YAŞAR UNIVERSITY

GRADUATE SCHOOL

PHD THESIS

MULTI-OBJECTIVE GREEN HYBRID FLOWSHOP SCHEDULING PROBLEMS

HANDE ÖZTOP

THESIS ADVISOR: PROF. DR. LEVENT KANDİLLER

CO-ADVISOR: PROF. DR. MEHMET FATİH TAŞGETİREN

INDUSTRIAL ENGINEERING

PRESENTATION DATE: 15.06.2020

BORNOVA / İZMİR

JUNE 2020

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

Jury Members:	**Signature:**
Prof. Dr. Levent KANDİLLER Yaşar University	[illegible]
Prof. Dr. Mustafa Arslan ÖRNEK Yaşar University	[illegible]
Prof. Dr. Bilge BİLGEN Dokuz Eylül University	B Bilgen
Prof. Dr. Mehmet Cemali DİNÇER Yaşar University	[illegible]
Prof. Dr. Ceyda OĞUZ Koç University	Ceyda

[illegible]

Prof. Dr. Cüneyt GÜZELİŞ
Director of the Graduate School

ABSTRACT

MULTI-OBJECTIVE GREEN HYBRID FLOWSHOP SCHEDULING PROBLEMS

Öztop, Hande
Ph.D., Industrial Engineering
Advisor: Prof. Dr. Levent KANDİLLER
Co-Advisor: Prof. Dr. Mehmet Fatih TAŞGETİREN
June 2020

The hybrid flowshop scheduling problem (HFSP) has been extensively studied in the literature with various production-efficiency related objectives. Nevertheless, studies that consider energy consumption and environmental impacts have rather been limited for the HFSP in the literature. This thesis addresses the trade-off between makespan and total energy consumption objectives in hybrid flowshops, where machines can operate at varying speed levels. In this thesis, new bi-objective mixed-integer linear programming (MILP) and bi-objective constraint programming (CP) models are proposed for the energy-efficient HFSP employing a speed scaling method, where both job-based and job-machine (matrix)-based versions of the speed scaling are considered. Since the objectives of minimizing makespan and total energy consumption are contradicting with each other, the augmented ε-constraint method is employed for obtaining the Pareto-optimal solutions. While close approximations for the Pareto-optimal frontier are obtained for small instances, sets of non-dominated solutions are found for large instances by solving the proposed MILP and CP models under a time-limit. Since the studied problem is NP-hard, new bi-objective metaheuristic algorithms are also proposed for both job-based and matrix-based versions of the energy-efficient HFSP as well as a constructive heuristic. Namely, two variants of the iterated greedy algorithm, a variable block insertion heuristic and four variants of an ensemble of metaheuristic algorithms are proposed for the job-based version of the problem. Furthermore, two variants of the iterated greedy algorithm, a variable block insertion heuristic and an ensemble of metaheuristic algorithms are proposed for the matrix-based version of the problem. This thesis also presents two

new heuristic fitness calculation approaches for the HFSP. The performances of the proposed bi-objective metaheuristics are compared with each other as well as the MILP and CP solutions on a well-known HFSP benchmark set in terms of cardinality, diversity and closeness of the solutions. Initially, the performance of the metaheuristics is tested on small instances with regard to the Pareto-optimal solutions. Subsequently, it is shown that the proposed metaheuristics are very effective for solving large instances in terms of both solution quality and computational time.

Key Words: hybrid flowshop scheduling, energy-efficient scheduling, multi-objective optimization, metaheuristics, mixed-integer linear programming, constraint programming

ÖZ

ÇOK-AMAÇLI ENERJİ-VERİMLİ HİBRİD AKIŞ TİPİ ÇİZELGELEME PROBLEMLERİ

Öztop, Hande
Doktora Tezi, Endüstri Mühendisliği
Danışman: Prof. Dr. Levent KANDİLLER
Yardımcı Danışman: Prof. Dr. Mehmet Fatih TAŞGETİREN
Haziran 2020

Literatürde, hibrid akış tipi çizelgeleme problemi çeşitli üretim verimliliği bazlı amaç fonksiyonları düşünülerek yaygın bir şekilde çalışılmıştır. Ancak, hibrid akış tipi çizelgeleme problemi için enerji tüketimi ve çevresel etkileri dikkate alan çalışmalar literatürde oldukça azdır. Bu tez, makinelerin değişen hız seviyelerinde çalışabildiği hibrid akış tipi atölyelerindeki, maksimum tamamlanma zamanı ve toplam enerji tüketimi amaç fonksiyonları arasındaki çelişkiyi ele almaktadır. Bu tezde, enerji-verimli hibrid akış tipi çizelgeleme problemi için, hız ölçeklendirme yöntemi kullanılarak, özgün iki-amaçlı karma-tamsayılı doğrusal programlama ve iki-amaçlı kısıt programlama model formülasyonları önerilmiştir. Bu tezde, hız ölçeklendirme yönteminin hem iş-bazlı hem de iş-tezgah (matris)-bazlı versiyonları çalışılmıştır. Maksimum tamamlanma zamanını ve toplam enerji tüketimini minimize etme amaç fonksiyonları birbirleriyle çeliştiklerinden dolayı, Pareto-optimal çözümleri elde etmek için genişletilmiş epsilon kısıt yöntemi kullanılmıştır. Küçük örnekler için Pareto-optimal eğriye oldukça yakın yaklaşımlar elde edilirken, büyük örnekler için ise önerilen karma-tamsayılı doğrusal programlama ve kısıt programlama model formülasyonları belirli bir süre limiti altında çözülerek baskın olmayan çözüm kümeleri elde edilmiştir. Ayrıca, çalışılan problemin NP-zor sınıfına ait bir problem olmasından dolayı, enerji-verimli hibrid akış tipi çizelgeleme probleminin hem iş-bazlı hem de matris-bazlı versiyonları için özgün iki-amaçlı metasezgisel algoritmalar özgün bir yapıcı sezgisel ile birlikte önerilmiştir. Problemin iş-bazlı versiyonu için iki tip yinelemeli açgözlü algoritma, bir değişken blok yerleştirme sezgiseli ve dört tip bütünleşik-metasezgisel algoritmalar önerilmiştir. Ayrıca, problemin matris-bazlı

versiyonu için iki tip yinelemeli açgözlü algoritma, bir değişken blok yerleştirme sezgiseli ve bir bütünleşik-metasezgisel algoritma önerilmiştir. Bunların yanı sıra, bu tez, hibrid akış tipi çizelgeleme problemi için iki özgün sezgisel amaç fonksiyonu değeri hesaplama yöntemi de önermektedir. Literatürde oldukça bilinen hibrid akış tipi çizelgeleme problemi örnekleri kullanılarak, önerilen iki-amaçlı metasezgisellerin performansları birbirleriyle ve karma-tamsayılı doğrusal programlama ve kısıt programlama model formülasyonlarının çözümleri ile; çözümlerin sayısallığı, çeşitliliği ve yakınlığı açılarından kıyaslanmıştır. Öncelikle, metasezgisellerin performansı küçük örnekler üzerinde Pareto-optimal çözümler ile kıyaslanarak test edilmiştir. Ardından, önerilen metasezgisellerin büyük örnekleri çözmek adına hem çözüm kalitesi hem de çözüm süresi açısından oldukça etkin olduğu gösterilmiştir.

Anahtar Kelimeler: hibrid akış tipi çizelgeleme, enerji-verimli çizelgeleme, çok-amaçlı optimizasyon, metasezgiseller, karma-tamsayılı doğrusal programlama, kısıt programlama

ACKNOWLEDGEMENTS

I would like to express my gratitude to my thesis advisors Prof. Dr. Levent Kandiller and Prof. Dr. Mehmet Fatih Taşgetiren, for their guidance and motivation during this thesis study. Firstly, I would like to express my sincere appreciation to my thesis advisor Prof. Dr. Levent Kandiller, who has an important influence on my intellectual development for his guidance, support and immense knowledge. His valuable guidance significantly helped me in all the time of my graduate studies. Secondly, I would like to express my gratitude to my thesis co-advisor Prof. Dr. Mehmet Fatih Taşgetiren for his valuable guidance during this thesis study. His guidance and immense knowledge helped me to improve my skills as a researcher.

I would also like to thank Prof. Dr. Deniz Türsel Eliiyi for her guidance during the early phases of this thesis. Additionally, I would like to express my gratitude to jury members Prof. Dr. Mehmet Cemali Dinçer, Prof. Dr. Mustafa Arslan Örnek, Prof. Dr. Ceyda Oğuz and Prof. Dr. Bilge Bilgen for their insightful suggestions, valuable comments and contributions.

Furthermore, and foremost, I would like to thank my family. They were always there, supporting me and encouraging me with their best wishes. I am very grateful to them for their patience, unfailing support and continuous encouragement.

Finally, the Department of Industrial Engineering at Yasar University was like my second home for more than six years. I would like to appreciate faculty, colleagues and particularly my office mates whose encouragement is always with me.

Hande Öztop
İzmir, 2020

TEXT OF OATH

I declare and honestly confirm that my study, titled "MULTI-OBJECTIVE GREEN HYBRID FLOWSHOP SCHEDULING PROBLEMS" and presented as a Ph.D. Thesis, has been written without applying to any assistance inconsistent with scientific ethics and traditions. I declare, to the best of my knowledge and belief, that all content and ideas drawn directly or indirectly from external sources are indicated in the text and listed in the list of references.

Hande Öztop

Signature

........................[illegible].....

