1-2-2-2-3-3-2-2-2-1
9	393	145	387	748.26	10-9-7-4-2-1-3-5-6-8	1-1-2-2-2-2-2-2-2-1
10	361.5	166.3	361	814.82	10-8-7-6-3-2-1-4-5-9	1-1-1-2-2-2-1-2-1-1
11	330	201	330	110	10-9-6-5-2-1-3-4-7-8	1-1-1-1-1-1-1-1-1-1

In Table 9, Point 1 with $e = 645$ (which is the value of the TECN), the optimization process of TET can find the $TETU = 100.5$, i.e., the best value of TET. In the next point, the value of e is stricter, hence the TET value increases and the TEC value decreases. The similar manner applied for the following point until the last point with $e = 330$ (which is the value of TECU), the optimization result is $TET = 201$, which is the TETN. Under column “Sequence”, job 10 (job with longest processing time) is always placed in the first position with the lowest speed level as shown in column “speed”. This condition is in line with Property 5 that is derived in chapter 3.

The mechanism of the AUGMECON to generate Pareto Solutions is given in Figure 6. The mechanism starts by creating the payoff table that consists of the Utopian and Nadir values of the TET and TEC objective functions. Then it follows several steps until the algorithm successfully generates the Pareto efficient solution and stops.

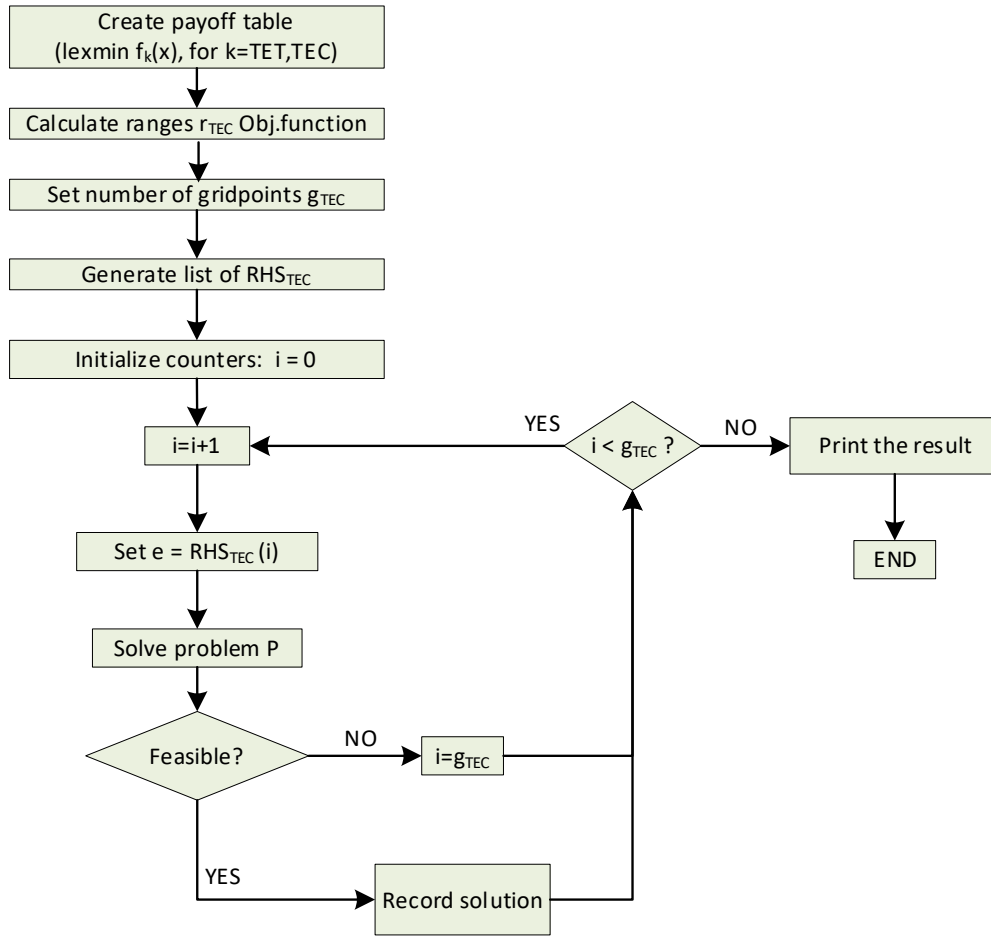


Figure 6 Mechanism of the AUGMECON Method

CHAPTER 5

THE ASCENDING SPEED LEVEL (ASL) HEURISTIC

In this chapter, a heuristic is proposed to solve the basic single machine E / T scheduling problems with energy consideration. An iterative procedure is applied to find the optimal scheduling while a weighted method is used to integrate both objectives. Initially, the speed level of all jobs is set at the lowest speed. Then, in the next iteration, the speed level of a selected job is increased to the next speed level. Based on the properties defined in section 3.3, in each iteration, a job having the lowest speed level and the shortest processing time is selected for increasing its speed level. Also, jobs from the LPT list are assigned to sets A and B (alternating between A and B starting with set B) to construct the corresponding schedule. Note that all assumptions stated in Chapter 1 apply to the problem including unrestrictive common due date.

5.1 Notations

Several terms will be used and defined as follows:

TEC = The Total Energy Consumption.

TECN = The worst possible value of TEC (Nadir point of TEC).

TECU = The best possible value of TEC (Utopian point of TEC).

TET = The Total Earliness and Tardiness.

TETN = The Highest possible value of TET (Nadir point of TET).

$TETU$ = The Lowest possible value of TET (Utopian point of TET).

Set B = Set of jobs completed before or on the due date.

Set A = Set of jobs completed after the due date.

Z value= the objective value

Best_Z = Best value of Z

Best_TEC = Best value of TEC

Best_TCT = Best value of TCT

Best_JOBSEC = Best job sequence and speed level

L = Index of jobs in SPT order

K = Current level of speed

w = Weighted value of TET objective function

SL_list = speed level configuration for each job, where jobs are sorted in SPT order

5.2 Procedure of ASL Method

There are 14 steps regarding the implementation of the proposed algorithm:

1. Calculate $TETU$ and $TECN$ by algorithm 1.
2. Calculate $TECU$ and $TETN$ by algorithm 2.
3. Generate list of w value, w_{list}
4. Start the iteration for specific w value in w_{list}
5. Specify the parameters and the variables
6. Generate the speed levels for each job.
7. Calculate processing time and order the list in LPT order

8. Assign jobs from LPT list alternating between A and B sets starting with set B.
Then, sort set B in LPT order and set A in SPT order.
9. Calculate TEC, TET and z value.
10. Check if the current z value greater than Best_Z, go to step 11. Otherwise, set Best_Z = z value, Best_TEC = TEC, Best_TET = TEC, and record the current sequence as the Best sequence, then go to step 11.
11. Update L and K. Then check if all speed combinations are generated or not. If not, go to step 4. Otherwise go to step 12.
12. Report Best value of Z, TET, TEC and best job sequence for specific value of w.
13. Repeat the iteration for other w value in w_{list} , if any.
14. Generate the Pareto optimum solutions.

Figure 7. below shows the flowchart of the ASL algorithm. It gives a more comprehensive view of how this algorithm works to generate Pareto optimum solutions.

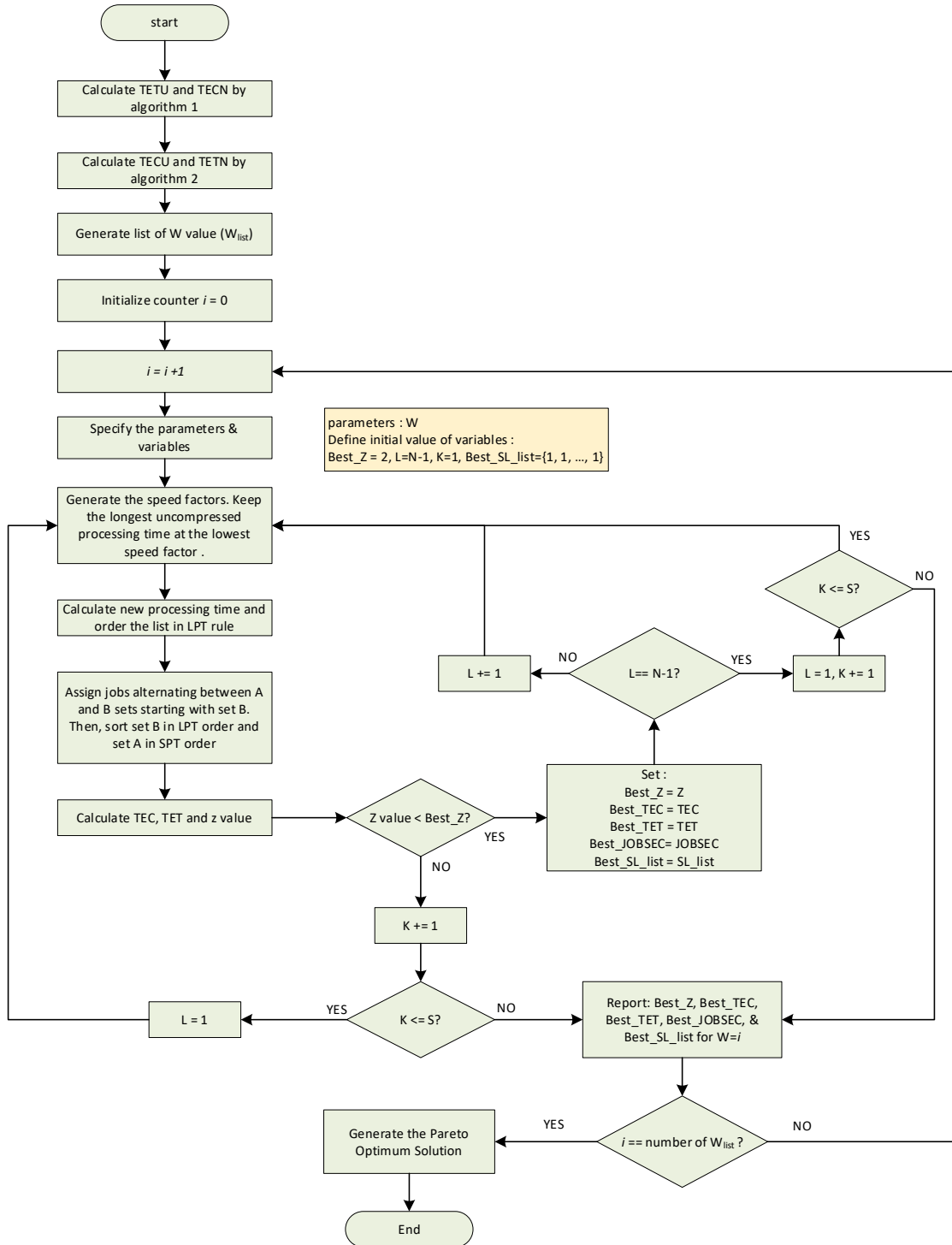


Figure 7 Flowchart of ASL Method

The steps of the procedure are explained in detail as follows:

Step#1:

The algorithm is initiated by calculating the payoff table. It consists of the best possible value (utopian) and the worst value (nadir) values of both objectives used to calculate the integration value of both objectives. Because the units of the two objectives are different, *Normal Boundary Intersection* (NBI) [11][23] was then utilized to calculate the value of integrated objectives. Since both objectives are contradictory, calculating the utopian value for one objective gives the other objective the nadir value. Algorithm 1 below is used to calculate $TETU$ and $TECN$.

Algorithm 1:

1. Set the speed level for all jobs in the lowest speed level (v_1)
2. Construct the LPT list based on P_{j1} then assign jobs by alternating between A and B sets starting with set B.
3. Order jobs in set B in LPT order and jobs in set A in SPT order based on P_{j1} .
4. Calculate the TET and the TEC values
5. Set $TECN = TEC$ and $TETU = TET$

Step #2:

In this step we calculate TECU and TETN. All jobs are processed at the highest speed level except the longest one. This satisfies Property 5 in chapter 3. Algorithm 2 used to evaluate both values TECU and TETN.

Algorithm 2:

1. Set speed level for all jobs in the highest possible speed level (v_s), except the job with the longest processing time.
2. Calculate the actual processing time of each job (P_{js}) where $P_{js} = \frac{P_j}{v_s}$ for all $j = 1, \dots, n-1$
3. List jobs in LPT order based on P_{js} , then Assign jobs by alternating between A and B sets starting with set B.
4. Order jobs in set B in LPT order and jobs in set A in SPT order based on (P_{js})
5. Calculate the values of TET and TEC
6. Set $TECU = TEC$ and $TETN = TET$

Step #3:

To generating a Pareto solution, the solution algorithm should be iterated using different w values. This parameter is used to define the weighted TET objective function in the integrated objective where $0 \leq w \leq 1$. Hence, $(1-w)$ is the weight of the TEC objectives. Note that $w=1$ means that only the TEC objective is considered and $w=0$ means that only the TET objective is considered. Therefore, in this step, we generate the list of w values (w_{list}). To do that, the range of w values from zero to one is split uniformly into k -intervals resulting in $k+1$ different w values.

Step #4:

In this step, the looping procedure is initiated by selecting a w value from the w_{list} . The selected w value is then used as the parameter for the next step.

Step #5:

In this step, the initial values for several variables are set. The initial value for Best_Z is set to be any number greater than one since the maximum possible $Z = 1$. The initial value of L is set at $L = N - 1$ (since the job with the longest processing time is assigned optimally on the first position in the sequence using Property 5). The initial value of K is set at the lowest speed level, i.e., $K = 1$. The SL_list is set to be $\{1, 1, \dots, 1\}$.

Step #6:

In this step, the speed level for each job is generated then used to calculate the actual processing time as follows. Based on Property 5, assign the job with the longest uncompressed processing time to the first position on the lowest processing speed level. Here, we fix the decision for the first job and its processing speed in the optimal sequence. The next step, we focus on finding the best combination for the remaining jobs.

Optimum scheduling in the Basic E/T Scheduling has a V-shape structure. Hence, increasing the speed level for jobs that are close to the due date will give advantage of reducing the earliness and tardiness performance. Contrary, processing jobs that are far from the due date at a low-speed level has less impact on the earliness and tardiness performance. Based on Property 6, we propose a procedure to generate speed level for each iteration as follow:

The idea is to increase the speed level of a job that has the lowest speed level and shortest processing time by one level in each iteration. In order to do that, update SL_List by increasing the speed level of a job J_L to be K. Later, on step 9, the algorithm will update the

value of L and K to generate the next speed level combination (updating the SL_list) for the next iterations.

This method will search for the best schedule with best speed level configuration in a gradient descent manner and will end up in the best value of Z as shown in Figure 8. The number of iterations needed to find the best value is at most $(N - 1)s$ and the minimum is 2 iterations. Hence, this method can be classified as a polynomial solution.

Step #7:

After a new configuration of the processing speed level for every job is generated in step 6, the actual processing time of each job is calculated using equation 5.1 below. Afterward, jobs are put in LPT order based on p_{ji} .

$$p_{ji} = \frac{p_j}{v_i} \quad (5.1)$$

Step # 8:

In this step, we split jobs into two sets, set B and set A. Set B consists of jobs that complete before or at the due date and set A consist of jobs that complete after the due date. Assign jobs from LPT list (from step 7) by alternating between sets A and B starting with set B. Then, list jobs in set B in LPT order and jobs in set A in SPT order based on p_{ji} . The objective of this step is to construct a V-shape schedule that can optimize E/T performance for the assigned speeds.

Step # 9:

Here, the performance measure of the V-shape schedule obtained in step 8 is calculated i.e., total earliness and tardiness, total energy consumption and the Z value.

Step # 10:

In this step, the current Z value from step 10 is compared with best_Z. If $Z < \text{Best_Z}$ then update value of $\text{Best_Z} = Z$, $\text{Best_TEC} = \text{TEC}$, $\text{Best_TCT} = \text{TCT}$, $\text{Best_JOBSEC} = \text{Current}$ sequence and its speed level combination. Then go to step 11. Otherwise, if $Z \geq \text{Best_Z}$, go to step 11 directly.

Step # 11:

In this step, update the value of L and K based on the Z value from the previous step:

- If at step 10, $Z \leq \text{Best_Z}$, then check if $L = N-1$. If **YES**, update $L = L+1$ then repeat step 4. If **NO**, set $L = 1$ and increase K by one ($K = K+1$). Afterward check if $K \leq S$. If **YES** then go to step 6, otherwise go to step 12.
- If at step 10, $Z > \text{Best_Z}$, then update $K = K+1$. Afterward check if $K \leq s$. If **YES** then set $L = 1$ and go to step 6. If **NO** then go to step 12.

Step #12:

In this step, we stop the current iteration and report the value of Best_Z, Best_TET, Best_TEC, and Best_JOBSET. This result corresponds to the specific value of w that is already defined at the beginning of the looping procedure.

Step #13:

In this step, we check if all values of w in w_{list} are already considered. If not, then go to step 4. Otherwise, go to step 14 to generate the Pareto optimum chart.

Step #14:

In this step, the results of all w values are combined to generate the Pareto optimum solution as the result of the optimization process of the E/T scheduling process with energy consumption consideration. Sometimes, two (or more) values of w will give the same optimum point if the differences of both w values are not significant. Hence, the number of solutions in the Pareto chart may be less than the number of the w values in the w_{list} .

Example 5: Consider 7 jobs that should be scheduled on a single machine. All jobs have a common due date that is large enough to be unconstrained. Tables 10 and 11 presents the data of example 5.

Table 10 Job Processing Time of Example 5

Jobs	J1	J2	J3	J4	J5	J6	J7
P _j (hr)	4	5	7	8	10	11	12

Table 11. Speed Level and Its Power Consumption of Example 5

Speed level	1	2	3	4	5
V_i	1	1.5	2	2.5	3
Π_i (kW)	3	6	10	15	21

Suppose that $w = 0.6$, then step 6 to 11 will generate several solutions iteratively as shown in Figure 8 below.

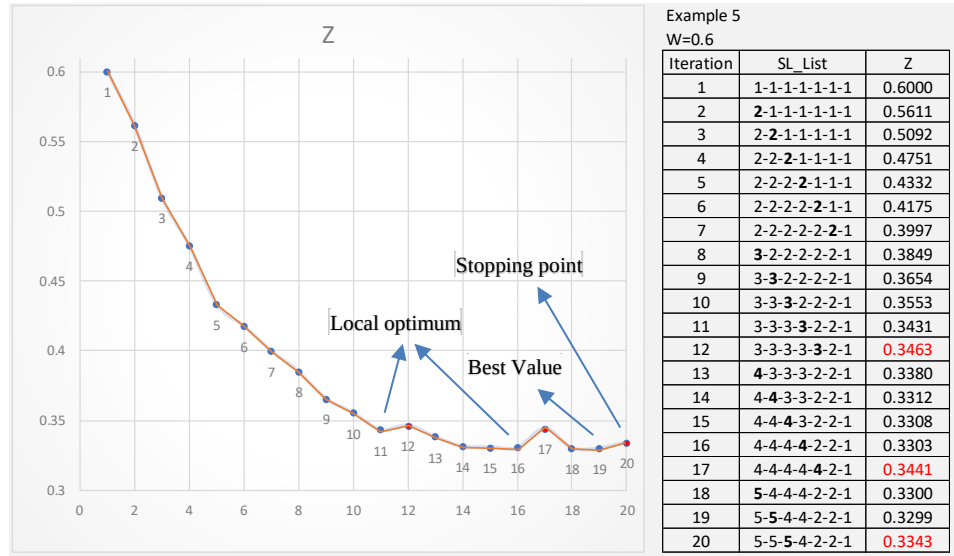


Figure 8 The Detail Iterations of Example 5 with W=0.6

Figure 8 shows the detailed iterations of the ASL method in finding the best Z value and the appropriate speed level configuration for example 5 with $w=0.6$. As mentioned in step 5, the speed level configuration in the initial search procedure for all jobs is 1. In each iteration, SL_list is updated (see numbers in bold) and the Z value is calculated. The 12th and 17th Iterations produce worse Z value than the previous iteration, so the speed level configuration for those iterations are ignored and the algorithm continues to increase the speed level of the first job. The 20th iteration also produces a worse Z value compared to the previous iteration. But there is no option to increase the speed level, hence this point becomes the stopping condition and reports the result of the 19th iteration as the best possible Z value for $w=0.6$.

The final result of the Example 5 that was solved by the ASL can be seen in Figure 9 below.

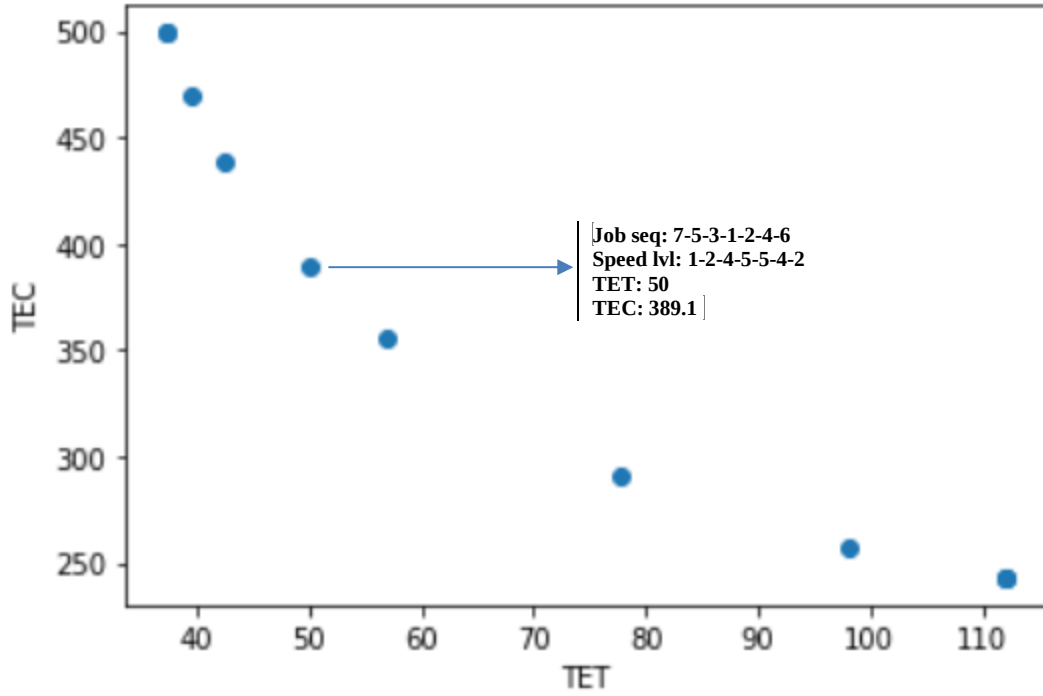


Figure 9 Pareto optimum of Example 5 Using ASL Method

The first Pareto point (in the top left) on Figure 9 is the result of the optimization for $w=1$ and $w=0.9$ with $TET=37.4$ and $TEC=499.1$. In this example, those two values of w produced the same Pareto point. The 4th point from the left is for $w = 0.6$ with $TET=50$ and $TEC=389.1$. For this point, the optimum sequence is $J_7- J_5- J_3- J_1- J_2- J_4- J_6$ and the optimum speed level configuration is 1-2-4-5-5-4-2, consecutively. The detailed information related to each point can be seen in Table 12.

Table 12 Result of Example 5 using ASL Method

Point	W	TET	TEC	Integrated objective	Run time (s)	sequence	speed
1	1	37.4	499.1	0.001	<1	7-5-3-1-2-4-6	1-5-5-5-5-5-5
2	0.9	37.4	499.1	0.1	<1	7-5-3-1-2-4-6	1-5-5-5-5-5-5
3	0.8	39.4	469.1	0.198	<1	7-5-3-1-2-4-6	1-4-5-5-5-5-4
4	0.7	42.4	439.1	0.277	<1	7-5-3-1-2-4-6	1-3-5-5-5-5-3
5	0.6	50	389.1	0.33	<1	7-5-3-1-2-4-6	1-2-4-5-5-4-2
6	0.5	56.8	355	0.349	<1	7-5-3-1-2-4-6	1-2-3-4-4-3-2
7	0.4	77.7	291	0.328	<1	7-5-3-1-2-4-6	1-1-2-3-3-2-1
8	0.3	98	257	0.282	<1	7-5-3-1-2-4-6	1-1-1-2-2-1-1
9	0.2	112	243	0.2	<1	7-5-3-1-2-4-6	1-1-1-1-1-1-1
10	0.1	112	243	0.1	<1	7-5-3-1-2-4-6	1-1-1-1-1-1-1
11	0	112	243	0.001	<1	7-5-3-1-2-4-6	1-1-1-1-1-1-1

In Table 12, $w=1$, means that the optimization process only considers TET objective performance because the weighted of TET is one and the weight of TEC is zero. So, the value of TET =37.4 is the TETU and TEC = 499.1 is the TECN. Conversely, while $w=0$ the optimization process only considers TEC objective hence TECU=TEC=243 and TETN=TET=112. Under column “Sequence”, job 7 (job with the longest processing time) is always processed in the first position with the lowest speed level as shown under column “speed”. This condition is in line with Property 5 that is developed in Chapter 3.

CHAPTER 6

RESULTS AND CONCLUSION

Experiments were conducted to examine the performance of the proposed methods. More than a hundred problem were solved by the MILP weighted method, the MILP AUGMECON Method, and The ASL Method. The results are then compared in terms of the optimality of the solution as well as the computational time. In this chapter, the results of the experiment are presented and analyzed. Several conclusions were drawn based on the experiments. In addition, some insights are presented in this section.

6.1 Design of Experiments

In this section, the design of the experiments conducted to examine the performance of the proposed methods is presented. The objective of the experiment is to examine how good the proposed heuristic method (ASL) in solving the E/T scheduling problem with energy consumption consideration. Another objective is to compare the results of the MILP Weighted method and the AUGMECON method in terms of computational time and the number of Pareto solution with equal number of iterations.

The response variables of the comparison between the Heuristic ASL method and the Weighed Method are: (1) the z value, and (2) the computational time. The z value is used to check how close the result of the Heuristic ASL method to the optimum value, while the computational time is used to compare the computational efficiency of the two methods.

The response variables of the comparison between the MILP Weighted method and the MILP Augmecon method are: (1) the number of points in Pareto optimum, and (2) the computational time.

Two variables i.e., the number of jobs n and the number of possible speed level s are used to generate 12 different sizes of E/T scheduling problems with energy consumption consideration. For each size, 10 different problems are generated randomly. So, the total problem used in this experiment is 120 problems. The details of the variables are as follows:

- *Problem sizes:* $n = 7, 8, 9,$ and 10
- *Speed levels:* $s = 3, 4,$ and 5 .

The data that should be generated related the variable n is the processing time of jobs at the lowest speed level P_j . In order to do that, the function “randbetween” in Excel is utilized with values of (5;20), (10;40), (10;60), (10;80), and (10;100). This generates uniform processing time. The due date for each problem is selected to be large enough to be an unrestricted problem (at least equal to the total processing time of all jobs). The data that should be generated regarding the s is the speed factor v_i and power consumption π_i . For those data, we select the data carefully because the data is discrete and the increment must be in exponential form. The examples of the speed level data can be seen in Table 13.

Table 13 The Example Data of Five Possible Speed Levels

Two variations of $s=5$					
i	V_i	π_i	i	V_i	π_i
1	1	3	1	1	2
2	1.5	6	2	1.5	4.5
3	2	10	3	2	7.5
4	2.5	15	4	2.5	11
5	3	21	5	3	15

Several examples of the experiment data are present in Table 14. The full data of the experiment can be seen in Appendix 4.

Table 14 Example Data of The Experiment

Experiment 21			Experiment 26			Experiment 42			Experiment 47			Experiment 63			Experiment 68		
DATASET DS1			DATASET DS6			DATASET DS12			DATASET DS17			DATASET DS23			DATASET DS28		
J_i	P_j		J_i	P_j		J_i	P_j		J_i	P_j		J_i	P_j		J_i	P_j	
1	6		1	5		1	12		1	11		1	17		1	24	
2	8		2	7		2	14		2	18		2	23		2	34	
3	9		3	9		3	15		3	19		3	25		3	36	
4	11		4	11		4	19		4	22		4	31		4	40	
5	14		5	13		5	24		5	26		5	35		5	48	
6	16		6	14		6	28		6	27		6	40		6	54	
7	17		7	15		7	29		7	31		7	41		7	57	
d =	119		d	112		8	30		8	32		8	47		8	59	
i	V_i	π_i	i	V_i	π_i	d	191		d	206		9	52		9	60	
1	1	3	1	1	2	i	V_i	π_i	i	V_i	π_i	d	331		d	432	
2	1.5	6	2	1.5	4.5	1	1	3	1	1	2	i	V_i	π_i	i	V_i	π_i
3	2	10	3	2	7.5	2	1.5	6	2	1.5	4.5	1	1	3	1	1	2
4	2.5	15	4	2.5	11	3	2	10	3	2	7.5	2	1.5	6	2	1.5	4.5
5	3	21	5	3	15	4	2.5	15	4	2.5	11	3	2	10	3	2	7.5

Table 14 shows some examples of the experiment data. The left side of the table shows the data of experiment 21 that consists of seven jobs from data set DS86 (see appendix 4) with a common due date $d=119$. This experiment has five possible speed levels. The table also shows the data for experiments 26, 42, 47, 63, and 68 in the same manner.