July 3, 2020

TABLE OF CONTENTS

ABSTRACT v

ÖZ ix

ACKNOWLEDGEMENTS xiii

TEXT OF OATH xv

TABLE OF CONTENTS xvii

LIST OF FIGURES xxiii

LIST OF TABLES xxvii

SYMBOLS AND ABBREVIATIONS xxxi

CHAPTER 1 INTRODUCTION 1

CHAPTER 2 THE HYBRID FLOWSHOP SCHEDULING PROBLEM 5

CHAPTER 3 LITERATURE REVIEW & MOTIVATION OF THE THESIS 9
- 3.1 Literature Review 9
- 3.2 Motivation of the Thesis 18

CHAPTER 4 PROBLEM DEFINITION & MODEL FORMULATIONS 23
- 4.1 Mixed-Integer Linear Programming Models for EHFSP-V1 and EHFSP-V2 25
- 4.2 Constraint Programming Models for EHFSP-V1 and EHFSP-V2 26
  - 4.2.1 Constraint Programming 26
  - 4.2.2 CP Models for EHFSP-V1 and EHFSP-V2 28
- 4.3 Conflicting Objectives 30

CHAPTER 5 MULTI-OBJECTIVE OPTIMIZATION 33
- 5.1 Terminology 33
- 5.2 Solution Methods 34
- 5.3 Weighting Method 36
- 5.4 ε-Constraint and Augmented ε-Constraint Methods 36

CHAPTER 6 METAHEURISTIC ALGORITHMS 41
- 6.1 Solution Representation & Fitness Value Calculation 41
  - 6.1.1 Standard Forward Scheduling Approach 41

6.1.2 Heuristic Fitness Calculation Approaches 42
6.1.3 Solution Representation & Fitness Value Calculation for the EHFSP-V1 46
6.1.4 Solution Representation & Fitness Value Calculation for the EHFSP-V2 47
6.2 Constructive Heuristic & Single-Objective Algorithms for the HFSP with Makespan Criterion.. 47
6.2.1 Constructive Heuristic for the HFSP with Makespan Criterion........................... 48
6.2.2 Single-Objective Algorithms with Makespan Criterion 49
6.3 Energy-efficient Bi-Objective Metaheuristic Algorithms for the EHFSP-V1 53
6.3.1 Initial Population.. 53
6.3.2 E_IG and E_IG$_{\mathrm{ALL}}$ Algorithms 54
6.3.3 E_VBIH Algorithm 55
6.3.4 Ensemble of Metaheuristic Algorithms $(\mathrm{E\_EM}, \mathrm{E\_EM}_{\mathrm{HFR}}, \mathrm{E\_EM}_{\mathrm{HFN}}, \mathrm{E\_EM}_{\mathrm{HFRN}})$ 56
6.4 Energy-efficient Bi-Objective Metaheuristic Algorithms for the EHFSP-V2 58
6.4.1 Initial Population.. 58
6.4.2 E_IG2 and E_IG2$_{\mathrm{ALL}}$ Algorithms... 59
6.4.3 E_VBIH2 Algorithm 61
6.4.4. Ensemble of Metaheuristic Algorithms (E_EM2) 61
6.4.5. Heuristic Fitness Calculation & Mutation Operators.. 62
6.5 Archive Set Update Procedure 63
CHAPTER 7 COMPUTATIONAL RESULTS.. 67
7.1 Parameter Calibration of the Algorithms 70
7.1.1 Parameter Calibration of the IG, IG$_{\mathrm{ALL}}$ and VBIH Procedures............................ 70
7.1.2 Parameter Calibration of the HFN Approach... 74
7.2 Comparison of Constructive Heuristics based on C_{max} Criterion 78
7.3 Small Instances for the EHFSP-V1 82
7.4 Medium & Large Instances for the EHFSP-V1... 88
7.5 Small Instances for the EHFSP-V2 100
7.6 Medium & Large Instances for the EHFSP-V2.. 101
CHAPTER 8 CONCLUSION... 113

REFERENCES 121

LIST OF FIGURES

Figure 2. 1. Hybrid Flowshop Layout 6

Figure 2. 2. A Gantt Chart for a HFSP Example 7

Figure 4. 1. Pareto Frontier of a Small Problem (EHFSP-V1) 30

Figure 4. 2. Pareto Frontier of a Small Problem (EHFSP-V2) 31

Figure 5. 1. ε-Constraint Method 38

Figure 5. 2. Augmented ε-Constraint Method 39

Figure 5. 3. General Outline of the Augmented ε-Constraint Method 40

Figure 6. 1. Gantt Chart with $C_{max} = 14$ 42

Figure 6. 2. Gantt Chart with $C_{max} = 13$ 43

Figure 6. 3. Heuristic Fitness Calculation with Random Swap Moves 44

Figure 6. 4. Heuristic Fitness Calculation with Neighbor Swap Moves 45

Figure 6. 5. Solution Representation for the EHFSP-V1 46

Figure 6. 6. Solution Representation for the EHFSP-V2 47

Figure 6. 7. NEH Heuristic 48

Figure 6. 8. NEH_M(x) Heuristic 49

Figure 6. 9. IG Algorithm 50

Figure 6. 10. First-Improvement Insertion Neighborhood Structure 51

Figure 6. 11. IG$_{ALL}$ Algorithm 52

Figure 6. 12. Variable Block Insertion Algorithm 53

Figure 6. 13. Energy-Efficient First-Improvement Insertion Neighborhood Structure for the EHFSP-V1 55

Figure 6. 14. General Outline of the Energy-Efficient Ensemble of Metaheuristic Algorithms for the EHFSP-V1 58

Figure 6. 15. Energy-Efficient First-Improvement Insertion Neighborhood Structure for the EHFSP-V2 60

Figure 6. 16. General Outline of the Energy-Efficient Ensemble of Metaheuristic Algorithms for the EHFSP-V2 62

Figure 6. 17. Archive Set Update Procedure.. 65

Figure 7. 1. ANOVA Results for Parameters of the IG 71

Figure 7. 2. Main Effects Plot for Parameters of the IG 72

Figure 7. 3. ANOVA Results for Parameters of the IG_{ALL} 72

Figure 7. 4. Main Effects Plot for Parameters of the IG_{ALL} 73

Figure 7. 5. ANOVA Results for Parameters of the VBIH.................................... 74

Figure 7. 6. Main Effects Plot for Parameters of the VBIH................................... 74

Figure 7. 7. ANOVA Results for Parameters of the E_IG2.................................... 76

Figure 7. 8. Main Effects Plot for Parameters of the E_IG2................................... 76

Figure 7. 9. ANOVA Results for Parameters of the E_IG2_{ALL} 77

Figure 7. 10. Main Effects Plot for Parameters of the E_IG2_{ALL} 77

Figure 7. 11. ANOVA Results for Parameters of the E_VBIH2............................ 78

Figure 7. 12. Main Effects Plot for Parameters of the E_VBIH2........................... 78

Figure 7. 13. Comparison of Algorithms for an Instance with 5 Jobs 83

Figure 7. 14. Comparison of Algorithms for an Instance with 15 Jobs 93

LIST OF TABLES

Table 2. 1. Common Objective Functions for the HFSP 8

Table 3. 1. Shop Setting and Optimization Criteria Notation 15

Table 3. 2. Literature Review (Other Energy Saving Strategies) 16

Table 3. 3. Literature Review (Speed Scaling Strategy) 17

Table 3. 4. Literature Review for the EHFSP with Speed Scaling Strategy 20

Table 4. 1. Problem Notation 24

Table 7. 1. Comparison of Constructive Heuristics 79

Table 7. 2. Comparison of Single-Objective Algorithms 81

Table 7. 3. Performance Comparison of Algorithms on Small Instances (Set 1) for the EHFSP-V1 84

Table 7. 4. Performance Comparison of Algorithms on Small Instances (Set 2) for the EHFSP-V1 86

Table 7. 5. Performance Comparison of Algorithms on Medium Instances with 10 Jobs for the EHFSP-V1 90

Table 7. 6. *p*-values of Wilcoxon Signed-Rank Tests for Medium Instances with 10 Jobs for the EHFSP-V1 92

Table 7. 7. Performance Comparison of Algorithms on Medium Instances with 15 Jobs for the EHFSP-V1 94

Table 7. 8. *p*-values of Wilcoxon Signed-Rank Tests for Medium Instances with 15 Jobs for the EHFSP-V1 96

Table 7. 9. Performance Comparison of Algorithms on Large Instances for the EHFSP-V1 98

Table 7. 10. *p*-values of Wilcoxon Signed-Rank Tests for Large Instances for the EHFSP-V1 100

Table 7. 11. Performance Comparison of Algorithms on Small Instances for the EHFSP-V2 101

Table 7. 12. Performance Comparison of Algorithms on Medium Instances with 10 Jobs for the EHFSP-V2 103

Table 7. 13. Results of Statistical Tests for Medium Instances with 10 Jobs for the EHFSP-V2 105

Table 7. 14. Performance Comparison of Algorithms on Medium Instances with 15 Jobs for the EHFSP-V2 107

Table 7. 15. Results of Statistical Tests for Medium Instances with 15 Jobs for the EHFSP-V2 109

Table 7. 16. Performance Comparison of Algorithms on Large Instances for the EHFSP-V2 110

Table 7. 17. Results of Statistical Tests for Large Instances for the EHFSP-V2 112