6.2 Comparison between The AUGMECON and The Weighted Method

In this study, two Mixed Integer Linear Programming models are proposed to solve the E/T problem with energy considerations, namely the AUGMECON Method and the Weighted Method. The two methods have different approaches in terms of compromising the two objectives, the E/T Performance and the Energy Consumption. Although both methods can produce the Pareto solutions, the results may slightly differ as a consequence of different approaches in building the MILP model.

The AUGMECON Method selects only one objective to define the MILP objective function and treats the other objective as an augmented constraint. In this study, the E/T performance was chosen to be the MILP objective function and the energy consumption as the augmented constraint. In contrast, the Weighted Method combines both objectives to define the MILP objective function.

6.2.1 Number of Points in the Pareto Solution Result

The AUGMECON Method and The Weighted Method has different approaches to solve the E/T problem with energy consumption consideration. In order to generate Pareto optimum solution with The AUGMECON Method, the range of energy consumption is split into R uniform intervals to produces $R+1$ different RHS values of the augmented ε -constrain. Whereas the Weighted Method splits the range value from 0 to 1 uniformly into R intervals in order to assign $R+1$ different values of w .

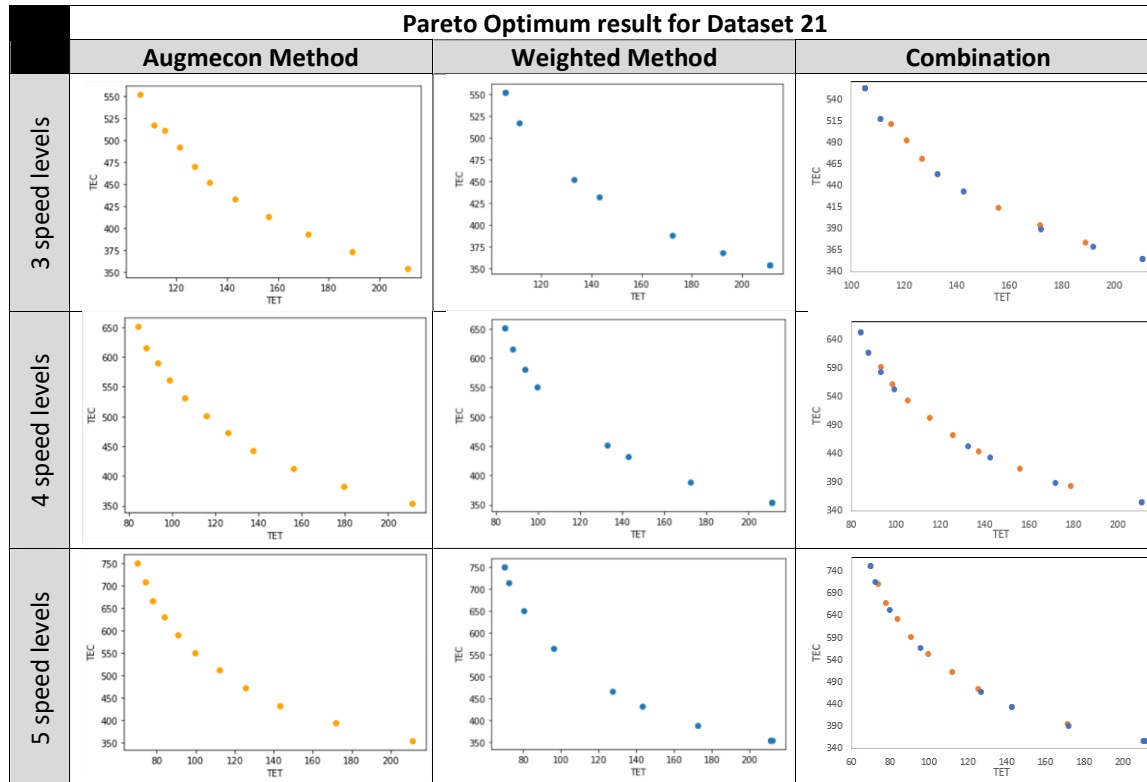


Figure 10 The Pareto Optimum for Dataset 21

Figure 10 shows the Pareto optimum result of the AUGMECON Method and the Weighted Method for Dataset 21 with three, four, and five possible speed levels. Both methods succeeded in producing the Pareto solutions. However, it shows that the Pareto points are not necessarily the same.

Processing with eleven iterations for different ϵ value, The AUGMECON Method was able to produce eleven different points in the Pareto chart, even for the case with three possible speed levels. Contrary, The Weighted Method was unable to produce eleven Pareto points in the Pareto chart while this method is iterated with eleven different values of w since

several w values produce a similar optimum point. The number of optimum points depends on the number of speed levels and the number of jobs as shown in Table 15 below.

Table 15 The Avg. Number of Optimum Point in The Pareto solution with 11 iterations

11 Iterations	The Weighted Method			The AUGMECON Method		
Number of Jobs	S= 3	S= 4	S= 5	S= 3	S= 4	S= 5
7	5.2	6.6	7.8	11	11	11
8	6.7	7.9	8.4	11	11	11
9	7.4	8.1	8.6	11	11	11
10	7.6	8	9.3	11	11	11

Table 15 presents the average number of optimum points in the Pareto solution with 11 iterations. Each method is iterated 11 times, means that the Weighted method was iterated using 11 different w value while the Augmecon method was iterated using 11 different e value. For each method, there are four different number of jobs ($n=7,8,9,10$) and three different possible speed levels ($s=3,4,5$). For example, value 11 in the bottom right of the table shows the average number of optimum points in the Pareto solution for the AUGMECON method with $n=10$ and $s=5$. Each average value above is calculated from 10 data.

Hence, The AUGMECON Method has an advantage in terms of producing the number of optimum points in the Pareto Chart compared to the Weighted Method, especially when the number of jobs and the number of speed levels are not high. In conclusion, to produce a certain number of points in the Pareto optimum, it is easier to do it with the Augmented Method because the Weighted Method may require more iterations to provide the desired number of Pareto optimum points.

6.2.2 Computational Time to Produce Pareto Optimum

To compare the computation time for solving the problem using the two methods, the AUGMECON Method and the Weighted Method were tested to solve several scheduling problems with E/T performance and energy consumption consideration. The tests were conducted on a similar device with the specification of Intel Core i5 2.4GHz Memory 8GB. The code is written in Jupyter notebook (Anaconda 3) using Python 3.0 language. The results are presented in Table 16 below.

Table 16 The Average Computational Time of AUGMECON and Weighted Method (in second)

AUGMECON METHOD				WEIGHTED METHOD			
JOBS N=7	NUMBER OF SPEED LEVEL			JOBS N=7	NUMBER OF SPEED LEVEL		
	3	4	5		3	4	5
DS1	154.5	109.8	111	DS86	58.1	52.8	70.7
DS2	116.9	122.5	135.6	DS87	67.6	58.4	70.9
DS3	169.6	144.6	124.3	DS88	68.9	59.4	84.2
DS4	144.1	151	164.2	DS89	68.9	63.1	77.9
DS5	104.1	106.2	130.9	DS90	46.3	55.1	71.1
DS6	101	111.9	107.8	DS91	49.1	67.9	59.5
DS7	131.7	153.8	128.5	DS92	66.9	67	66.1
DS8	112.3	122	112.1	DS93	55.3	63.9	61.8
DS9	86.6	87.7	78.5	DS94	50	39.2	68.4
DS10	157.9	165.9	153.5	DS95	60.6	63.9	86.7
AVG	127.87	127.54	124.64	AVG	59.17	59.07	71.73
JOBS N=8	NUMBER OF SPEED LEVEL			JOBS N=8	NUMBER OF SPEED LEVEL		
	3	4	5		3	4	5
DS11	208	214	198.3	DS61	109.4	170.5	158.1
DS12	216.5	247	196.5	DS62	117.5	126.1	198.2
DS13	247.8	237.8	217.2	DS63	114.6	116.2	173
DS14	310.3	222.9	190.7	DS64	112.4	124.8	176.8
DS15	236.3	349.8	188.3	DS65	194.8	125.5	177.4
DS16	329.1	201.2	193	DS66	133.5	114.7	116.8
DS17	228	352.4	241.2	DS67	129	136.7	146.6
DS18	251.2	265.3	226.3	DS68	167.5	141	141.3
DS19	163.9	229.9	101	DS69	78.2	72.9	68.3
DS20	250.8	298.4	252.9	DS70	144.8	151.9	159.8
AVG	244.19	261.87	200.54	AVG	130.17	128.03	151.63

AUGMECON METHOD				WEIGHTED METHOD			
JOBS N=9	NUMBER OF SPEED LEVEL			JOBS N=9	NUMBER OF SPEED LEVEL		
	3	4	5		3	4	5
DS21	1152.6	575.7	599.8	DS36	501.8	554.4	366.6
DS22	1038.3	569	633.6	DS37	469.6	376.9	362.1
DS23	864.4	685.6	701.2	DS38	545.3	381	379.9
DS24	1276.3	917.6	686.7	DS39	519.7	375.4	350.2
DS25	2181.6	970.3	786.8	DS40	577.1	371.9	435.5
DS26	1330.6	770	564.6	DS41	481.8	327.4	311.4
DS27	895	995.6	694.7	DS42	405.3	384.8	386
DS28	1124	772.8	841.8	DS43	458.2	444.4	481.2
DS29	777.7	267.1	249	DS44	195.1	160.9	161.4
DS30	1189.2	696	769.4	DS45	416.6	412.1	427.6
AVG	1182.97	721.97	652.76	AVG	457.05	378.92	366.19
JOBS N=10	NUMBER OF SPEED LEVEL			JOBS N=10	NUMBER OF SPEED LEVEL		
	3	4	5		3	4	5
DS31	3239.2	2193	2587.6	DS1	1426.25	1451.9	1226.1
DS32	5857.7	5092.4	5794.6	DS2	3136.81	2746.8	2976.5
DS33	4033.4	4118.9	3370.4	DS3	2234.77	1569	1404.1
DS34	586.5	856.7	968.2	DS4	222.6	259.9	204.4
DS35	45471	20400	18005	DS5	8242.3	6205.9	6075
AVG	11837.56	3065.25	3180.2	AVG	3052.5	2446.7	2377.2

Table 16 shows the average computational time of the MILP Weighed method vs MILP Augmecon method in seconds. The left side of the table is the computational time of the Weighted method and the right side is for the AUGMECON method. DS1 means Data Set 1. The number 154.5, at the top left of the table, is the average computational time of data set 1 (n=7) with three possible speed levels (s=3) solved by the AUGMECON method.

6.2.3 Testing the hypothesis

In order to compare the results of both methods, statistical testing was used to compare the mean of the computational time for each method. Since the size of the sample is less than

30, a two-sample t-test was selected. $H_0: \mu_{The\ Wieghted\ Method} = \mu_{The\ Augmecon\ Method}$. Using $\alpha=0.05$, the P-value of all comparisons for each dataset and each number of speed levels are less than the alpha value. Hence, we reject the H_0 in each test and conclude that the mean of the computational time of the MILP Weighted Method is statistically different compared to the mean of the computational time of the AUGMECON Method.

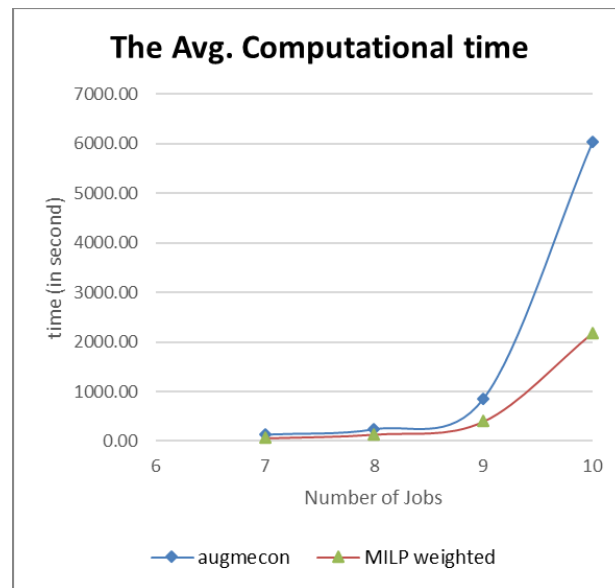


Figure 11 The Average Computational time

On average, the Weighted Method performs better in terms of computational time compared to the AUGMECON method. As shown in Figure 11, for problems with sizes of 7 jobs up to 10 jobs, the computational time of the AUGMECON method is always higher than the computational time of the Weighted Method. The trend shows that the AUGMECON will always take a longer time compared to the Weighted Method. For the problems with 7 jobs, the computational time of the AUGMECON Method is almost twice the time of the Weighted Method. The graph shows that the difference in computation time is increases exponentially with the size of the problem.

6.3 Comparison between the MILP Weighted Method and the ASL Heuristic

In this section, the performance of the proposed heuristic method in solving the basic E/T scheduling problem with energy consumption consideration is examined. Here, the solution produced by the Ascending Speed Level (ASL) Heuristic was compared with the results from MILP Weighted solution. Around 120 different cases were solved with both methods in order to draw a conclusion regarding the performance of the ASL. The size of the problems is varying with the number of jobs from 7 to 10 and the number of speed levels from 3 to 5.

6.3.1 Results

The MILP Weighted Method has the advantage that the optimal result is the global optimal solution. This is not the case with the Heuristic Method, because it does not always find a global optimum and may be trapped in a local optimum. Therefore, the comparison of the optimization results generated by the ASL versus the MILP Weighted Method provides important information about how effective the proposed heuristic method is in finding the optimal solution.

Two parameters are used to compare the performance of the two methods, i.e., the optimum integrated value (Z Value) and the configuration of the speed level of each job in optimum conditions. There are 120 scheduling problems with 4 different number of jobs and 3 different number of speeds levels. In order to produce a Pareto optimum result, each job

was iterated with eleven different values of w that distributed uniformly from 0 to 1. Thus, there are 1320 Pareto points that can be compared.

Table 17 shows the comparison of the optimization result data between the ASL Method and the MILP Weighted Method. In this table, the number of ASL Pareto points in each category is presented. The parameter of the optimum integrated value has two categories: “Found the Global Optimum” and “Near Global Optimum”. “Similar” and “Slightly Different” are two categories for parameters of Speed Level Configuration. The data in Table 17 shows the number of problems in which the optimal Z is found and the number of optimum speed level configuration identified.

Table 17 The Performance of ASL Optimization Method

Number of Jobs	Number of Speed levels	Z value	Speed level configuration
		Global Optimum Found	Found
7	3	110	110
	4	110	110
	5	99	110
8	3	110	110
	4	110	110
	5	98	107
9	3	108	110
	4	108	110
	5	101	110
10	3	103	108
	4	108	108
	5	101	107
Sum		1266	1310
Percentage		95.9%	99.2%

Results estimates that around 95.9% of the optimum points produced by the ASL method are global optimum.

In terms of speed level configuration, the comparison shows that 99.2% of the ASL speed level configuration is identical to the results of the MILP Weighted Method.

Table 18 highlights the results for specific w values in which both methods have different speed level configurations. For experiment number 56 with $w=0.5$, the speed level configuration of the Weighted method and the ASL has different at the first job where the speed level of the first job produced by the Weighted method equal to 5, whereas it is 4 for the configuration produced by the ASL method.

Table 18 The Comparison of Different Speed Level Configuration

Experiment	W	Method	TET	TEC	configuration	Z value
56	0.5	MILP Weighted	78.9	279.4	5-4-4-3-3-1-1-1	0.373
		ASL Heuristic	80.2	276.4	4-4-4-3-3-1-1-1	0.373
57	0.5	MILP Weighted	176.7	551.7	4-4-4-3-2-1-1-1	0.373
		ASL Heuristic	184	535.2	4-4-4-2-2-1-1-1	0.373
	0.8	MILP Weighted	103	834	5-5-5-5-5-5-5-1	0.200
		ASL Heuristic	106.9	799.2	5-5-5-5-5-4-4-1	0.200
97	0.4	MILP Weighted	444.7	600.5	3-2-2-1-1-1-1-1-1	0.367
		ASL Heuristic	424.6	620.4	3-2-2-2-1-1-1-1-1	0.366
	0.5	MILP Weighted	369	684.5	3-3-3-2-3-1-1-1-1	0.384
		ASL Heuristic	380	668	3-3-3-2-2-1-1-1-1	0.384
105	0.5	MILP Weighted	340.7	1116	4-3-3-3-3-2-2-1-1-1	0.353
		ASL Heuristic	364.7	1068	4-3-3-2-2-2-2-1-1-1	0.353
110	0.3	MILP Weighted	599.3	748	2-2-2-1-1-1-1-1-1-1	0.297
		ASL Heuristic	632.7	723	2-2-1-1-1-1-1-1-1-1	0.297
112	0.6	MILP Weighted	73.7	421	5-5-5-4-4-3-3-2-2-1	0.321
		ASL Heuristic	76.6	409.9	5-4-4-4-4-3-3-2-2-1	0.321
114	0.5	MILP Weighted	206.7	1152	5-4-4-3-3-3-3-2-2-1	0.346
		ASL Heuristic	207.3	1150	4-4-4-3-3-3-3-2-2-1	0.346
117	0.7	MILP Weighted	187.8	1120.8	5-5-5-5-5-5-5-3-3-1	0.283
		ASL Heuristic	200.7	1063.1	5-5-5-5-5-5-5-2-2-1	0.283

When the speed level configuration at optimum condition produces by ASL Method is different from the results of the MILP Weighted Method, it doesn't necessarily mean that the result of the ASL Method is not optimum or is trapped in the local optimum. But it may be an alternative optimum point especially when the z values are equal. Consider for example experiment number 112 (see Appendix 4). Figure 12 combines the Pareto optimum produced by the ASL Heuristic and MILP Weighted method. The optimum point of both methods are equal except for $w=0.6$. The optimum point of the Weighted Method and the ASL method for $w = 0.6$ are different as marked with the orange circle, but the Z value of both points are equal i.e., 0.321. This means that the ASL method found an alternative optimum point.

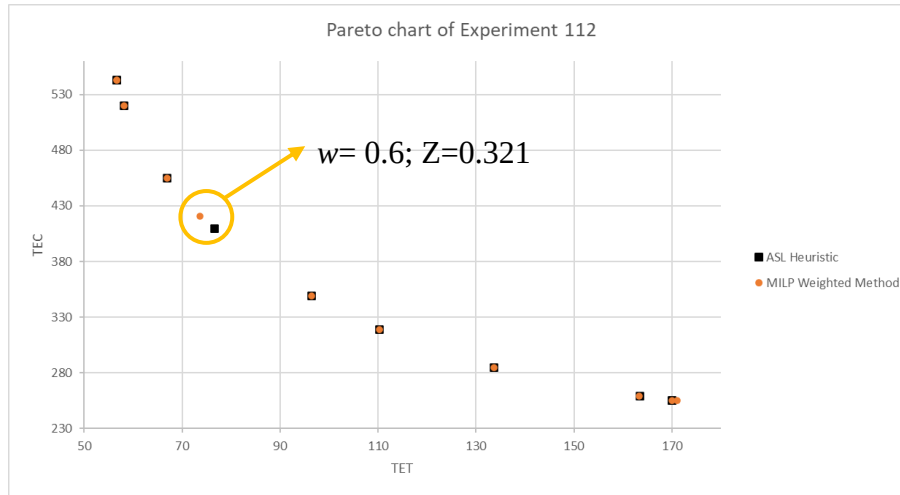


Figure 12 Pareto Optimum of Experiment 112

In conclusion, experiments show that the ASL method has an excellent performance in finding the optimum solution for the scheduling problem with E/T performance and energy consumption consideration for the tested problems. Here, we did not compare the job

sequence in the optimum condition since the E/T basic problem may have more than one optimum job sequence configuration, see Baker [4].

6.3.2 Computational time to Produce Pareto Optimum

One of the drawbacks of the MILP approach in solving multi-objective optimization problems is the computational time requirement. Therefore, many researchers try to construct heuristics for finding near optimal solutions. In this section, the computational time between ASL method and MILP Weighted method is compared to analyze the performance of the two methods in producing Pareto optimum solutions for the problem under consideration. Table 19 shows the average computational time of the Weighted method and ASL Method in solving the problem. For example, the time required to solve the problem of 7 jobs and 3 possible speed levels by the MILP Weighted method is in average 59.17 seconds. Figure 13 shows a chart of the average computational time of the Weighted method for all tested problems. It shows that the computation time of this method is sensitive to the number of jobs (increase exponentially with the number of jobs) but less sensitive to the number of possible speed levels.

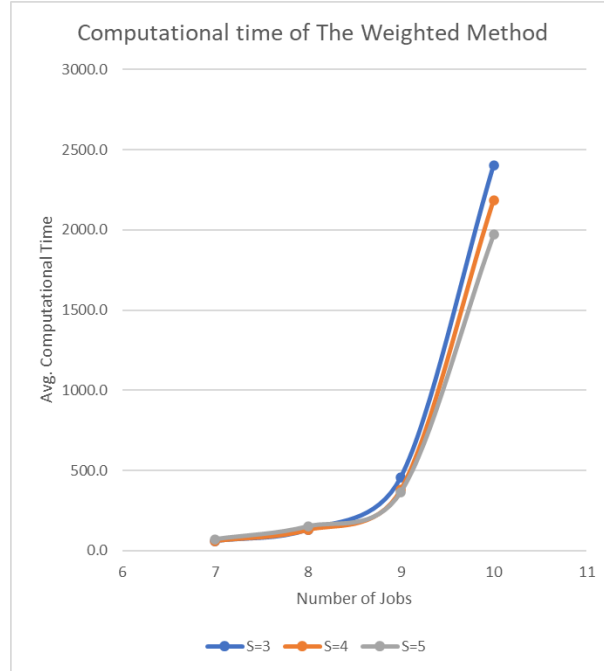


Figure 13 The Avg. Computational Time of The Weighted Method

Several insights can be drawn from table 19. First, the computational time of the MILP weighted method increases exponentially as the number of jobs increase as shown in figure 13. For problems with the size of 11 jobs, the time solution is more than 1 day. On the other hand, increasing the number of speed levels has no significant effect on the computational time since the P-value is greater than alpha ($\alpha = 0.05$).

Table 19 The Avg. Computational time of Weighted Method and ASL Method (in Seconds)

Number of Jobs	MILP Weighted Method			ASL Method		
	S=3	S=4	S=5	S=3	S=4	S=5
7	59.17	59.07	71.73	0.03	0.03	0.07
8	130.17	128.03	151.63	0.06	0.08	0.09
9	457.05	378.92	366.19	0.06	0.07	0.06
10	2404.39	2187.22	1970.20	0.05	0.07	0.07

Second, the computational time of the ASL method is much faster than MILP. All data shows that the computational time is less than one second to produces a Pareto optimum solution for the largest problem size in the experiment. Preliminary tests indicate that this heuristic solve problems of size $n=20$ and $s=10$ in less than 1 second (0.55 second). Therefore, the ASL method has superior performance compared to the MILP Weighted method in terms of computational time.

6.4 Conclusions

Based on the results of the experiment and the analysis in the previous section, several conclusions can be summarized regarding the proposed methodology for solving the early/tardy scheduling problem with energy consumption consideration as follows:

1. In this research, the study of early/tardy scheduling problems with energy consumption consideration was conducted. Two MILP models and a heuristic model are proposed i.e., the MILP Weighted Method, The AUGMECON Method, and the Ascending Speed Level (ASL) Heuristic.
2. The two proposed MILP models have different approaches in compromising the two objectives, minimizing E/T and energy consumption. The AUGMECON method uses only one of the objective performances (early/tardy) to define the MILP objective function and treating the energy consumption objective as an augmented ϵ -constrain. Differently, The Weighted Method uses both objectives to define the MILP objective function by assigning weights for each objective.

3. In terms of the number of Pareto optimum solutions generated, the AUGMECON method performs better compared to the Weighted Method. In terms of the computational time, the Weighted Method outperforms the AUGMECON method since it needs less time to find the solution.
4. The time to solve the problem with the MILP methods increases exponentially with the number of jobs.
5. The ASL Heuristic has an approach similar to the MILP Weighted Method in defining the minimization objective function. The two contradicted objective functions are combined by assigning a weight for each objective function. Also, the Normal Boundary Intersection was utilized to calculate the integrated value in order to provide a robust calculation method. Therefore, the performance of the ASL can be compared with the MILP Weighted Method.
6. The experiments show that the ASL Method has an excellent performance in finding the Pareto solution for a scheduling problem with E/T performance and energy consumption consideration for the tested problem sizes.
7. The experiment shows that the ASL Method is superior in terms of the Computational time compared with the MILP weighted method for the tested problem sizes.

6.5 Recommendation

The study in this thesis can be extended in several directions. First, the ASL heuristic needs to be tested for large and medium size problems. since the MILP weighted methods takes long time to solve the medium and large size problems, the performance of the ASL

heuristic can be tested by either comparing with another well-known heuristic for medium-large scheduling problem or by limiting the computational time of the MILP method.

2. Some machines in manufacturer can change speed continuously, other than step wise changes. Hence it is necessary to study the problem in this thesis under continuous speed levels.

3. The proposed MILP models in this study can solve the E/T scheduling problem with energy consumption consideration under restricted and unrestricted due dates. But the ASL heuristic is designed only for scheduling with an unrestricted due date. Hence, the study can be extended to develop a heuristic method to solve the problem under restricted common due date.