SYMBOLS AND ABBREVIATIONS

AS	Archive Set
C_{max}	Maximum Completion Time (Makespan)
CH	Constructive Heuristic
CMFOA	Collaborative Multi-Objective Fruit Fly Optimization Algorithm
CP	Constraint Programming
C_p	Ratio of the Pareto-Optimal Solutions Found
DC	Destruction-Construction
DS	Distribution Spacing
EHFSP	Energy-Efficient Hybrid Flowshop Scheduling Problem
EHFSP-V1	Energy-Efficient Hybrid Flowshop Scheduling Problem with Job-Based Speed Scaling Strategy
EHFSP-V2	Energy-Efficient Hybrid Flowshop Scheduling Problem with Matrix-Based Speed Scaling Strategy
EM	Ensemble of Metaheuristic Algorithms
E_EM	Energy-Efficient Ensemble of Metaheuristic Algorithms
E_EM2	Energy-Efficient Ensemble of Metaheuristic Algorithms for the EHFSP-V2
E_EM_{HFR}	Energy-Efficient Ensemble of Metaheuristic Algorithms with HFR
E_EM_{HFN}	Energy-Efficient Ensemble of Metaheuristic Algorithms with HFN
$\text{E\_EM}_{\text{HFRN}}$	Energy-Efficient Ensemble of Metaheuristic Algorithms with HFR and HFN
E_IG	Energy-Efficient Iterated Greedy
E_IG2	Energy-Efficient Iterated Greedy for the EHFSP-V2
E_IG_{ALL}	Energy-Efficient Iterated Greedy with Additional Local Search
$\text{E\_IG2}_{\text{ALL}}$	Energy-Efficient Iterated Greedy with Additional Local Search for the EHFSP-V2

E_VBIH	Energy-Efficient Variable Block Insertion Heuristic
E_VBIH2	Energy-Efficient Variable Block Insertion Heuristic for the EHFSP-V2
FFSP	Flexible Flowshop Scheduling Problem
GA	Genetic Algorithm
GRASP	Greedy Randomized Adaptive Search Procedure
HFN	Heuristic Fitness Calculation with Neighbor Swap Moves
HFR	Heuristic Fitness Calculation with Random Swap Moves
HFSP	Hybrid Flowshop Scheduling Problem
IG	Iterated Greedy
IG_{ALL}	Iterated Greedy with Additional Local Search
IGD	Inverted Generational Distance
MDABC	Multi-Objective Discrete Artificial Bee Colony Algorithm
MIP	Mixed-Integer Programming
MILP	Mixed-Integer Linear Programming
MINLP	Mixed-Integer Nonlinear Programming
MOBIH	Multi-Objective Block Insertion Heuristic
MOEA	Multi-Objective Evolutionary Algorithm
MOEA/D	Multi-Objective Evolutionary Algorithm based on Decomposition
MOGA	Multi-Objective Genetic Algorithm
MOIG	Multi-Objective Iterated Greedy
MONEH	Multi-Objective NEH Algorithm
MOP	Multi-Objective Optimization Problem
MOVBIH	Multi-Objective Variable Block Insertion Heuristic
MOVILS	Multi-Objective Variable Iterated Local Search

NEH_M(x)	Modified NEH Heuristic with x Solutions
NSGA-II	Non-dominated Sorting Genetic Algorithm II
PFSP	Permutation Flowshop Scheduling Problem
PS	Population Size
PSO	Particle Swarm Optimization
RPD	Relative Percentage Deviation
TEC	Total Energy Consumption
TLBO	Teaching-Learning-Based Optimization
TOU	Time-of-Use
VBIH	Variable Block Insertion Heuristic

CHAPTER 1
INTRODUCTION

Recently, green manufacturing with energy consumption consideration has gained attention due to the scarce energy resources and a series of environmental effects. It is commonly known that the rising amount of greenhouse gas emissions (CO_2) caused by fossil fuel consumption initiates environmental pollution and global warming. Since energy is commonly generated through fossil fuels, effective usage of energy will provide a considerable reduction in carbon dioxide emissions and slow down the rapid exhaustion of fuel resources.

According to Fang et al. (2011), the energy consumption of the industrial sector is almost 50% of the world’s total energy consumption. In the USA, the manufacturing companies are responsible for approximately one-third of the energy consumption and contributes to almost 28% of greenhouse gas emissions (Mouzon and Yildirim, 2008). In Germany, the manufacturing enterprises consume approximately 47% of the total national electricity usage, and the resultant amount of carbon dioxide emissions caused by this electricity is 18–20% (Dai et al., 2013).

Due to increasing concerns related to environmental deterioration, the manufacturing sector can be faced with additional taxes and regulations related to carbon footprints. As manufacturing companies are responsible for the high energy consumption and related carbon emissions, they are faced with pressure to reduce their energy consumption (Fang et al., 2011; Mouzon and Yildirim, 2008). Consequently, manufacturing enterprises have made attempts to develop energy efficient approaches to reduce their energy consumption and carbon emissions.

One approach for minimizing energy consumption in manufacturing systems is to install energy-efficient machines. However, the significant financial investment needed makes it almost impracticable for most of the manufacturing sector, particularly for small-sized companies. Instead, the current practice is to operate the existing machinery by taking their energy consumption into consideration.

In this thesis, energy efficiency is studied from an operational planning perspective for the hybrid flowshops, which arise in various manufacturing environments including electronics (Wittrock, 1988; Liu and Chang, 2000; Jin et al., 2002), textile (Grabowski and Pempera, 2000), steel (Pan et al., 2013) and paper (Sherali et al., 1990) industries. Two comprehensive reviews on the hybrid flowshop scheduling problem (HFSP) can be found in (Ruiz and Vazquez Rodriguez, 2010; Ribas et al., 2010). Due to its practical relevance, the HFSP has been widely studied in the literature with the objectives related to production efficiency. However, studies regarding energy efficiency and environmental effects have been minimal.

The HFSP can be considered as a generalization of two classical scheduling problems: the parallel machine scheduling problem and the flowshop scheduling problem. In the HFSP, n jobs must be processed in a series of $m(m > 1)$ stages, optimizing a given objective function. All jobs must be sequentially processed following the same production order: stage 1, stage 2,…, stage m. Each job requires a nonnegative processing time in stage k, where each stage k has $|I_k| \geq 1$ identical parallel machines, and in at least one of the stages $|I_k| > 1$. Then, each job must be processed by one of the machines in each stage. Since the HFSP considers both assignment and scheduling of the jobs in each stage, the HFSP is harder to solve than the standard flowshop scheduling problem. The HFSP has already been proven as NP-hard (Gupta, 1988) for a hybrid flowshop with only two stages, where there is a single machine in one of the stages.

This thesis addresses the trade-off between the makespan (C_{max}) and the total energy consumption (TEC) in a hybrid flowshop environment. Note that, makespan, well known as maximum completion time, is the main performance criterion for increasing the utilization of resources and obtaining a high throughput. However, the TEC criterion is also critical to decrease fuel consumption and slow down environmental deterioration.

In this thesis, a speed scaling strategy is proposed for the energy-efficient hybrid flowshop scheduling problem (EHFSP), where the machines can operate at varying speed levels. In this strategy, the speed levels create a contradiction between the processing time and energy consumption, i.e., the energy consumption increases at higher speed levels, while the processing time decreases. Hence, the studied EHFSP in this thesis is a bi-objective optimization problem with two conflicting objectives of minimizing makespan and minimizing TEC.

In this thesis, two variants of the speed scaling strategy are studied for the EHFSP: a job-based speed scaling strategy (i.e., same speed level is employed for a job through all stages) and a matrix-based speed scaling strategy (i.e., speed of a job can vary from stages to stages). The EHFSP with job-based speed scaling strategy is denoted as EHFSP-V1 and the EHFSP with matrix-based speed scaling strategy is denoted as EHFSP-V2.

In this thesis, new bi-objective mixed-integer linear programming (MILP) and bi-objective constraint programming (CP) models are proposed for the EHFSP-V1 and EHFSP-V2. Benchmark instances are also developed by modifying the well-known HFSP benchmarks from the literature (Carlier and Neron, 2000; Liao et al., 2012; Öztop et al., 2019). For small instances, MILP and CP models are solved through the augmented ε-constraint method without a time limit to obtain the Pareto-optimal solutions. Since the problem is NP-hard (Gupta, 1988) and the solution time grows exponentially, the sets of non-dominated solutions are obtained with augmented ε-constraint method under a time limit for larger instances.

New bi-objective metaheuristic algorithms are also proposed for the EHFSP-V1 and EHFSP-V2. Namely, seven bi-objective metaheuristic algorithms are proposed for the EHFSP-V1, which are two variants of iterated greedy (IG) algorithm (Ruiz and Stützle, 2007), a variable block insertion heuristic (VBIH) and four variants of the ensemble of metaheuristic algorithms (EM). Additionally, four bi-objective metaheuristic algorithms are proposed for the EHFSP-V2, which are two variants of the IG algorithm, a VBIH algorithm and an ensemble of metaheuristic algorithms.

In this thesis, a new constructive heuristic is also presented for the HFSP with the makespan criterion by extending the NEH heuristic (Nawaz et al., 1983). Furthermore, two new heuristic fitness calculation approaches are proposed to compensate for the inefficiency of the standard forward scheduling approach for fitness function calculation in HFSP.

As mentioned in Chapter 3, the EHFSP with a speed scaling strategy is scarcely studied in the scheduling literature. Hence, this thesis contributes to the energy-efficient scheduling literature by applying the speed scaling strategy to the HFSP, presenting new bi-objective MILP and CP models for the EHFSP, developing original seven effective bi-objective metaheuristic algorithms for the EHFSP-V1 and developing original four effective bi-objective metaheuristic algorithms for the EHFSP-V2. To the best of our knowledge, this thesis presents a constraint programming approach to the EHFSP for the first time in the literature. This thesis also contributes to the hybrid flowshop scheduling literature by presenting a new constructive heuristic and two new heuristic fitness calculation approaches for the HFSP.

The remainder of this thesis is organized as follows. In Chapter 2, the basic HFSP and its common variants are explained. In Chapter 3, a comprehensive literature review is provided and the motivation of this thesis is given. Chapter 4 formally defines the EHFSP-V1 and EHFSP-V2, and presents the proposed bi-objective MILP and CP models. Chapter 5 explains the commonly used multi-objective optimization techniques as well as the related terminology. The augmented ε-constraint method is also explained in Chapter 5.

Chapter 6 presents the proposed heuristic fitness calculation approaches and bi-objective metaheuristic algorithms for both EHFSP-V1 and EHFSP-V2, as well as the constructive heuristic and the single-objective algorithms for the initial solution generation. In Chapter 7, computational results are provided to evaluate the performance of the proposed solution approaches. Finally, Chapter 8 addresses the concluding remarks and future research directions.

CHAPTER 2
THE HYBRID FLOWSHOP SCHEDULING PROBLEM

In a shop scheduling problem, there is a set of n jobs where each job has m operations corresponding to m machines. Processing times of the operations are assumed to be known in advance. At any time, each machine can process at most one operation, and each job can be processed on at most one machine. A job is said to be completed if all its operations have been completed. There are three basic shop scheduling problems due to the scheduling restrictions of operations:

- **Flowshop (*Fm*):** There are m machines in series. A set of n jobs must be processed on these machines following the same order, i.e., each job has to be processed first on machine 1, then on machine 2, and so on. If there is a restriction that each machine must also process the jobs in the same order, the machine environment is named as *permutation flowshop*.
- **Job Shop (*Jm*):** It is a generalization of flowshop in which each job has its own pre-specified order to follow.
- **Open shop (*Om*):** There are no restrictions on the order of each job through the machine environment.

The hybrid flowshop is a generalization of the flowshop and the parallel machine environments. Hybrid flowshop has also been referred to as a flexible flowshop, multi-processor flowshop, or flowshop with parallel machines in the literature. Instead of m machines in series, there are m ($m > 1$) stages in series where each stage consists of at least one machine in parallel, and at least one of these stages has more than one machine. All jobs must flow through every stage in the same order, i.e., each job has to be processed first at stage 1, then at stage 2, and so on. A hybrid flowshop layout is shown in Figure 2.1.

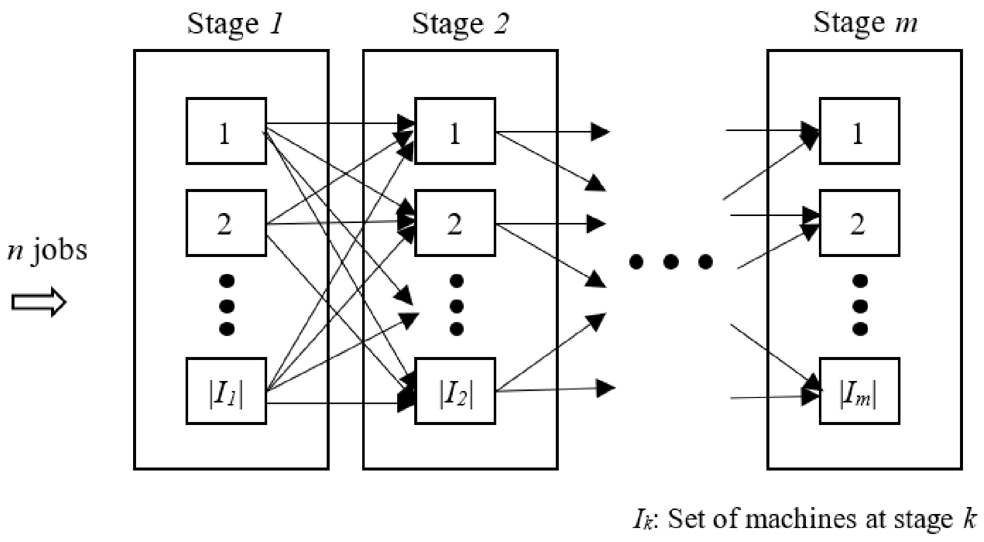


Figure 2. 1. Hybrid Flowshop Layout

In the basic form of the hybrid flowshop scheduling problem (HFSP), there are following assumptions: machines are identical in each stage; all jobs and machines are available at time zero; a job is processed by only one machine at each stage; setup times can be ignored; job preemption is not allowed; there is infinite intermediate storage between stages, and problem data is deterministic. In most environments, the HFSP is NP-hard, since the HFSP with only two stages is known to be NP-hard (Gupta, 1988).

In order to provide a better understanding, a feasible Gantt chart is illustrated in Figure 2.2 for a basic HFSP example that has 3 stages and 4 jobs. Both stages 1 and 3 have two identical parallel machines, while stage 2 has one machine. The processing times of the jobs are given as follows, where p_{kj} is the processing time of job j at stage k: $p_{kj} = \begin{bmatrix} 3 & 4 & 3 & 2 \\ 5 & 2 & 4 & 4 \\ 3 & 4 & 3 & 5 \end{bmatrix}$. As shown in Figure 2.2, the makespan (C_{max}) value of this example is 20, which is the maximum completion time of all jobs.

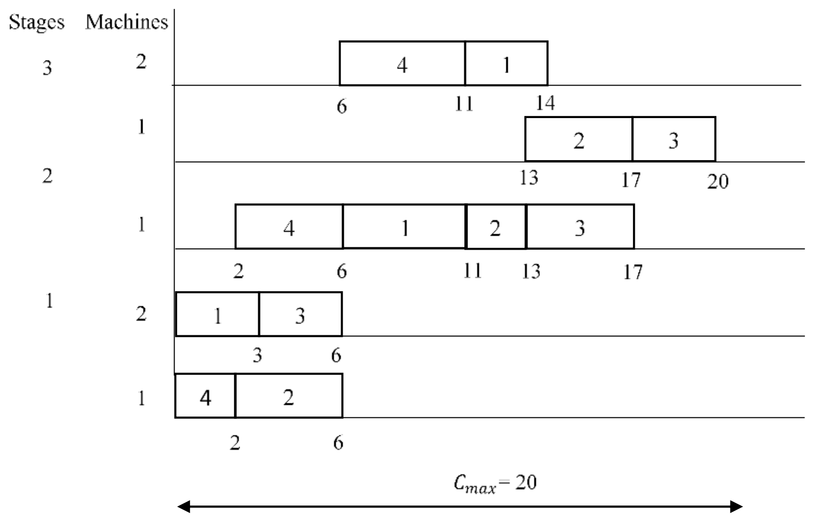


Figure 2. 2. A Gantt Chart for an HFSP Example

Different HFSP variants can be described by modifying assumptions, objectives and/or constraints of the basic problem. The notation proposed in Vignier et al. (1999) is generally used to identify them, which follows the three-field notation (α|β|γ) proposed by Graham et al. (1979). The α field describes the shop configuration, and it includes four parameters $\alpha_1\alpha_2, \left(\alpha_3\alpha_4{}^{(k)}\right)$. α_1 indicates the general structure of the shop, α_2 is the number of stages in the shop, α_3 and α_4 represent the characteristics of the machines in each stage k. Particularly, α_3 defines the machine type and α_4 indicates the number of machines in the stage (Ruiz and Vazquez Rodriguez, 2010). For instance, *FHm*, $((PM^{(k)})_{k=1}^{m})$ denotes a hybrid flowshop (*FH*) with *m* stages, where there is any number of identical parallel machines in each stage.

There are three basic machine types in shop scheduling: identical (P), uniform (Q) and unrelated (R). In the HFSP with identical machines, all machines within each stage are identical. Therefore, the processing time of a job in each stage does not vary from machine to machine. In the case of uniform machines, machines have different speeds, where each machine i has a speed v_i, and job j requires p_{kj} / v_i time units when it is assigned to machine i of stage k. On the other hand, in the HFSP with unrelated machines, each machine i has a speed v_{ij} for each job j, and job j requires p_{kj} / v_{ij} time units when it is assigned to machine i of stage k. (Pinedo, 2002).

The β field describes the constraints and assumptions. The most common among these constraints are listed below (Ruiz and Vazquez Rodriguez, 2010):

- Release dates (r_j): job j cannot start to be processed before its release time r_j.
- Sequence-dependent setup times (ST_{sd}): there exist sequence-dependent setup times between jobs.
- Precedence constraints (*prec*): there are precedence relations between jobs.
- No-wait (*no-wait*): jobs are not permitted to wait between two consecutive stages.
- Blocking (*block*): there are limited buffers between consecutive stages. Therefore, jobs can wait in the previous stage until an adequate area is released.
- Machine eligibility restrictions (M_j): job j can be processed by only a subset of machines M_j at each stage.
- Preemption (*prmp*): job preemptions are allowed.

Finally, the γ field contains the objective function. The most common among these objectives are listed in Table 2.1 (Ruiz and Vazquez Rodriguez, 2010). The necessary notation is explained accordingly. Let C_j be the completion time of job j in the last stage. $F_j = C_j - r_j$ is the flow time of job j, which is the time job j spends in the system. The lateness of job j is denoted by $L_j = C_j - d_j$, where d_j is the due date of job j. $T_j = max(L_j, 0)$ is the tardiness and $E_j = max(d_j - C_j, 0)$ is the earliness of job j. The weight w_j is a priority factor for job j.