4. Many assumptions can be relaxed in order to extend this study in other directions.

Appendix 1

The code below is the Python Code of The MILP Weighted Method (for example 4):

```
1  from pulp import *
2
3  # SETS
4  JOBS = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]    # J
5  VSPEED = [1, 2, 3]                      # v
6
7  # PARAMETES
8  Pj = {1: 4,
9        2: 5,
10       3: 7,
11       4: 8,
12       5: 10,
13       6: 11,
14       7: 12,
15       8: 15,
16       9: 18,
17       10: 20}
18  Vi = {1: 1,
19        2: 1.5,
20        3: 2}
21  Phi = {1: 3,
22         2: 6,
23         3: 13}
24  d = 80
25  M = 2000
26
27  # DECISION VARIABLES
28  Sj = LpVariable.dicts("StartTime", JOBS, lowBound=0)
29  Cj = LpVariable.dicts("CompletionTime", JOBS, lowBound=0)
30  Ej = LpVariable.dicts("Earliness", JOBS, lowBound=0)
31  Tj = LpVariable.dicts("Tardiness", JOBS, lowBound=0)
32  TET = LpVariable("TET", lowBound=0)
33  TEC = LpVariable("TEC", lowBound=0)
34  Xjv = LpVariable.dicts("Xjv", [(j,v) for j in JOBS for v in VSPEED], lowBound=0, upBound=1, cat="Integer")
35  Yjk = LpVariable.dicts("Yjk", [(j,k) for j in JOBS for k in JOBS if k!=j], lowBound=0, upBound=1, cat="Integer")
36
37  # OBJECTIVE FUNCTIONS TO FIND UTOPIAN AND NADIR POINT
38  TET_OBJ = lpSum(Ej[j] + Tj[j] for j in JOBS)
39  TEC_OBJ = lpSum(Pj[j] * Phi[v] / Vi[v] * Xjv[(j,v)] for j in JOBS for v in VSPEED)
40
41  # FUNCTION FOR MAIN CONSTRAINTS
42  def MainConstrain (x):
43      # CONSTRAINTS
44      #CONSTRAIN_3
45      for j in JOBS:
46          x += (lpSum(Xjv[(j,v)] for v in VSPEED) == 1)
47      #CONSTRAIN_4
48      for j in JOBS:
49          x += (Cj[j] == Sj[j] + lpSum(Pj[j] / Vi[v] * Xjv[(j,v)] for v in VSPEED))
50      #CONSTRAIN_5
51      for j in JOBS:
52          x += (Cj[j] + Ej[j] - Tj[j] == d)
53      #CONSTRAIN_6
54      for j in JOBS:
55          for k in JOBS:
56              if k == j:
57                  continue
58              x += (Sj[k] >= Cj[j] - M * (1 - Yjk[(j,k)]))
59      #CONSTRAIN_7
60      for j in JOBS:
61          for k in JOBS[j:]:
62              x += (Yjk[(j,k)] + Yjk[(k,j)] == 1)
63
64  # OPTIMIZE TEC ONLY
65  TECopt = LpProblem('TECopt', LpMinimize)
66  TECopt += TEC_OBJ
67  MainConstrain(TECopt)
68  TECopt.solve()
69  TECU = value(TECopt.objective)
70  print("TECU =", TECU)
71
72  # OPTIMIZE TET ONLY
73  TETopt = LpProblem('TETopt', LpMinimize)
74  TETopt += TET_OBJ
75  MainConstrain(TETopt)
```

```

76 TETOpt.solve()
77 TETU = value(TETOpt.objective)
78 print("TETU =", TETU)
79
80 # FIND TECN WITH TET OPTIMAL
81 TECNOpt = LpProblem('TECNOpt', LpMinimize)
82 TECNOpt += TEC_OBJ
83 TECNOpt += (TET_OBJ <= TETU)
84 MainConstrain(TECNOpt)
85 TECNOpt.solve()
86 TECN = value(TECNOpt.objective)
87 print("TECN =", TECN)
88
89 # FIND TETN WITH TEC OPTIMAL
90 TETNOpt = LpProblem('TETNOpt', LpMinimize)
91 TETNOpt += TET_OBJ
92 TETNOpt += (TEC_OBJ <= TECU)
93 MainConstrain(TETNOpt)
94 TETNOpt.solve()
95 TETN = value(TETNOpt.objective)
96 print("TETN =", TETN)
97
98 # INTEGRATED MODEL
99 import numpy as np
100 import time
101
102 # LIST OF RESULT
103 Zlist = []
104 TEClist = []
105 TETlist = []
106
107 # DEFINE THE INTEGRATED PROBLEM
108 def IntegratedProb (i,x):
109     W = i
110     x = LpProblem(str(x), LpMinimize)
111     x += (W * (TET-TETU) / (TETN - TETU) + (1-W) * (TEC - TECU) / (TECN - TECU))
112     MainConstrain (x)
113     x += (TEC == TEC_OBJ)
114     x += (TET == TET_OBJ)
115
116     start = time.perf_counter()
117     x.solve()
118     SolutionTime = time.perf_counter() - start
119
120     Zlist.append(value(x.objective))
121     TETlist.append(TET.varValue)
122     TEClist.append(TEC.varValue)
123
124     print('Status:', LpStatus[x.status])
125     print(i, "=", value(x.objective))
126
127     print('solution time:', SolutionTime)
128     print("TET=", TET.varValue)
129     print("TEC=", TEC.varValue)
130     for v in Sj:
131         print (Sj[v].name, "=", Sj[v].varValue)
132     for v in Xjv:
133         if Xjv[v].varValue > 0:
134             print (Xjv[v].name, "=", Xjv[v].varValue)
135     print('-----')
136
137 Wlist=[]
138 for i in np.arange (0.0, 1.1, 0.10):
139     Wlist.append(round(i,1))
140 Wlist[0] += 0.001
141 Wlist[10] -= 0.001
142
143 ProbNo = 1
144 for i in Wlist:
145     IntProb = "IntProb" + str(ProbNo)
146     IntegratedProb (i, IntProb)
147     ProbNo += 1
148
149 print("Zlist")
150 print(Zlist)
151 print("TETlist")
152 print(TETlist)
153 print("TEClist")
154 print(TEClist)
155
156 from matplotlib import pyplot as plt
157 plt.scatter(TETlist,TEClist, marker = 'o')
158 plt.xlabel("TET")
159 plt.ylabel("TEC")
160 plt.show()

```

Appendix 2

The code below is the Python Code of The AUGMECON Method (for example 4):

```
1  from pulp import *
2
3  # SETS
4  JOBS = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]    # J
5  VSPEED = [1, 2, 3]                      # v
6
7  # PARAMETES
8  Pj = {1: 4,
9        2: 5,
10       3: 7,
11       4: 8,
12       5: 10,
13       6: 11,
14       7: 12,
15       8: 15,
16       9: 18,
17       10: 20}
18  Vi = {1: 1,
19        2: 1.5,
20        3: 2}
21  Phi = {1: 3,
22         2: 6,
23         3: 10}
24  d = 80
25  M = 2000
26
27  # DECISION VARIABLES
28  Sj = LpVariable.dicts("StartTime", JOBS, lowBound=0)
29  Cj = LpVariable.dicts("CompletionTime", JOBS, lowBound=0)
30  Ej = LpVariable.dicts("Earliness", JOBS, lowBound=0)
31  Tj = LpVariable.dicts("Tardiness", JOBS, lowBound=0)
32  TET = LpVariable("TET", lowBound=0)
33  TEC = LpVariable("TEC", lowBound=0)
34  Xjv = LpVariable.dicts("Xjv", [(j,v) for j in JOBS for v in VSPEED], lowBound=0, upBound=1, cat="Integer")
35  Yjk = LpVariable.dicts("Yjk", [(j,k) for j in JOBS for k in JOBS if k!=j], lowBound=0, upBound=1, cat="Integer")
36
37  # OBJECTIVE FUNCTIONS TO FIND UTOPIAN AND NADIR POINT
38  TET_OBJ = lpSum(Ej[j] + Tj[j] for j in JOBS)
39  TEC_OBJ = lpSum(Pj[j] * Phi[v] / Vi[v] * Xjv[(j,v)] for j in JOBS for v in VSPEED)
40
41  # FUNCTION FOR MAIN CONSTRAINTS
42  def MainConstrain (x):
43      # CONSTRAINTS
44      #CONSTRAIN_3
45      for j in JOBS:
46          x += (lpSum(Xjv[(j,v)] for v in VSPEED) == 1)
47      # CONSTRAIN_4
48      for j in JOBS:
49          x += (Cj[j] == Sj[j] + lpSum(Pj[j] / Vi[v] * Xjv[(j,v)] for v in VSPEED))
50      # CONSTRAIN_5
51      for j in JOBS:
52          x += (Cj[j] + Ej[j] - Tj[j] == d)
53      # CONSTRAIN_6
54      for j in JOBS:
55          for k in JOBS:
56              if k == j:
57                  continue
58              x += (Sj[k] >= Cj[j] - M * (1 - Yjk[(j,k)]))
59      # CONSTRAIN_7
60      for j in JOBS:
61          for k in JOBS[j:]:
62              x += (Yjk[(j,k)] + Yjk[(k,j)] == 1)
63
64  # OPTIMIZE TEC ONLY
65  TECopt = LpProblem('TECopt', LpMinimize)
66  TECopt += TEC_OBJ
67  MainConstrain(TECopt)
68  TECopt.solve()
69  TECU = value(TECopt.objective)
70  print("TECU =", TECU)
71
72  # OPTIMIZE TET ONLY
73  TETopt = LpProblem('TETopt', LpMinimize)
74  TETopt += TET_OBJ
75  MainConstrain(TETopt)
```

```

76 TETOpt.solve()
77 TETU = value(TETOpt.objective)
78 print("TETU =", TETU)
79
80 # FIND TECN WITH TET OPTIMAL
81 TECNOpt = LpProblem('TECNOpt', LpMinimize)
82 TECNOpt += TEC_OBJ
83 TECNOpt += (TET_OBJ <= TETU)
84 MainConstrain(TECNOpt)
85 TECNOpt.solve()
86 TECN = value(TECNOpt.objective)
87 print("TECN =", TECN)
88
89 # FIND TETN WITH TEC OPTIMAL
90 TETNOpt = LpProblem('TETNOpt', LpMinimize)
91 TETNOpt += TET_OBJ
92 TETNOpt += (TEC_OBJ <= TECU)
93 MainConstrain(TETNOpt)
94 TETNOpt.solve()
95 TETN = value(TETNOpt.objective)
96 print("TETN =", TETN)
97
98 # AUGMECON PROCEDURE TO INTEGRATE BOTH OBJECTIVES
99 # TET WAS USED AS OBJECTIVE AND TEC AS CONSTRAIN
100 import time
101
102 TEclstRHS = []
103 Zlist = []
104 TETlist = []
105 TEclst = []
106
107 # PARAMETER AUGMECON
108 Point = 11
109
110 # CALCULATE E_CONS FOR TEC
111 interval = TECN - TECU
112 step = interval / (Point - 1)
113 for i in range(Point):
114     e_cons = TECN - i * step
115     TEclstRHS.append(round(e_cons, 2))
116
117 # GENERATE SUB PROBLEM AUGMECON
118 eps = 0.001
119
120 # DEFINE LP PROBLEM FOR AUGMECON
121 S1 = LpVariable('Slack', lowBound=0)
122 stop_condition = 0
123
124 def AUGstep(x, i):
125     global stop_condition
126
127     x = LpProblem("problem-" + str(x), LpMinimize)
128     x += TET_OBJ - eps * S1 / interval
129     x += (TEC_OBJ + S1 == TEclstRHS[i])
130     MainConstrain(x)
131
132     start = time.perf_counter()
133     x.solve()
134     SolutionTime = round((time.perf_counter() - start), 1)
135     if LpStatus[x.status] != "Optimal":
136         stop_condition = 1
137
138     Zlist.append(round(value(x.objective), 3))
139     TETlist.append(round(value(TET_OBJ), 1))
140     TEclst.append(round(value(TEC_OBJ), 1))
141     print('Status:', LpStatus[x.status])
142     print('solution time:', SolutionTime)
143     print("Optimal Solution to the problem:", value(x.objective))
144     print("TET:", value(TET_OBJ))
145     print("TEC:", value(TEC_OBJ))
146     print("Individual decision variables:")
147     for v in Sj:
148         print(Sj[v].name, "=", Sj[v].varValue)
149     for v in Xjv:
150         if Xjv[v].varValue > 0:
151             print(Xjv[v].name, "=", Xjv[v].varValue)

```

```

151     print("S1 =", S1.varValue)
152
153 # RUN THE ITERATION STEP
154 start_global = time.perf_counter()
155 for i in range (Point):
156     AUGprob = "Problem_AUG" + str(i)
157     print (AUGprob, '-----')
158     AUGstep(AUGprob,i)
159     if stop_condition == 1:
160         break
161 total_solution_time= round((time.perf_counter() - start_global),1)
162 print("-----")
163 print(TEClist)
164 print(TETlist)
165 print("total solution time=",total_solution_time)
166
167 # PRINT THE PARETO FRONTIER OPTIMIZATION CHART
168
169 from matplotlib import pyplot as plt
170 #plt.xlim(90, 310)
171 #plt.ylim(310, 740)
172 plt.scatter(TETlist,TEClist, marker = 'o')
173 plt.xlabel("TET")
174 plt.ylabel("TEC")
175 plt.show()

```

Appendix 3

The code below is the Python Code of The ASL Method (for example 4):

```
1  # INPORT THE LIBRARIES
2
3  import random
4  import copy
5  import numpy as np
6
7  # DEFINE THE REQUIRED FUNCTIONS
8
9  def recalculate_p(x):
10     for i in range (len(x)):
11         x[i]['p'] = round ( x[i]['t'] / Vi[x[i]['SL']], 2)
12
13  def calculate_TE(x): # create function to calculate TE
14     TE_value = 0
15     for i in range (len(x)):
16         earliness = i*x[i]['p']
17         TE_value += earliness
18     return TE_value
19
20  def calculate_TT(x): # create function to calculate TT
21     TT_value = 0
22     for i in range (len(x)):
23         tardiness = (len(x)-i)*x[i]['p']
24         TT_value += tardiness
25     return TT_value
26
27  def calculate_TEC(x): # create function to calculate TEC
28     TEC_value = 0
29     for i in range (len(x)):
30         energy = round ( x[i]['p'] * PEi[x[i]['SL']], 2)
31         TEC_value += energy
32     return TEC_value
33
34  def algorithm1 (x): # create function to calculate TECU & TETN
35     global TECU, TETN
36     vlist = [] # ----- set all speed level = 1
37     for i in range (len(x)):
38         vlist.append(1)
39     for i in range (len(x)):
40         x[i]['SL'] = vlist[i]
41     recalculate_p(x) # ----- recalculate processing time
42     x.sort(reverse=True, key=lambda k: k['p']) # --- List job in LPT order based on P
43     setB = x[::2] # ----- split into setB and setA
44     setA = x[1::2]
45     setB.sort(reverse=True, key=lambda k: k['p']) # --- short setB in LPT and setA in SPT
46     setA.sort(reverse=False, key=lambda k: k['p'])
47     TE= calculate_TE(setB) #----- calculate the performance measure
48     TT= calculate_TT(setA)
49     TET = TE+TT
50     TEC = calculate_TEC(x)
51
52     TECU = TEC # ----- set TECU & TETN
53     TETN = TET
54
55  def algorithm2 (x): # create function to calculate TECN & TETU
56     global TECN, TETU
57     x.sort(reverse=False, key=lambda k: k['p']) # set speed Level = 3 except the longest job
58     vlist = []
59     for i in range (len(x)-1):
60         vlist.append(3)
61     for i in range (len(vlist)):
62         x[i]['SL'] = vlist[i]
63     recalculate_p(x) # ----- recalculate processing time
64     x.sort(reverse=True, key=lambda k: k['p']) # --- List job in LPT order based on P
65     setB = x[::2] # ----- split into setB and setA
66     setA = x[1::2]
67     setB.sort(reverse=True, key=lambda k: k['p']) # --- short setB in LPT and setA in SPT
68     setA.sort(reverse=False, key=lambda k: k['p'])
69     TE= calculate_TE(setB) #----- calculate the performance measure
70     TT= calculate_TT(setA)
71     TET = TE+TT
72     TEC = calculate_TEC(x)
73     TECN = TEC # ----- set TECN & TETU
74     TETU = TET
75
76  def Iter_Step():
```

```

76 global Opt_Z, Opt_JOBSEC, Opt_TEC, Opt_TET, K, L, Opt_SL_list
77 JOBSEC = copy.deepcopy(JOBS) # ----- copy JOBS data before iteration
78 JOBSEC.sort(key=lambda k: k['t']) # --- sort job in SPT order
79 SL_list = copy.deepcopy(Opt_SL_list)
80 SL_list[L] = K # ----- assign new speed level combination
81 for i in range(len(JOBSEC)):
82     JOBSEC[i]['SL'] = SL_list[i]
83     recalculate_p(JOBSEC) # ----- update processing time of each job
84     JOBSEC.sort(reverse=True, key=lambda k: k['p']) # --- sort jobs in LPT order
85     setB = JOBSEC[::2] # ----- split and sort job in setB and setA
86     setA = JOBSEC[1::2]
87     setB.sort(reverse=True, key=lambda k: k['p'])
88     setA.sort(reverse=False, key=lambda k: k['p'])
89     newJobSec = setB + setA
90     TE = calculate_TE(setB) # ----- calculate performance value
91     TT = calculate_TT(setA)
92     TET = TE+TT
93     TEC = calculate_TEC(newJobSec)
94     z = W*((TET-TETU)/(TETN-TETU))+(1-W)*((TEC-TECU)/(TECN-TECU))
95     if z < Opt_Z: # ----- cek Z value
96         Opt_JOBSEC = copy.deepcopy(newJobSec)
97         Opt_TEC = TEC
98         Opt_TET = TET
99         Opt_Z = z
100        Opt_SL_list = copy.deepcopy(SL_list)
101
102        if L == (len(JOBS)-2): # Len(JOBS)-2: Last posisiton before last jobs
103            L = 0
104            K += 1
105        else:
106            L += 1
107    else:
108        K += 1
109        if K <= len(Vi):
110            L = 0
111
112def get_solution (x):
113    global Opt_TEC_list, Opt_TET_list, Opt_Z_list, List_Opt_SL_list, L, K, W, Opt_JOBSEC, Opt_TEC, Opt_TET, Opt_Z,
114    Opt_SL_list
115
116    W = x
117    Opt_Z = 2 # or any number greater than 1
118    Opt_TET = 0
119    Opt_TEC = 0
120    Opt_JOBSEC = []
121
122    L = len(JOBS)-2 # L: index of job that will be increased its SL (start from 0 not 1)
123    K = 1 # K: new Speed Level
124
125    Opt_SL_list = [] # Generate list of speed Level for all job (in SPT order) equal 1
126    for i in range (len(JOBS)):
127        Opt_SL_list.append(1)
128
129    while K <= len(Vi):
130        Iter_Step()
131
132        Opt_TEC_list.append(Opt_TEC)
133        Opt_TET_list.append(Opt_TET)
134        Opt_Z_list.append(Opt_Z)
135        List_Opt_SL_list.append(Opt_SL_list)
136
137        print ('W:', W)
138        print ('TEC:', Opt_TEC)
139        print ('TET:', Opt_TET)
140        print ('Z:', Opt_Z)
141        for i in Opt_JOBSEC:
142            print (i)
143
144    # ENTER THE DATA
145    Vi = {1: 1, # speed factor : Speed machining
146          2: 1.5,
147          3: 2}
148    PEi = {1: 3, # speed factor : power energy
149           2: 6,
150           3: 13}