Table 2. 1. Common Objective Functions for the HFSP

Objective	**Description**
Makespan / Maximum completion time (C_{max})	$max_j\ C_j$
Total completion time	$\sum C_j$
Total weighted completion time	$\sum w_j C_j$
Total flow time	$\sum F_j$
Total weighted flow time	$\sum w_j F_j$
Total tardiness	$\sum T_j$
Total weighted tardiness	$\sum w_j T_j$
Total earliness	$\sum E_j$
Total weighted earliness	$\sum w_j E_j$

CHAPTER 3
LITERATURE REVIEW & MOTIVATION OF THE THESIS

3.1 Literature Review

The HFSP has been extensively studied in the literature considering different machine environments, constraints and objectives. Numerous exact algorithms, heuristics and metaheuristics have been proposed for the HFSP due to its complexity and practical relevance. Two comprehensive reviews on HFSP can be found in Ribas et al. (2010) and Ruiz and Vazquez Rodriguez (2010).

In the HFSP literature, most of the studies deal with a single objective related to production efficiency, where the most common among these objectives is to minimize the makespan, total/average completion time, flow time and tardiness. Relatively fewer studies consider several of these objectives together (Ruiz and Vazquez Rodriguez, 2010). For instance, Jungwattanakit et al. (2008, 2009) proposed heuristic algorithms to minimize the weighted sum of makespan and the number of tardy jobs in a hybrid flowshop environment. Behnamian and Fatemi (2011) proposed a metaheuristic algorithm for the HFSP with sequence-dependent setup times, which minimizes makespan and resource allocation costs.

Even though the objectives related to production efficiency have been widely studied in the scheduling literature, studies that regard energy-efficient scheduling have rather been limited. A recent review of the energy-efficient scheduling problems is provided by Gahm et al. (2016).

One of the most well-known studies is the work by Mouzon et al. (2007). The authors pointed out that a significant amount of energy can be saved by turning the machines off during idle times. They proposed several dispatching rules for scheduling jobs on a single CNC machine. These rules state that the machine can be shut down if the energy consumption for turning it on/off is less than the idle energy consumption. However, energy savings during machine operation are not considered. In a further study, Mouzon and Yildirim (2008) extended this strategy to the single machine scheduling problem with the objectives of TEC and total tardiness. Later, Dai et al. (2013) extended the turn-off strategy to the multi-objective flexible flowshop scheduling problem (FFSP) with unrelated parallel machines, considering two conflicting objectives of minimizing makespan and total energy consumption. Recently, Che et al. (2017) studied the single-machine scheduling problem with a power-down mechanism to minimize both total energy consumption and maximum tardiness. They proposed a mixed-integer programming (MIP) model and an ε-constraint method to obtain the Pareto frontier. They also developed a local search, a preprocessing technique and valid inequalities to strengthen the ε-constraint method.

Although the turn-off strategy can provide energy savings, it may not be applicable in certain shop floors where the machines cannot be turned off entirely during production. Moreover, frequent use of this strategy can significantly shorten the service life of some machines. Hence, other energy saving strategies have also been proposed in the literature.

From the energy consumption viewpoint, another direction is to consider time-of-use (TOU) electricity prices for scheduling problems. Under the TOU tariff system, electricity prices depend on the time of the day and can vary from hour to hour. Hence, the demand can be decreased during hours of high prices by shifting some operations to hours of lower prices. Luo et al. (2013) considered TOU electricity prices and proposed an ant colony algorithm for the HFSP with uniform machines that minimizes makespan and electric power cost. Moon et al. (2013) also proposed a genetic algorithm (GA) for minimizing the weighted sum of makespan and time dependent electricity costs in an unrelated parallel machine environment. Similarly, Shrouf et al. (2014) proposed a turn on/off strategy considering fluctuating energy prices in a day. They proposed a mathematical model and a GA to minimize energy consumption costs for a single machine scheduling problem.

Zhang et al. (2014) also consider TOU electricity tariffs for the flowshop scheduling problem by proposing a time-indexed integer programming formulation that minimizes electricity cost and the carbon footprint. Recently, Ding et al. (2016a) studied the unrelated parallel machine scheduling problem under a TOU pricing scheme, where the objective is to minimize the total electricity cost with a restriction on makespan. They proposed a time-interval-based MIP formulation and a Dantzig–Wolfe decomposition approach for the problem. More recently, Zhang et al. (2018)

proposed a greedy insertion heuristic for the energy efficient single machine scheduling problem under the TOU tariff system. Wang et al. (2018) also presented a MIP model, a constructive heuristic (CH) and an NSGA-II algorithm to solve the bi-objective identical parallel machine scheduling problem under TOU electricity prices.

As another approach to improve energy efficiency, Fang et al. (2011) proposed a speed scaling strategy for flowshop scheduling that minimizes peak power consumption, carbon footprint, and makespan. They proposed a multi-objective formulation in which operation speed is considered as an independent factor that can be changed to affect the peak load and energy consumption. In the speed scaling strategy, it is assumed that the machines can operate at multiple speed levels, in which the speed levels of the machines can be tuned for the operations of the jobs. In this strategy, speed levels create a contradiction between processing time and energy consumption, i.e., the energy consumption increases at higher speed level, while the processing time decreases. Thus, speed levels of the operations should be determined carefully to improve both production and energy efficiency. As the machines can operate at multiple speed levels in many real-life production environments, the speed-scaling strategy has been widely adopted for scheduling problems in the energy-efficient scheduling literature due to its practicability.

Fang et al. (2013) presented MIP formulations for the permutation flowshop scheduling problem (PFSP) with peak power consumption constraints considering both discrete and continuous processing speeds. Fang and Lin (2013) proposed an integer programming formulation and several heuristics for the parallel machine scheduling problem, where the processing speeds of the machines can be tuned during operation. Ding et al. (2016b) also considered the same strategy for the carbon-efficient PFSP and proposed a multi-objective NEH algorithm and a modified iterated greedy algorithm. Furthermore, Mansouri et al. (2016) considered variable speed levels for the two-machine sequence-dependent PFSP. They developed a multi-objective MIP model and lower bounds with the objectives of minimizing makespan and energy consumption. Later, Mansouri and Aktas (2016) extended the study of Mansouri et al. (2016) by developing a heuristic algorithm and multi-objective genetic algorithms (MOGA) for the same problem.

The energy-efficient PFSP with total flow time criterion was also studied by Öztop et al. (2018) employing a speed scaling framework. The authors proposed a multi-objective MILP model and a multi-objective IG (MOIG) algorithm to solve the problem, where they evaluated the performance of these solution methods only on small scale instances. Later, Öztop et al. (2020) extended the study of Öztop et al. (2018) by presenting two new variants of the MOIG, a multi-objective VBIH (MOVBIH) algorithm and a constructive heuristic for the same problem. In this study, extensive computational experiments were performed to test the performance of the MILP model, constructive heuristic and metaheuristics, employing both small and large instances.

The speed scaling approach was also employed for the energy-efficient job shop scheduling problems. Zhang and Chiong (2016) proposed a MOGA using a machine speed scaling framework in order to minimize the total weighted tardiness and total energy consumption in a job shop scheduling problem. Salido et al. (2016) also developed an energy-efficient genetic algorithm for the job shop scheduling problem and compared the performance of their genetic algorithm with a CP optimizer tool of a commercial solver.

The speed scaling approach was also implemented to the energy-efficient single machine scheduling problem with release dates and sequence-dependent setup times by Tasgetiren et al. (2018a). Later, the energy-efficient single machine scheduling total weighted tardiness problem with sequence-dependent setup times was also studied by Tasgetiren et al. (2018b), where the authors proposed a MILP model, a multi-objective block insertion heuristic (MOBIH) and a MOIG to solve the problem. Che et al. (2015) also presented two MIP models to solve speed-scalable energy-efficient single machine scheduling problems with bounded maximum tardiness.

Furthermore, Wu and Che (2019) presented a memetic differential evolution algorithm for the energy-efficient unrelated parallel machine scheduling problem employing the speed scaling strategy. Zheng and Wang (2018) also developed a collaborative multi-objective fruit fly optimization algorithm (CMFOA) for the energy-efficient unrelated parallel machine scheduling problem with resource constraints. Recently, Jiang and Wang (2019) proposed a mathematical model and a multi-objective evolutionary algorithm to solve the energy-efficient PFSP with sequence-dependent setup times. Additionally, Lu et al. (2017) considered the energy-efficient PFSP with sequence-dependent setup and controllable transportation time and proposed a hybrid multi-objective backtracking search algorithm.

The speed scaling approach was also employed for other variants of the energy-efficient scheduling problems. More recently, Tasgetiren et al. (2019) proposed a multi-objective MILP model, and a multi-objective variable iterated local search (MOVILS) algorithm and two variants of the MOGA for the energy-efficient no-idle flowshop scheduling problem employing a speed scaling strategy. Yin et al. (2017) also proposed a mathematical model and a MOGA for the flexible job-shop environment that optimizes makespan, energy efficiency and noise reduction. In their model, the machining spindle speed, which affects production time, power and noise, is treated as an independent decision variable. The speed scaling strategy was also employed for the distributed energy efficient flowshop scheduling problems (Jiang et al., 2017; Deng et al., 2016; Wang et al., 2017; Wang and Wang, 2018).

As seen in the above discussions, metaheuristics are widely employed for multi-objective optimization problems due to their complexities. In recent years, many multi-objective evolutionary algorithms (MOEA) have been developed, where the most well-known ones are NSGA-II (Deb et al., 2002) and multi-objective evolutionary algorithm based on decomposition (MOEA/D) (Zhang and Li, 2007). A comprehensive review of different types of MOEAs can also be found in Zhou et al. (2011).

Various approaches have been reported for the energy-efficient flexible or hybrid flowshop scheduling problems in the literature (Dai et al., 2013; Liu et al., 2008; Bruzonne et al., 2012; Luo et al., 2013; Tang et al., 2016; Yan et al., 2016; Li et al., 2018; Meng et al., 2019; Wu et al., 2018; Zeng et al., 2018; Zhang et al., 2019a; Liu and Huang, 2014; Chen et al., 2018; Zhang et al., 2019b; Lei et al., 2018; Li et al., 2019; Shen et al., 2017).

Dai et al. (2013) applied a turn-off strategy to the multi-objective FFSP by presenting an improved genetic-simulated annealing algorithm. Liu et al. (2008) proposed a mixed-integer nonlinear programming (MINLP) model for the HFSP that minimizes the energy consumption and limits the makespan. Bruzonne et al. (2012) proposed a MIP formulation for the FFSP that minimizes the weighted sum of total tardiness and the makespan. Luo et al. (2013) considered the time-of-use electricity prices and proposed an ant colony algorithm for the HFSP with uniform machines in which the price of electricity depends on the time of the day. Tang et al. (2016) proposed a particle swarm optimization (PSO) for the energy-efficient FFSP with unrelated parallel machines considering dynamic factors. A multi-level approach optimization (machine tool and shop floor levels) was proposed for the energy-efficient FFSP in Yan et al. (2016), which integrates power models of single machine and cutting parameters optimization into the energy-efficient scheduling problems. Recently, Li et al. (2018) presented an energy-aware multi-objective optimization algorithm for solving the HFSP with the objectives of makespan and energy consumption in the case of sequence-dependent setup times. In their energy consumption calculation, they considered three types of energy consumption, i.e., processing, standby and setup energy consumptions.

Furthermore, Meng et al. (2019) proposed an improved genetic algorithm for the energy-conscious HFSP with unrelated parallel machines employing a turn-off strategy. Wu et al. (2018) presented a MIP model and a hybrid NSGA-II with a variable local search to solve a multi-objective FFSP that considers variable processing time due to renewable energy. In their study, there are several types of capacitated power supply systems that lead to different processing times and different energy consumptions. Zeng et al. (2018) proposed a MIP model and a hybrid NSGA-II for the multi-objective FFSP with batch processing that minimizes makespan, electricity consumption and material waste. More recently, Zhang et al. (2019a) proposed a three-stage multi-objective approach based on decomposition for the energy-efficient HFSP with the consideration of machines with different energy usage ratios, sequence-dependent setups, and machine-to-machine transportation operations.

To the best of our knowledge, the speed scaling approach has been employed for the energy-efficient HFSP/FFSP in only a few studies (Liu and Huang, 2014; Chen et al., 2018; Zhang et al., 2019b; Lei et al., 2018; Li et al., 2019; Shen et al., 2017). The details of these studies and the differences of this thesis between the existing studies are explained in the following subsection (Section 3.2) as well as the motivation of this thesis.

Although the majority of studies on shop floor scheduling so far have not considered energy related criteria, the aforementioned attempts form a basis for a study on energy-efficient scheduling, especially from energy saving strategy and modeling viewpoints.

Finally, Table 3.1 describes the notation that has been used to define shop setting and optimization criteria of scheduling problems. Then, Tables 3.2 and 3.3 summarize the literature review for the energy efficient scheduling problems. Table 3.2 presents the literature review for the studies that employ other energy saving strategies (turn on/off strategy, TOU electricity prices, etc.) except the speed scaling strategy. On the other hand, Table 3.3 presents the literature review for the studies that employ speed scaling strategy. In these tables, the second and third columns present the shop (machine) setting and optimization criteria addressed in each paper. The fourth column represents the energy saving strategy. The objective structure is also described according to two categories (single/multi objective) in the fifth column. The category of single objective primarily represents the approaches that consider one of the optimization criteria as a constraint. The sixth column presents the proposed solution approaches.

Table 3. 1. Shop Setting and Optimization Criteria Notation

Shop Setting		**Optimization Criteria**	
Notation	**Description**	**Notation**	**Description**
1	Single machine	TEC	Total energy consumption
Pm	*m* parallel machines	$\sum C_j$	Total completion time
Qm	*m* uniform parallel machines	$\sum T_j$	Total tardiness
Rm	*m* unrelated parallel machines	$\sum w_j T_j$	Total weighted tardiness
Fm	Flowshop with *m* machines	$\sum F_j$	Total flow time
Jm	Job shop with *m* machines	C_{max}	Makespan
FFm	Flexible flowshop with *m* stages	P_{max}	Peak power consumption
FHm	Hybrid flowshop with *m* stages	G_{max}	Carbon footprint
FJm	Flexible job shop with *m* stages	T_{max}	Maximum tardiness
prmu	Permutation		
ST_{sd}	Sequence-dependent setup times		
prec	Precedence constraints		
r_j	Release dates		
no-idle	No-idle scheduling		
no-wait	No-wait scheduling		
dyn	Dynamic scheduling		
M_j	Machine eligibility		
batch	Batch processing		
dist	Distributed flowshop		

Table 3. 2. Literature Review (Other Energy Saving Strategies)

Reference	Shop Setting (Comments)	Criteria	Energy Saving Strategy	Objective Structure	Solution Approach
Bruzonne et al. (2012)	FFm	$C_{max}, \sum T_j, P_{max}$	Other	Objective: $C_{max} + \sum T_j$ (Constraint: P_{max})	MIP
Che et al. (2017)	$1(r_j)$	TEC and T_{max}	Turn on/off strategy	Multi-objective	MIP, Valid Inequalities, Cluster Analysis
Dai et al. (2013)	FFm (Rm)	TEC and C_{max}	Turn on/off strategy	Multi-objective	MIP, Genetic-simulated annealing algorithm
Ding et al. (2016a)	Rm	TEC and C_{max}	TOU electricity prices	Objective: TEC (Constraint: C_{max})	MILP, Dantzig– Wolfe Decomposition
Li et al. (2018)	FHm (ST_{sd})	TEC and C_{max}	Other	Multi-objective	Heuristics
Liu et al. (2008)	FHm	TEC and C_{max}	Other	Objective: TEC (Constraint: C_{max})	MINLP, improved GA
Luo et al. (2013)	FHm (Qm)	TEC and C_{max}	TOU electricity prices	Multi-objective	Ant colony optimization
Meng et al. (2019)	FHm (Rm)	TEC and C_{max}	Turn on/off strategy	Single objective	MIP, GA
Moon et al. (2013)	Rm	TEC and C_{max}	TOU electricity prices	Objective: TEC + C_{max}	GA
Mouzon et al. (2007)	1	TEC and $\sum C_j$	Turn on/off strategy	Multi-objective	MIP, Dispatching rules
Mouzon & Yildirim (2008)	1	TEC and $\sum T_j$	Turn on/off strategy	Multi-objective	MINLP, GRASP
Shrouf et al. (2014)	1	TEC	TOU electricity prices & turn on/off strategy	Single objective	MIP, GA
Tang et al. (2016)	FFm(Rm,dyn)	TEC and C_{max}	Other	Multi-objective	MIP, PSO
Wang et al. (2018)	Pm	TEC and C_{max}	TOU electricity prices	Multi-objective	MIP, CH, NSGA-II
Wu et al. (2018)	FFm	C_{max} and G_{max}	Other	Multi-objective	MIP, Hybrid NSGA-II
Yan et al. (2016)	FFm	TEC and C_{max}	Other	Multi-objective (weighted objectives)	Multi-level optimization, GA
Zeng et al. (2018)	FFm ($batch$)	TEC, C_{max} and Material Waste	Other	Multi-objective	MIP, Hybrid NSGA-II
Zhang et al. (2014)	Fm	TEC and G_{max}	TOU electricity prices	Multi-objective	Time-indexed integer programming
Zhang et al. (2018)	1	TEC	TOU electricity prices	Single-objective	MIP, Greedy Insertion Heuristic
Zhang et al. (2019a)	FHm (ST_{sd})	TEC and C_{max}	Other	Multi-objective	MILP, Decomposition based multi-objective approach

Table 3. 3. Literature Review (Speed Scaling Strategy)