```

```

151 JOBS = [
152     {'job':1, 't': 4 },
153     {'job':2, 't': 5 },
154     {'job':3, 't': 7 },
155     {'job':4, 't': 8 },
156     {'job':5, 't': 10},
157     {'job':6, 't': 11},
158     {'job':7, 't': 12},
159     {'job':8, 't': 15},
160     {'job':9, 't': 18},
161     {'job':10, 't': 20}
162 ]
163 for item in JOBS:    # add key of Speed Level and actual processing time
164     item.update({'SL':1})
165     item.update({'p':0})
166 recalculate_p (JOBS)
167
168 # DEFINE THE VARIABLES
169 Opt_TEC_list = []
170 Opt_TET_list = []
171 Opt_Z_list = []
172 List_Opt_SL_list = []
173
174 # Calculate TETU and TECN
175 algorithm1 (JOBS)
176
177 # Calculate TECU and TETN
178 algorithm2 (JOBS)
179
180 print(TETU, TETN)
181 print(TECU, TECN)
182
183 # GENERATE Z LIST
184 Wlist=[]
185 for i in np.arange (0.0, 1.1, 0.10):
186     Wlist.append(round(i,1))
187 Wlist[0] += 0.001
188 Wlist[10] -= 0.001
189
190 # start iteration
191 import time
192
193 start = time.perf_counter()
194 for i in Wlist:
195     get_solution (i)
196     solution_time = time.perf_counter() - start
197
198 print('Z List:', Opt_Z_list )
199 print('TET List:', Opt_TET_list )
200 print('TEC List:', Opt_TEC_list )
201
202 for i in List_Opt_SL_list:
203     print (i)
204     print ('solution time=', round(solution_time,2))
205
206 from matplotlib import pyplot as plt
207 plt.scatter(Opt_TET_list, Opt_TEC_list, marker = 'o')
208 plt.xlabel("TET")
209 plt.ylabel("TEC")
210 plt.show()

```

Appendix 4

Experiment 1 DATASET DS1	Experiment 2 DATASET DS2	Experiment 3 DATASET DS3	Experiment 4 DATASET DS4	Experiment 5 DATASET DS5	Experiment 6 DATASET DS6	Experiment 7 DATASET DS7	Experiment 8 DATASET DS8	Experiment 9 DATASET DS9	Experiment 10 DATASET DS10
J_i P_j	J_i P_j	J_i P_j	J_i P_j	J_i P_j	J_i P_j	J_i P_j	J_i P_j	J_i P_j	J_i P_j
1 6	1 12	1 17	1 23	1 19	1 5	1 11	1 24	1 12	1 38
2 8	2 14	2 23	2 30	2 49	2 7	2 18	2 34	2 13	2 40
3 9	3 15	3 25	3 36	3 50	3 9	3 19	3 36	3 16	3 51
4 11	4 19	4 31	4 42	4 52	4 11	4 22	4 40	4 19	4 59
5 14	5 24	5 35	5 46	5 54	5 13	5 26	5 48	5 22	5 62
6 16	6 28	6 40	6 48	6 56	6 14	6 27	6 54	6 28	6 63
7 17	7 29	7 41	7 50	7 56	7 15	7 31	7 57	7 34	7 69
d = 119	d 191	d 232	d = 295	d 380	d 112	d 206	d 313	d 164	d 402
i Vi π_i	i Vi π_i	i Vi π_i	i Vi π_i	i Vi π_i	i Vi π_i	i Vi π_i	i Vi π_i	i Vi π_i	i Vi π_i
1 1 3	1 1 3	1 1 3	1 1 3	1 1 3	1 1 2	1 1 2	1 1 2	1 1 2	1 1 2
2 1.5 6	2 1.5 6	2 1.5 6	2 1.5 6	2 1.5 6	2 1.5 4.5	2 1.5 4.5	2 1.5 4.5	2 1.5 4.5	2 1.5 4.5
3 2 10	3 2 10	3 2 10	3 2 10	3 2 10	3 2 7.5	3 2 7.5	3 2 7.5	3 2 7.5	3 2 7.5

Experiment 11 DATASET DS1	Experiment 12 DATASET DS2	Experiment 13 DATASET DS3	Experiment 14 DATASET DS4	Experiment 15 DATASET DS5	Experiment 16 DATASET DS6	Experiment 17 DATASET DS7	Experiment 18 DATASET DS8	Experiment 19 DATASET DS9	Experiment 20 DATASET DS10
J_i P_j	J_i P_j	J_i P_j	J_i P_j	J_i P_j	J_i P_j	J_i P_j	J_i P_j	J_i P_j	J_i P_j
1 6	1 12	1 17	1 23	1 19	1 5	1 11	1 24	1 12	1 38
2 8	2 14	2 23	2 30	2 49	2 7	2 18	2 34	2 13	2 40
3 9	3 15	3 25	3 36	3 50	3 9	3 19	3 36	3 16	3 51
4 11	4 19	4 31	4 42	4 52	4 11	4 22	4 40	4 19	4 59
5 14	5 24	5 35	5 46	5 54	5 13	5 26	5 48	5 22	5 62
6 16	6 28	6 40	6 48	6 56	6 14	6 27	6 54	6 28	6 63
7 17	7 29	7 41	7 50	7 56	7 15	7 31	7 57	7 34	7 69
d = 119	d 191	d 232	d = 295	d 380	d 112	d 206	d 313	d 164	d 402
i Vi π_i	i Vi π_i	i Vi π_i	i Vi π_i	i Vi π_i	i Vi π_i	i Vi π_i	i Vi π_i	i Vi π_i	i Vi π_i
1 1 3	1 1 3	1 1 3	1 1 3	1 1 3	1 1 2	1 1 2	1 1 2	1 1 2	1 1 2
2 1.5 6	2 1.5 6	2 1.5 6	2 1.5 6	2 1.5 6	2 1.5 4.5	2 1.5 4.5	2 1.5 4.5	2 1.5 4.5	2 1.5 4.5
3 2 10	3 2 10	3 2 10	3 2 10	3 2 10	3 2 7.5	3 2 7.5	3 2 7.5	3 2 7.5	3 2 7.5
4 2.5 15	4 2.5 15	4 2.5 15	4 2.5 15	4 2.5 15	4 2.5 11	4 2.5 11	4 2.5 11	4 2.5 11	4 2.5 11

Experiment 21 DATASET DS1	Experiment 22 DATASET DS2	Experiment 23 DATASET DS3	Experiment 24 DATASET DS4	Experiment 25 DATASET DS5	Experiment 26 DATASET DS6	Experiment 27 DATASET DS7	Experiment 28 DATASET DS8	Experiment 29 DATASET DS9	Experiment 30 DATASET DS10
J_i P_j	J_i P_j	J_i P_j	J_i P_j	J_i P_j	J_i P_j	J_i P_j	J_i P_j	J_i P_j	J_i P_j
1 6	1 12	1 17	1 23	1 19	1 5	1 11	1 24	1 12	1 38
2 8	2 14	2 23	2 30	2 49	2 7	2 18	2 34	2 13	2 40
3 9	3 15	3 25	3 36	3 50	3 9	3 19	3 36	3 16	3 51
4 11	4 19	4 31	4 42	4 52	4 11	4 22	4 40	4 19	4 59
5 14	5 24	5 35	5 46	5 54	5 13	5 26	5 48	5 22	5 62
6 16	6 28	6 40	6 48	6 56	6 14	6 27	6 54	6 28	6 63
7 17	7 29	7 41	7 50	7 56	7 15	7 31	7 57	7 34	7 69
d = 119	d 191	d 232	d = 295	d 380	d 112	d 206	d 313	d 164	d 402
i Vi π_i	i Vi π_i	i Vi π_i	i Vi π_i	i Vi π_i	i Vi π_i	i Vi π_i	i Vi π_i	i Vi π_i	i Vi π_i
1 1 3	1 1 3	1 1 3	1 1 3	1 1 3	1 1 2	1 1 2	1 1 2	1 1 2	1 1 2
2 1.5 6	2 1.5 6	2 1.5 6	2 1.5 6	2 1.5 6	2 1.5 4.5	2 1.5 4.5	2 1.5 4.5	2 1.5 4.5	2 1.5 4.5
3 2 10	3 2 10	3 2 10	3 2 10	3 2 10	3 2 7.5	3 2 7.5	3 2 7.5	3 2 7.5	3 2 7.5
4 2.5 15	4 2.5 15	4 2.5 15	4 2.5 15	4 2.5 15	4 2.5 11	4 2.5 11	4 2.5 11	4 2.5 11	4 2.5 11
5 3 21	5 3 21	5 3 21	5 3 21	5 3 21	5 3 15	5 3 15	5 3 15	5 3 15	5 3 15

Experiment 31 DATASET DS11	Experiment 32 DATASET DS12	Experiment 33 DATASET DS13	Experiment 34 DATASET DS14	Experiment 35 DATASET DS15	Experiment 36 DATASET DS16	Experiment 37 DATASET DS17	Experiment 38 DATASET DS18	Experiment 39 DATASET DS19	Experiment 40 DATASET DS20
J_i P_j	J_i P_j	J_i P_j	J_i P_j	J_i P_j	J_i P_j	J_i P_j	J_i P_j	J_i P_j	J_i P_j
1 6	1 12	1 17	1 23	1 19	1 5	1 11	1 24	1 12	1 38
2 8	2 14	2 23	2 30	2 49	2 7	2 18	2 34	2 13	2 40
3 9	3 15	3 25	3 36	3 50	3 9	3 19	3 36	3 16	3 51
4 11	4 19	4 31	4 42	4 52	4 11	4 22	4 40	4 19	4 59
5 14	5 24	5 35	5 46	5 54	5 13	5 26	5 48	5 22	5 62
6 16	6 28	6 40	6 48	6 56	6 14	6 27	6 54	6 28	6 63
7 17	7 29	7 41	7 50	7 56	7 15	7 31	7 57	7 34	7 69
8 18	8 30	8 47	8 74	8 83	8 18	8 32	8 59	8 70	8 70
d = 119	d 191	d 279	d = 369	d 463	d 112	d 206	d 372	d 234	d 472
i Vi π_i	i Vi π_i	i Vi π_i	i Vi π_i	i Vi π_i	i Vi π_i	i Vi π_i	i Vi π_i	i Vi π_i	i Vi π_i
1 1 3	1 1 3	1 1 3	1 1 3	1 1 3	1 1 2	1 1 2	1 1 2	1 1 2	1 1 2
2 1.5 6	2 1.5 6	2 1.5 6	2 1.5 6	2 1.5 6	2 1.5 4.5	2 1.5 4.5	2 1.5 4.5	2 1.5 4.5	2 1.5 4.5
3 2 10	3 2 10	3 2 10	3 2 10	3 2 10	3 2 7.5	3 2 7.5	3 2 7.5	3 2 7.5	3 2 7.5