Reference	Shop Setting (Comments)	Criteria	Energy Saving Strategy	Objective Structure	Solution Approach
Che et al. (2015)	$1(r_j)$	TEC and T_{max}	Speed scaling	Objective: TEC (Constraint: T_{max})	MIP
Chen et al. (2018)	*FHm* (*Rm*, ST_{sd}, r_j, M_j, *lot-streaming*)	TEC and C_{max}	Speed scaling	Multi-objective	MIP, MOGA
Deng et al. (2016)	*Fm* (*prmu,dist*)	C_{max} and G_{max}	Speed scaling	Multi-objective	Competitive memetic algorithm
Ding et al. (2016b)	*Fm* (*prmu*)	C_{max} and G_{max}	Speed scaling	Multi-objective	MONEH, MOIG
Fang et al. (2011)	*Fm*	C_{max}, P_{max}, G_{max}	Speed scaling	Multi-Objective	MIP
Fang et al. (2013)	*Fm* (*prmu*)	C_{max} and P_{max}	Speed scaling	Objective: C_{max} (Constraint: P_{max})	MIPs, Valid inequalities
Fang and Lin (2013)	*Pm*	TEC and $\sum w_j T_j$	Speed scaling	Objective: TEC + $\sum w_j T_j$	IP, Heuristics, PSO
Jiang et al. (2017)	*Fm* (*prmu,dist*)	C_{max} and G_{max}	Speed scaling	Multi-objective	MOEA/D
Jiang and Wang (2019)	*Fm* (*prmu*,ST_{sd})	TEC and C_{max}	Speed scaling & turn on/off strategy	Multi-objective	MIP, MOEA/D
Lei et al. (2018)	*FHm* (*Rm*)	TEC and $\sum T_j$	Speed scaling	Multi-objective (lexicographic optimization)	TLBO algorithm
Li et al. (2019)	*FHm*	TEC, $\sum T_j$ and C_{max}	Speed scaling	Multi-objective (different importance of objectives)	Two-level imperialist competitive algorithm
Liu & Huang (2014)	*FH2* (*batch*)	P_{max}, G_{max}, $\sum w_j T_j$	Speed scaling	Multi-objective	NSGA-II, adaptive MOGA
Lu et al. (2017)	*Fm* (*prmu*,ST_{sd})	TEC and C_{max}	Speed scaling & turn on/off strategy	Multi-objective	MIP, MOGA with backtracking search
Mansouri et al. (2016)	*F2* (*prmu*, ST_{sd})	TEC and C_{max}	Speed scaling	Multi-objective	MIP, Lower bounds, heuristic
Mansouri and Aktas (2016)	*F2* (*prmu*, ST_{sd})	TEC and C_{max}	Speed scaling	Multi-objective	Heuristics, MOGA
Öztop et al. (2018)	*Fm* (*prmu*)	TEC and $\sum F_j$	Speed scaling	Multi-objective	MILP, MOIG
Öztop et al. (2020)	*Fm* (*prmu*)	TEC and $\sum F_j$	Speed scaling	Multi-objective	MILP, MOIG, MOVBIH, CH

Table 3. 3. (Cont'd) Literature Review (Speed Scaling Strategy)

Reference	**Shop Setting (Comments)**	**Criteria**	**Energy Saving Strategy**	**Objective Structure**	**Solution Approach**
Salido et al. (2016)	*Jm*	TEC and C_{max}	Speed scaling	Multi-objective (weighted objectives)	GA
Shen et al. (2017)	*FHm*	C_{max} and P_{max}	Speed scaling	Objective: C_{max} (Constraint: P_{max})	MIP, Discrete TLBO
Tasgetiren et al. (2018a)	$1(ST_{sd}, r_j)$	TEC and C_{max}	Speed scaling	Multi-objective	MILP, MOVBIH
Tasgetiren et al. (2018b)	$1(ST_{sd})$	TEC and $\sum w_j T_j$	Speed scaling	Multi-objective	MILP, MOIG, MOBIH
Tasgetiren et al. (2019)	*Fm* (*prmu*, *no-idle*)	TEC and C_{max}	Speed scaling	Multi-objective	MILP, MOGA, MOVILS
Wang et al. (2017)	*Fm* (*no-wait*, *dist*)	TEC and C_{max}	Speed scaling	Multi-objective	Cooperative heuristic algorithm
Wang and Wang (2018)	*Fm* (*prmu*,*dist*)	TEC and C_{max}	Speed scaling	Multi-objective	MIP, Knowledge-based cooperative algorithm
Wu and Che (2019)	*Rm*	TEC and C_{max}	Speed scaling	Multi-objective	MIP, Memetic Differential Evolution Algorithm
Yin et al. (2017)	*FJm*	TEC, C_{max} and noise emission	Speed scaling	Multi-objective	MIP, MOGA
Zhang & Chiong (2016)	*Jm*	TEC and $\sum w_j T_j$	Speed scaling	Multi-objective	MOGA
Zhang et al. (2019b)	*FHm* (ST_{sd})	TEC and C_{max}	Speed scaling	Multi-objective	MIP, MDABC based Decomposition
Zheng and Wang (2018)	*Rm*	C_{max} and G_{max}	Speed scaling	Multi-objective	MIP, CMFOA

3.2 Motivation of the Thesis

This thesis addresses the trade-off between the makespan and the total energy consumption (TEC) in the energy efficient HFSP (EHFSP) employing a speed-scaling strategy. This problem is chosen as the thesis research subject for the following motives.

From a practical viewpoint, hybrid flowshop setting is a common shop floor setting and it can be seen in various real production environments, such as chemical (Deal et al., 1994), ceramic tiles (Ruiz and Maroto, 2006), steel (Pan et al., 2013), paper (Sherali et al., 1990), textile (Grabowski and Pempera, 2000) and electronics (Wittrock, 1988; Liu and Chang, 2000; Jin et al., 2002) industries. As mentioned in Chapter 1, makespan, is the main performance criterion for increasing the utilization of resources and obtaining a high throughput. On the other hand, the TEC criterion is important to decrease fuel consumption and slow down environmental deterioration. Note that minimizing energy consumption is an important issue for manufacturing companies due to a series of environmental effects and the increasing energy costs. Therefore, the proposed energy-efficient scheduling techniques can be applied to various real manufacturing environments.

The managers can make decisions considering both production and energy efficiency by using the developed solution methods in this thesis. The proposed methods that employ a speed scaling strategy do not require a significant financial investment from a managerial perspective. Note that machines can operate at multiple speed levels in many real-life production environments. Since there is no need for installing costly energy-efficient machinery, they can also be employed by small and medium-sized enterprises. Consequently, proposed energy-efficient scheduling approaches can provide economic savings from energy resource consumptions as well as the environmental benefits, without making a significant financial investment.

From an academic viewpoint, this thesis will fill the research gap that the multi-objective energy efficient scheduling methods for the hybrid flowshop environment have not been well explored from the perspective of speed scaling strategy. To the best of our knowledge, the speed scaling approach has been employed for the energy-efficient HFSP/FFSP in only a few studies (Liu and Huang, 2014; Chen et al., 2018; Zhang et al., 2019b; Lei et al., 2018; Li et al., 2019; Shen et al., 2017), which are explained as follows. These studies are also summarized in Table 3.4.

Liu & Huang (2014) proposed an NSGA-II and an adaptive MOGA for a very specific two-stage hybrid flowshop, which includes a batch-processing machine followed by two parallel-processing machines, to minimize the total weighted tardiness, carbon footprint, and peak power. Chen et al. (2018) proposed a multi-objective MIP model and a MOGA for the HFSP with lot streaming in order to minimize both makespan and electric power consumption, considering sequence-dependent setup times, release dates, unrelated machines, and machine eligibility restrictions. Recently, Zhang et al.

(2019b) proposed a multi-objective discrete artificial bee colony algorithm (MDABC) based on decomposition for the energy-efficient HFSP with sequence-dependent setup times, where there are different numbers of speed levels for the machines at different stages. Note that, in these aforementioned studies (Liu and Huang, 2014; Chen et al., 2018; Zhang et al., 2019b), special variants of the energy-efficient HFSP were considered such as a particular two-stage hybrid flowshop (Liu and Huang, 2014) and hybrid flowshops with lot streaming and sequence-dependent setup operations (Chen et al., 2018; Zhang et al., 2019b). In this thesis, a general *m*-stage HFSP with makespan and TEC criteria is considered employing a speed-scaling strategy.

Table 3. 4. Literature Review for the EHFSP with Speed Scaling Strategy

Reference	**Shop Setting (Comments)**	**Criteria**	**Energy Saving Strategy**	**Objective Structure**	**Solution Approach**
Chen et al. (2018)	*FHm* (*Rm*, ST_{sd}, r_j, M_j, *lot-streaming*)	TEC and C_{max}	Speed scaling	Multi-objective	MIP, MOGA
Lei et al. (2018)	*FHm* (*Rm*)	TEC and $\sum T_j$	Speed scaling	Multi-objective (lexicographic optimization)	TLBO algorithm
Li et al. (2019)	*FHm*	TEC, $\sum T_j$ and C_{max}	Speed scaling	Multi-objective (different importance of objectives)	Two-level imperialist competitive algorithm
Liu & Huang (2014)	*FH2* (*batch*)	P_{max}, G_{max}, $\sum w_j T_j$	Speed scaling	Multi-objective	NSGA-II, adaptive MOGA
Shen et al. (2017)	*FHm*	C_{max} and P_{max}	Speed scaling	Objective: C_{max} (Constraint:P_{max})	MIP, Discrete TLBO
Zhang et al. (2019b)	*FHm* (ST_{sd})	TEC and C_{max}	Speed scaling	Multi-objective	MIP, MDABC based Decomposition

Recently, Lei et al. (2018) presented a teaching-learning-based optimization (TLBO) algorithm to solve the energy-efficient HFSP with unrelated machines employing a speed scaling approach. They considered both total energy consumption and total tardiness criteria and applied a lexicographical method to deal with the total tardiness as a key objective. Li et al. (2019) also developed a two-level imperialist competitive algorithm for the energy-efficient HFSP with total tardiness, makespan, and total energy consumption criteria, where the energy consumption objective has lower importance. Note that, in Lei et al. (2018) and Li et al. (2019), the tardiness objective was considered together with a TEC criterion for the energy-efficient HFSP, where the studied objectives have different importance. In this thesis, the trade-off between the makespan and TEC is addressed for the energy-efficient HFSP by assuming that both objectives have equal importance. As mentioned before, both makespan and TEC are very important performance criteria for hybrid flowshop environments. Shen et al. (2017) also proposed a MIP model and a discrete teaching-learning-based optimization algorithm for the single-objective HFSP with the makespan criterion under the peak power consumption constraints. Note that, a single-objective HFSP was studied in Shen et al. (2017), where the energy-efficiency was handled with peak power consumption constraints. In this thesis, both makespan and TEC criteria are considered for the HFSP in a multi-objective setting.