Experiment 41	Experiment 42	Experiment 43	Experiment 44	Experiment 45	Experiment 46	Experiment 47	Experiment 48	Experiment 49	Experiment 50
DATASET DS11	DATASET DS12	DATASET DS13	DATASET DS14	DATASET DS15	DATASET DS16	DATASET DS17	DATASET DS18	DATASET DS19	DATASET DS20
J_i P_j	J_i P_j	J_i P_j	J_i P_j	J_i P_j	J_i P_j	J_i P_j	J_i P_j	J_i P_j	J_i P_j
1 6	1 12	1 17	1 23	1 19	1 5	1 11	1 24	1 12	1 38
2 8	2 14	2 23	2 30	2 49	2 7	2 18	2 34	2 13	2 40
3 9	3 15	3 25	3 36	3 50	3 9	3 19	3 36	3 16	3 51
4 11	4 19	4 31	4 42	4 52	4 11	4 22	4 40	4 19	4 59
5 14	5 24	5 35	5 46	5 54	5 13	5 26	5 48	5 22	5 62
6 16	6 28	6 40	6 48	6 56	6 14	6 27	6 54	6 28	6 63
7 17	7 29	7 41	7 50	7 56	7 15	7 31	7 57	7 34	7 69
8 18	8 30	8 47	8 74	8 83	8 18	8 32	8 59	8 70	8 70
d = 119	d 191	d 279	d = 369	d 463	d 112	d 206	d 372	d 234	d 472
i Vi π_i	i Vi π_i	i Vi π_i	i Vi π_i	i Vi π_i	i Vi π_i	i Vi π_i	i Vi π_i	i Vi π_i	i Vi π_i
1 1 3	1 1 3	1 1 3	1 1 3	1 1 3	1 1 2	1 1 2	1 1 2	1 1 2	1 1 2
2 1.5 6	2 1.5 6	2 1.5 6	2 1.5 6	2 1.5 6	2 1.5 4.5	2 1.5 4.5	2 1.5 4.5	2 1.5 4.5	2 1.5 4.5
3 2 10	3 2 10	3 2 10	3 2 10	3 2 10	3 2 7.5	3 2 7.5	3 2 7.5	3 2 7.5	3 2 7.5
4 2.5 15	4 2.5 15	4 2.5 15	4 2.5 15	4 2.5 15	4 2.5 11	4 2.5 11	4 2.5 11	4 2.5 11	4 2.5 11
Experiment 51	Experiment 52	Experiment 53	Experiment 54	Experiment 55	Experiment 56	Experiment 57	Experiment 58	Experiment 59	Experiment 60
DATASET DS11	DATASET DS12	DATASET DS13	DATASET DS14	DATASET DS15	DATASET DS16	DATASET DS17	DATASET DS18	DATASET DS19	DATASET DS20
J_i P_j	J_i P_j	J_i P_j	J_i P_j	J_i P_j	J_i P_j	J_i P_j	J_i P_j	J_i P_j	J_i P_j
1 6	1 12	1 17	1 23	1 19	1 5	1 11	1 24	1 12	1 38
2 8	2 14	2 23	2 30	2 49	2 7	2 18	2 34	2 13	2 40
3 9	3 15	3 25	3 36	3 50	3 9	3 19	3 36	3 16	3 51
4 11	4 19	4 31	4 42	4 52	4 11	4 22	4 40	4 19	4 59
5 14	5 24	5 35	5 46	5 54	5 13	5 26	5 48	5 22	5 62
6 16	6 28	6 40	6 48	6 56	6 14	6 27	6 54	6 28	6 63
7 17	7 29	7 41	7 50	7 56	7 15	7 31	7 57	7 34	7 69
8 18	8 30	8 47	8 74	8 83	8 18	8 32	8 59	8 70	8 70
d = 119	d 191	d 279	d = 369	d 463	d 112	d 206	d 372	d 234	d 472
i Vi π_i	i Vi π_i	i Vi π_i	i Vi π_i	i Vi π_i	i Vi π_i	i Vi π_i	i Vi π_i	i Vi π_i	i Vi π_i
1 1 3	1 1 3	1 1 3	1 1 3	1 1 3	1 1 2	1 1 2	1 1 2	1 1 2	1 1 2
2 1.5 6	2 1.5 6	2 1.5 6	2 1.5 6	2 1.5 6	2 1.5 4.5	2 1.5 4.5	2 1.5 4.5	2 1.5 4.5	2 1.5 4.5
3 2 10	3 2 10	3 2 10	3 2 10	3 2 10	3 2 7.5	3 2 7.5	3 2 7.5	3 2 7.5	3 2 7.5
4 2.5 15	4 2.5 15	4 2.5 15	4 2.5 15	4 2.5 15	4 2.5 11	4 2.5 11	4 2.5 11	4 2.5 11	4 2.5 11
5 3 21	5 3 21	5 3 21	5 3 21	5 3 21	5 3 15	5 3 15	5 3 15	5 3 15	5 3 15
Experiment 61	Experiment 62	Experiment 63	Experiment 64	Experiment 65	Experiment 66	Experiment 67	Experiment 68	Experiment 69	Experiment 70
DATASET DS21	DATASET DS22	DATASET DS23	DATASET DS24	DATASET DS25	DATASET DS26	DATASET DS27	DATASET DS28	DATASET DS29	DATASET DS30
J_i P_j	J_i P_j	J_i P_j	J_i P_j	J_i P_j	J_i P_j	J_i P_j	J_i P_j	J_i P_j	J_i P_j
1 6	1 12	1 17	1 23	1 19	1 5	1 11	1 24	1 12	1 38
2 8	2 14	2 23	2 30	2 49	2 7	2 18	2 34	2 13	2 40
3 9	3 15	3 25	3 36	3 50	3 9	3 19	3 36	3 16	3 51
4 11	4 19	4 31	4 42	4 52	4 11	4 22	4 40	4 19	4 59
5 14	5 24	5 35	5 46	5 54	5 13	5 26	5 48	5 22	5 62
6 16	6 28	6 40	6 48	6 56	6 14	6 27	6 54	6 28	6 63
7 17	7 29	7 41	7 50	7 56	7 15	7 31	7 57	7 34	7 69
8 18	8 30	8 47	8 74	8 83	8 18	8 32	8 59	8 70	8 70
9 19	9 38	9 52	9 76	9 91	9 20	9 35	9 60	9 79	9 100
d = 135	d 230	d 331	d = 445	d 530	d 132	d 241	d 432	d 313	d 572
i Vi π_i	i Vi π_i	i Vi π_i	i Vi π_i	i Vi π_i	i Vi π_i	i Vi π_i	i Vi π_i	i Vi π_i	i Vi π_i
1 1 3	1 1 3	1 1 3	1 1 3	1 1 3	1 1 2	1 1 2	1 1 2	1 1 2	1 1 2
2 1.5 6	2 1.5 6	2 1.5 6	2 1.5 6	2 1.5 6	2 1.5 4.5	2 1.5 4.5	2 1.5 4.5	2 1.5 4.5	2 1.5 4.5
3 2 10	3 2 10	3 2 10	3 2 10	3 2 10	3 2 7.5	3 2 7.5	3 2 7.5	3 2 7.5	3 2 7.5
Experiment 71	Experiment 72	Experiment 73	Experiment 74	Experiment 75	Experiment 76	Experiment 77	Experiment 78	Experiment 79	Experiment 80
DATASET DS21	DATASET DS22	DATASET DS23	DATASET DS24	DATASET DS25	DATASET DS26	DATASET DS27	DATASET DS28	DATASET DS29	DATASET DS30
J_i P_j	J_i P_j	J_i P_j	J_i P_j	J_i P_j	J_i P_j	J_i P_j	J_i P_j	J_i P_j	J_i P_j
1 6	1 12	1 17	1 23	1 19	1 5	1 11	1 24	1 12	1 38
2 8	2 14	2 23	2 30	2 49	2 7	2 18	2 34	2 13	2 40
3 9	3 15	3 25	3 36	3 50	3 9	3 19	3 36	3 16	3 51
4 11	4 19	4 31	4 42	4 52	4 11	4 22	4 40	4 19	4 59
5 14	5 24	5 35	5 46	5 54	5 13	5 26	5 48	5 22	5 62
6 16	6 28	6 40	6 48	6 56	6 14	6 27	6 54	6 28	6 63
7 17	7 29	7 41	7 50	7 56	7 15	7 31	7 57	7 34	7 69
8 18	8 30	8 47	8 74	8 83	8 18	8 32	8 59	8 70	8 70
9 19	9 38	9 52	9 76	9 91	9 20	9 35	9 60	9 79	9 100
d = 135	d 230	d 331	d = 445	d 530	d 132	d 241	d 432	d 313	d 572
i Vi π_i	i Vi π_i	i Vi π_i	i Vi π_i	i Vi π_i	i Vi π_i	i Vi π_i	i Vi π_i	i Vi π_i	i Vi π_i
1 1 3	1 1 3	1 1 3	1 1 3	1 1 3	1 1 2	1 1 2	1 1 2	1 1 2	1 1 2
2 1.5 6	2 1.5 6	2 1.5 6	2 1.5 6	2 1.5 6	2 1.5 4.5	2 1.5 4.5	2 1.5 4.5	2 1.5 4.5	2 1.5 4.5
3 2 10	3 2 10	3 2 10	3 2 10	3 2 10	3 2 7.5	3 2 7.5	3 2 7.5	3 2 7.5	3 2 7.5
4 2.5 15	4 2.5 15	4 2.5 15	4 2.5 15	4 2.5 15	4 2.5 11	4 2.5 11	4 2.5 11	4 2.5 11	4 2.5 11

Experiment 81 DATASET DS21	Experiment 82 DATASET DS22	Experiment 83 DATASET DS23	Experiment 84 DATASET DS24	Experiment 85 DATASET DS25	Experiment 86 DATASET DS26	Experiment 87 DATASET DS27	Experiment 88 DATASET DS28	Experiment 89 DATASET DS29	Experiment 90 DATASET DS30
J_i P_j	J_i P_j	J_i P_j	J_i P_j	J_i P_j	J_i P_j	J_i P_j	J_i P_j	J_i P_j	J_i P_j
1 6	1 12	1 17	1 23	1 19	1 5	1 11	1 24	1 12	1 38
2 8	2 14	2 23	2 30	2 49	2 7	2 18	2 34	2 13	2 40
3 9	3 15	3 25	3 36	3 50	3 9	3 19	3 36	3 16	3 51
4 11	4 19	4 31	4 42	4 52	4 11	4 22	4 40	4 19	4 59
5 14	5 24	5 35	5 46	5 54	5 13	5 26	5 48	5 22	5 62
6 16	6 28	6 40	6 48	6 56	6 14	6 27	6 54	6 28	6 63
7 17	7 29	7 41	7 50	7 56	7 15	7 31	7 57	7 34	7 69
8 18	8 30	8 47	8 74	8 83	8 18	8 32	8 59	8 70	8 70
9 19	9 38	9 52	9 76	9 91	9 20	9 35	9 60	9 79	9 100
d = 135	d 230	d 331	d = 445	d 530	d 132	d 241	d 432	d 313	d 572
i Vi π_i	i Vi π_i	i Vi π_i	i Vi π_i	i Vi π_i	i Vi π_i	i Vi π_i	i Vi π_i	i Vi π_i	i Vi π_i
1 1 3	1 1 3	1 1 3	1 1 3	1 1 3	1 1 2	1 1 2	1 1 2	1 1 2	1 1 2
2 1.5 6	2 1.5 6	2 1.5 6	2 1.5 6	2 1.5 6	2 1.5 4.5	2 1.5 4.5	2 1.5 4.5	2 1.5 4.5	2 1.5 4.5
3 2 10	3 2 10	3 2 10	3 2 10	3 2 10	3 2 7.5	3 2 7.5	3 2 7.5	3 2 7.5	3 2 7.5
4 2.5 15	4 2.5 15	4 2.5 15	4 2.5 15	4 2.5 15	4 2.5 11	4 2.5 11	4 2.5 11	4 2.5 11	4 2.5 11
5 3 21	5 3 21	5 3 21	5 3 21	5 3 21	5 3 15	5 3 15	5 3 15	5 3 15	5 3 15
Experiment 91 DATASET DS31	Experiment 92 DATASET DS32	Experiment 93 DATASET DS33	Experiment 94 DATASET DS34	Experiment 95 DATASET DS35	Experiment 96 DATASET DS36	Experiment 97 DATASET DS37	Experiment 98 DATASET DS38	Experiment 99 DATASET DS39	Experiment 100 DATASET DS40
J_i P_j	J_i P_j	J_i P_j	J_i P_j	J_i P_j	J_i P_j	J_i P_j	J_i P_j	J_i P_j	J_i P_j
1 4	1 4	1 6	1 2	1 20	1 13	1 14	1 10	1 12	1 13
2 5	2 5	2 12	2 5	2 20	2 16	2 17	2 14	2 15	2 14
3 7	3 6	3 14	3 10	3 21	3 28	3 17	3 15	3 17	3 25
4 8	4 7	4 17	4 12	4 23	4 36	4 20	4 24	4 23	4 30
5 10	5 8	5 21	5 24	5 25	5 37	5 22	5 28	5 24	5 33
6 11	6 9	6 22	6 28	6 28	6 39	6 27	6 30	6 31	6 40
7 12	7 10	7 23	7 36	7 28	7 43	7 31	7 35	7 40	7 44
8 15	8 11	8 24	8 42	8 36	8 55	8 32	8 40	8 42	8 46
9 18	9 12	9 26	9 50	9 36	9 59	9 45	9 51	9 42	9 49
10 20	10 13	10 38	10 60	10 37	10 60	10 46	10 56	10 45	10 54
d = 80	d 80	d 140	d = 160	d 160	d 220	d 180	d 196	d 190	d 210
i Vi π_i	i Vi π_i	i Vi π_i	i Vi π_i	i Vi π_i	i Vi π_i	i Vi π_i	i Vi π_i	i Vi π_i	i Vi π_i
1 1 3	1 1 3	1 1 3	1 1 3	1 1 3	1 1 2	1 1 2	1 1 2	1 1 2	1 1 2
2 1.5 6	2 1.5 6	2 1.5 6	2 1.5 6	2 1.5 6	2 1.5 4.5	2 1.5 4.5	2 1.5 4.5	2 1.5 4.5	2 1.5 4.5
3 2 13	3 2 10	3 2 11	3 2 11	3 2 10	3 2 7.5	3 2 7.5	3 2 7.5	3 2 7.5	3 2 7.5
Experiment 101 DATASET DS31	Experiment 102 DATASET DS32	Experiment 103 DATASET DS33	Experiment 104 DATASET DS34	Experiment 105 DATASET DS35	Experiment 106 DATASET DS36	Experiment 107 DATASET DS37	Experiment 108 DATASET DS38	Experiment 109 DATASET DS39	Experiment 110 DATASET DS40
J_i P_j	J_i P_j	J_i P_j	J_i P_j	J_i P_j	J_i P_j	J_i P_j	J_i P_j	J_i P_j	J_i P_j
1 4	1 4	1 6	1 2	1 20	1 13	1 14	1 10	1 12	1 13
2 5	2 5	2 12	2 5	2 20	2 16	2 17	2 14	2 15	2 14
3 7	3 6	3 14	3 10	3 21	3 28	3 17	3 15	3 17	3 25
4 8	4 7	4 17	4 12	4 23	4 36	4 20	4 24	4 23	4 30
5 10	5 8	5 21	5 24	5 25	5 37	5 22	5 28	5 24	5 33
6 11	6 9	6 22	6 28	6 28	6 39	6 27	6 30	6 31	6 40
7 12	7 10	7 23	7 36	7 28	7 43	7 31	7 35	7 40	7 44
8 15	8 11	8 24	8 42	8 36	8 55	8 32	8 40	8 42	8 46
9 18	9 12	9 26	9 50	9 36	9 59	9 45	9 51	9 42	9 49
10 20	10 13	10 38	10 60	10 37	10 60	10 46	10 56	10 45	10 54
d = 80	d 80	d 140	d = 160	d 160	d 220	d 180	d 196	d 190	d 210
i Vi π_i	i Vi π_i	i Vi π_i	i Vi π_i	i Vi π_i	i Vi π_i	i Vi π_i	i Vi π_i	i Vi π_i	i Vi π_i
1 1 3	1 1 3	1 1 3	1 1 3	1 1 3	1 1 2	1 1 2	1 1 2	1 1 2	1 1 2
2 1.5 6	2 1.5 6	2 1.5 6	2 1.5 6	2 1.5 6	2 1.5 4.5	2 1.5 4.5	2 1.5 4.5	2 1.5 4.5	2 1.5 4.5
3 2 10	3 2 10	3 2 10	3 2 10	3 2 10	3 2 7.5	3 2 7.5	3 2 7.5	3 2 7.5	3 2 7.5
4 2.5 15	4 2.5 15	4 2.5 15	4 2.5 15	4 2.5 15	4 2.5 11	4 2.5 11	4 2.5 11	4 2.5 11	4 2.5 11
Experiment 111 DATASET DS31	Experiment 112 DATASET DS32	Experiment 113 DATASET DS33	Experiment 114 DATASET DS34	Experiment 115 DATASET DS35	Experiment 116 DATASET DS36	Experiment 117 DATASET DS37	Experiment 118 DATASET DS38	Experiment 119 DATASET DS39	Experiment 120 DATASET DS40
J_i P_j	J_i P_j	J_i P_j	J_i P_j	J_i P_j	J_i P_j	J_i P_j	J_i P_j	J_i P_j	J_i P_j
1 4	1 4	1 6	1 2	1 20	1 13	1 14	1 10	1 12	1 13
2 5	2 5	2 12	2 5	2 20	2 16	2 17	2 14	2 15	2 14
3 7	3 6	3 14	3 10	3 21	3 28	3 17	3 15	3 17	3 25
4 8	4 7	4 17	4 12	4 23	4 36	4 20	4 24	4 23	4 30
5 10	5 8	5 21	5 24	5 25	5 37	5 22	5 28	5 24	5 33
6 11	6 9	6 22	6 28	6 28	6 39	6 27	6 30	6 31	6 40
7 12	7 10	7 23	7 36	7 28	7 43	7 31	7 35	7 40	7 44
8 15	8 11	8 24	8 42	8 36	8 55	8 32	8 40	8 42	8 46
9 18	9 12	9 26	9 50	9 36	9 59	9 45	9 51	9 42	9 49
10 20	10 13	10 38	10 60	10 37	10 60	10 46	10 56	10 45	10 54
d = 80	d 80	d 140	d = 160	d 160	d 220	d 180	d 196	d 190	d 210
i Vi π_i	i Vi π_i	i Vi π_i	i Vi π_i	i Vi π_i	i Vi π_i	i Vi π_i	i Vi π_i	i Vi π_i	i Vi π_i
1 1 3	1 1 3	1 1 3	1 1 3	1 1 3	1 1 2	1 1 2	1 1 2	1 1 2	1 1 2
2 1.5 6	2 1.5 6	2 1.5 6	2 1.5 6	2 1.5 6	2 1.5 4.5	2 1.5 4.5	2 1.5 4.5	2 1.5 4.5	2 1.5 4.5
3 2 10	3 2 10	3 2 10	3 2 10	3 2 10	3 2 7.5	3 2 7.5	3 2 7.5	3 2 7.5	3 2 7.5
4 2.5 15	4 2.5 15	4 2.5 15	4 2.5 15	4 2.5 15	4 2.5 11	4 2.5 11	4 2.5 11	4 2.5 11	4 2.5 11
5 3 21	5 3 21	5 3 21	5 3 21	5 3 21	5 3 15	5 3 15	5 3 15	5 3 15	5 3 15

References

- [1] J. Black, "Design rules for implementing the Toyota Production System," *Int. J. Prod. Res.*, vol. 45, no. 16, pp. 3639–3664, 2007.
- [2] J. Kilpatrick, "Lean Principles," *Manufacturing Extension Partnership*. Utah, pp. 1–5, 2003.
- [3] C. Low, R. K. Li, and G. H. Wu, "Minimizing Total Earliness and Tardiness for Common Due Date Single-Machine Scheduling with an Unavailability Interval," *Mathematical Problems in Engineering*, vol. 2016. Hindawi Publishing Corporation, 2016.
- [4] K. R. Baker and D. Trietsch, "Principles of Sequencing and Scheduling," in *Principles of Sequencing and Scheduling*, Hoboken, NJ: A John Wiley & Sons, Inc., 2009, pp. 86–105.
- [5] G. Mouzon and M. B. Yildirim, "A framework to minimise total energy consumption and total tardiness on a single machine," *Int. J. Sustain. Eng.*, vol. 1, no. 2, pp. 105–116, 2008.
- [6] K. Fang, N. Uhan, F. Zhao, and J. W. Sutherland, "A new approach to scheduling in manufacturing for power consumption and carbon footprint reduction," *J. Manuf. Syst.*, vol. 30, no. 4, pp. 234–240, 2011.
- [7] H. Zhang, F. Zhao, K. Fang, and J. W. Sutherland, "Energy-conscious flow shop scheduling under time-of-use electricity tariffs," *CIRP Ann. - Manuf. Technol.*, vol. 63, no. 1, pp. 37–40, 2014.
- [8] X. Wu and Y. Sun, "A green scheduling algorithm for flexible job shop with energy-saving measures," *J. Clean. Prod.*, vol. 172, pp. 3249–3264, 2018.
- [9] J. Y. Moon, K. Shin, and J. Park, "Optimization of production scheduling with time-dependent and machine-dependent electricity cost for industrial energy efficiency," *Int. J. Adv. Manuf. Technol.*, vol. 68, no. 1–4, pp. 523–535, 2013.
- [10] H. Mokhtari and A. Hasani, "An energy-efficient multi-objective optimization for flexible job-shop scheduling problem," *Comput. Chem. Eng.*, vol. 104, pp. 339–352, 2017.
- [11] A. Zandi, R. Ramezani, and L. Monplaisir, "Green parallel machines scheduling problem: A bi-objective model and a heuristic algorithm to obtain Pareto frontier," *J. Oper. Res. Soc.*, vol. 71, no. 6, pp. 967–978, 2020.
- [12] J. Rocholl and L. Monch, "Hybrid Algorithms for The Earliness-Tardiness Single-Machine Multiple Orders per Job Scheduling Problem with A Common Due Date," *RAIRO-Oper. Res.*, vol. 52, no. 4–5, pp. 1329–1350, 2018.

- [13] M. Shahriari, N. Shoja, A. E. Zade, S. Barak, and M. Sharifi, "JIT single machine scheduling problem with periodic preventive maintenance," *J. Ind. Eng. Int.*, vol. 12, no. 3, pp. 299–310, 2016.
- [14] A. Corominas, A. García-Villoria, N. A. González, and R. Pastor, "A multistage graph-based procedure for solving a just-in-time flexible job-shop scheduling problem with machine and time-dependent processing costs," *J. Oper. Res. Soc.*, vol. 70, no. 4, pp. 620–633, 2019.
- [15] H. Allaoua and B. Brahim, "New Properties for Solving the Single-Machine Scheduling Problem with Early/Tardy Jobs," *J. Intell. Syst.*, vol. 26, no. 3, pp. 531–543, 2017.
- [16] S. Radhakrishnan and J. A. Ventura, "Simulated annealing for parallel machine scheduling with earliness-tardiness penalties and sequence-dependent set-up times," *Int. J. Prod. Res.*, vol. 38, no. 10, pp. 2233–2252, 2000.
- [17] M. Khader, "A Solution Approach for The Single Machine Early/Tardy Scheduling Problems," King Fahd University of Petroleum and Minerals, 2008.
- [18] G. Rabadi, G. C. Anagnostopoulos, and M. Mollaghasemi, "A heuristic algorithm for the just-in-time single machine scheduling problem with setups: A comparison with simulated annealing," *Int. J. Adv. Manuf. Technol.*, vol. 32, no. 3–4, pp. 326–335, 2007.
- [19] J. M. S. Valente, J. F. Gonçalves, and R. A. F. S. Alves, "A hybrid genetic algorithm for the early/tardy scheduling problem," *Asia-Pacific J. Oper. Res.*, vol. 23, no. 3, pp. 393–405, 2006.
- [20] S. Hepdogan, R. Moraga, G. W. DePuy, and G. E. Whitehouse, "A metaheuristic for the early/tardy single machine scheduling problem," *2006 IIE Annu. Conf. Exhib.*, 2006.
- [21] D. P. Ronconi and E. G. Birgin, "Mixed-integer programming models for flowshop scheduling problems minimizing the total earliness and tardiness," *Springer Optim. Its Appl.*, vol. 60, pp. 91–105, 2012.
- [22] Y. Li, Y. He, Y. Wang, F. Tao, and J. W. Sutherland, "An optimization method for energy-conscious production in flexible machining job shops with dynamic job arrivals and machine breakdowns," *J. Clean. Prod.*, vol. 254, p. 120009, 2020.
- [23] R. Zhang and R. Chiong, "Solving the energy-efficient job shop scheduling problem: A multi-objective genetic algorithm with enhanced local search for minimizing the total weighted tardiness and total energy consumption," *Journal of Cleaner Production*, vol. 112, pp. 3361–3375, 2016.
- [24] I. Das and J. E. Dennis, "NORMAL-BOUNDARY INTERSECTION: A NEW METHOD FOR GENERATING THE PARETO SURFACE IN NONLINEAR MULTICRITERIA OPTIMIZATION PROBLEMS," *SIAM J. Optim.*, vol. 8, no. 3,

pp. 631–657, 1998.

- [25] T. C. E. Cheng, C. Oğuz, and X. D. Qi, “Due-date assignment and single machine scheduling with compressible processing times,” *Int. J. Prod. Econ.*, vol. 43, no. 1, pp. 29–35, 1996.
- [26] R. G. Vickson, “Two Single Machine Sequencing Problems Involving Controllable Job Processing Times,” *A I I E Trans.*, vol. 12, no. 3, pp. 58–262, 1980.
- [27] S. S. Panwalkar and R. Rajagopalan, “Single-machine sequencing with controllable processing times,” *Eur. J. Oper. Res.*, vol. 59, no. 2, pp. 298–302, 1992.
- [28] Z. L. Chen, Q. Lu, and G. Tang, “Single machine scheduling with discretely controllable processing times,” *Oper. Res. Lett.*, vol. 21, no. 2, pp. 69–76, 1997.
- [29] D. Shabtay and G. Steiner, “A survey of scheduling with controllable processing times,” *Discret. Appl. Math.*, vol. 155, no. 13, pp. 1643–1666, 2007.
- [30] F. Yao, A. Demers, and S. Shenker, “A scheduling model for reduced CPU energy,” *Proc. IEEE 36th Annu. Found. Comput. Sci.*, pp. 374–382, 1995.
- [31] W. C. Kwon and T. Kim, “Optimal voltage allocation techniques for dynamically variable voltage processors,” *ACM Trans. Embed. Comput. Syst.*, vol. 4, no. 1, pp. 211–230, 2005.
- [32] N. Bansal, T. Kimbrel, and K. Pruhs, “Speed scaling to manage energy and temperature,” *J. ACM*, vol. 54, no. 1, 2007.
- [33] K. Fang, N. A. Uhan, F. Zhao, and J. W. Sutherland, “Flow shop scheduling with peak power consumption constraints,” *Ann. Oper. Res.*, vol. 206, no. 1, pp. 115–145, 2013.
- [34] G. Mavrotas, “Effective implementation of the ϵ -constraint method in Multi-Objective Mathematical Programming problems,” *Appl. Math. Comput.*, vol. 213, no. 2, pp. 455–465, 2009.
- [35] R. E. Steuer, *Multiple criteria optimization: Theory, computation and application*. New York: John Wiley & Sons, 1986.
- [36] J. L. Cohon, *Multiobjective programming and planning*. New York: Dover Publications, Inc., 1978.

Vitae

Name :Hudaifah Abdullah Syawal |

Nationality :Indonesia |

Date of Birth :10/20/1991|

Email :hudaifahsyawal@gmail.com|

Address :Bawukan RT013/004, Kalikebo, Trucuk, Klaten, Jawa
Tengah, 57467, Indonesia. |

Academic Background :Hudaifah earned a Bachelor of Engineering in Industrial
Engineering from Sepuluh Nopember Institute of Technology (ITS), Surabaya,
Indonesia. He Earned Master of Science in Industrial & System engineering form
King Fahd Univeristy of Petroleum & Minerals (KFUPM), Dhahran, Saudi Arabia. |

Two papers of Hudaifah:

- [1] H. A. Syawal and U. M. Al-Turki, “Batching and resource allocation under stochastic unreliable flexible flow shop,” *Proc. Int. Conf. Ind. Eng. Oper. Manag.*, vol. 0, no. March, pp. 3034–3040, 2020..
- [2] H. A. Syawal and H. K. Alfares, “Inventory Optimization for Multiple Perishable Products with Dynamic Pricing, Dependent Stochastic Demand, and Dynamic Reorder Policy,” *2020 Ind. Syst. Eng. Conf. ISEC 2020*, 2020.