Based on these discussions, the motivation of this thesis is to develop effective optimization methods to address the trade-off between the makespan and the total energy consumption in the EHFSP employing a speed-scaling strategy, which has not been investigated very well. Lack of a fundamental model and related solution techniques for the EHFSP with the makespan and TEC criteria that employ speed scaling strategy are remarkable gaps in the current literature that needs to be filled. This thesis aims to fill this research gap by presenting new exact and heuristic solution methods for the problem.

In this thesis, two variants of the speed scaling strategy are studied for the EHFSP, namely, a job-based speed scaling strategy (EHFSP-V1) and a matrix-based speed scaling strategy (EHFSP-V2). New bi-objective MILP models and new bi-objective CP models are proposed for the EHFSP-V1 and EHFSP-V2. New bi-objective metaheuristic algorithms are also proposed for the EHFSP-V1 and EHFSP-V2. Namely, seven bi-objective metaheuristic algorithms are proposed for the EHFSP-V1, which are two variants of the IG, a VBIH and four variants of the ensemble of metaheuristic algorithms. Additionally, four bi-objective metaheuristic algorithms are proposed for the EHFSP-V2, which are two variants of the IG algorithm, a VBIH algorithm and an ensemble of metaheuristic algorithms. In order to evaluate the performance of the proposed methods, benchmark instances are also developed by modifying the well-known HFSP benchmarks from the literature (Carlier and Neron, 2000; Liao et al., 2012; Öztop et al., 2019).

Furthermore, in this thesis, a new constructive heuristic is presented for the single-objective HFSP with the makespan criterion. Two new heuristic fitness calculation approaches are also proposed to compensate for the inefficiency of the standard forward scheduling approach for fitness function calculation in HFSP.

Consequently, this thesis contributes to the energy-efficient scheduling literature by applying the speed scaling strategy to the HFSP, presenting new bi-objective MILP and CP models for the EHFSP, developing original seven effective bi-objective metaheuristic algorithms for the EHFSP-V1 and developing original four effective bi-objective metaheuristic algorithms for the EHFSP-V2. Note that, the augmented ε-constraint method is also employed to solve the proposed bi-objective MILP and CP models.

To the best of our knowledge, this thesis presents a constraint programming approach to the EHFSP for the first time in the literature. As far as we know, CP is employed for the HFSP only in the study of Jouglet et al. (2009), considering the multiprocessor tasks. The authors presented a memetic algorithm for the HFSP with multiprocessor tasks, where each job operation must be processed on several parallel machines simultaneously in each stage. The authors employed a constraint programming based branch & bound algorithm as the local search procedure of their memetic algorithm. Consequently, this thesis also contributes to the HFSP literature by presenting a constraint programming approach.

Furthermore, this thesis contributes to the hybrid flowshop scheduling literature by presenting a new constructive heuristic and two new heuristic fitness calculation approaches for the HFSP. Since the multi-objective studies on the hybrid flowshop scheduling problem have also been limited (Ruiz and Vazquez Rodriguez, 2010), this thesis also contributes to the literature on multi-objective hybrid flowshop scheduling.

CHAPTER 4
PROBLEM DEFINITION & MODEL FORMULATIONS

The HFSP can be considered as a generalization of two classical scheduling problems: the parallel machine scheduling problem and the flowshop scheduling problem. In the HFSP, n jobs must be processed in a series of $m(m > 1)$stages, optimizing a given objective function. All jobs must be sequentially processed following the same production order: stage 1, stage 2,…, stage m. Each job $j \in J$ requires a nonnegative and uninterrupted processing time p_{kj} in stage k. Note that, processing of job j in stage k is referred to the operation o_{kj}. Each stage $k \in M$ has $|I_k| \geq 1$ identical parallel machines, and in at least one of the stages $|I_k| > 1$. As all machines are identical at each stage k, a job can be assigned to any machine $i \in I_k$.

In this thesis, the EHFSP is studied by employing a speed scaling strategy. Unlike the standard HFSP, the machines have variable speed levels in the EHFSP and the speed of a machine can be easily adjusted between jobs. Therefore, the operation time of a job may change based on the chosen speed level. It is assumed that there are three processing speed levels for the machines: fast, normal, and slow. Increasing the speed of a machine decreases processing time, but leads to higher energy consumption. Hence, speed levels of the jobs should be determined carefully to improve both energy and production efficiency. There are two conflicting objectives: minimizing the total energy consumption (TEC) and minimizing the makespan (C_{max}). As mentioned before, the makespan criterion is important for increasing the utilization of resources and obtaining a high throughput. On the other hand, the TEC criterion is also very important in terms of energy-efficient scheduling.

In this thesis, two variants of speed scaling strategy are considered. In the first variant of the EHFSP (EHFSP-V1), speed-scaling is assumed to be job-based due to its simplicity and tractability, as proposed by Öztop et al. (2018), that is, a job must be processed with the same speed level at all stages. Note that, in some flowshops, a single processing speed level is defined for a job through its route for easily tracing and managing the speed arrangements of the job on all stages/machines. Subsequently, the machines process each job based on its pre-defined speed level. This type of job-based speed scaling strategy is practical, particularly for the flowshops with a higher number of stages/machines. In such large systems, defining a different speed level for a job on each stage/machine can be impractical in real-life practice. On the other hand, in the second variant of the EHFSP (EHFSP-V2), the job-based speed scaling strategy assumption is omitted, and it is assumed that the speed of a job can vary from stages to stages. Note that the second version of the problem is relatively more complex than the first version of the problem. For both versions of the EHFSP, MILP and CP models are presented in the following subsections.

The problem notation is given in Table 4.1, and further assumptions are explained as follows: All jobs and machines are available at time zero. No job can be processed on more than one machine at a time, and a machine can process only one operation at a time. Job pre-emption is not allowed. Jobs can wait between stages, and the capacity of buffers is unlimited. Travel times between consecutive stages and setup times are included in the processing times of jobs. Changing the speed of machines does not affect machine quality. All parameters are deterministic and known in advance. Based on the aforementioned assumptions and objectives, the studied bi-objective EHFSP is denoted as FHm, $((PM^{(k)})_{k=1}^{m})||\ C_{max}, TEC$ according to the notation proposed by Vignier et al. (1999), which follows the three-field notation of Graham et al. (1979).

Table 4. 1. Problem Notation

Sets	
M	Set of stages $\{1,2,\ldots,m\}$
J	Set of jobs
L	Set of processing speed levels {1,2,3}
I_k	Set of machines at stage $k \in M$
Parameters	
p_{kj}	Processing time of job $j \in J$ at stage $k \in M$
v_l	Speed factor of processing speed level $l \in L$
λ_l	Conversion factor for processing speed level $l \in L$
α_{ki}	Conversion factor for idle time on the machine $i \in I_k$ at stage $k \in M$
β_{ki}	Power of machine $i \in I_k$ at stage $k \in M$
Q	A very large number

4.1 Mixed-Integer Linear Programming Models for EHFSP-V1 and EHFSP-V2

Decision variables are listed below for the MILP models of EHFSP-V1 and EHFSP-V2:

s_{kj}: Starting time of job *j* at stage *k*

x_{kji}^{l}: 1 if job *j* is processed by machine *i* at stage *k* with speed level *l*, 0 otherwise

y_{kjr}: 1 if job *j* precedes job *r* at stage *k*, 0 otherwise

θ_{ki}: Idle time on machine *i* at stage *k*

C_{max}: Maximum completion time (makespan)

TEC: Total energy consumption

The MILP model is given below for the EHFSP-V1:

$$Minimize\ C_{max} \tag{4-1}$$

$$Minimize\ TEC \tag{4-2}$$

Subject to:

$$\sum_{i\in I_k} x_{kji}^{l} = \sum_{i\in I_{k+1}} x_{k+1,j,i}^{l} \qquad \forall (k,k+1)\in M, j\in J, l\in L \tag{4-3}$$

$$s_{mj} + \sum_{i\in I_m}\sum_{l\in L}\frac{p_{mj}}{v_l}x_{mji}^{l} \le C_{max} \qquad \forall j\in J \tag{4-4}$$

$$\sum_{i\in I_k}\sum_{l\in L} x_{kji}^{l} = 1 \qquad \forall j\in J, k\in M \tag{4-5}$$

$$s_{k+1,j} - s_{kj} \ge \sum_{i\in I_k}\sum_{l\in L}\frac{p_{kj}}{v_l}x_{kji}^{l} \quad \forall j\in J, (k,k+1)\in M \tag{4-6}$$

$$s_{kj} - \left(s_{kr} + \sum_{l\in L}\frac{p_{kr}}{v_l}x_{kri}^{l}\right) + Q\left(2 + y_{kjr} - \sum_{l\in L}x_{kji}^{l} - \sum_{l\in L}x_{kri}^{l}\right) \ge 0$$
$$\forall j,r\in J: j<r, k\in M, i\in I_k \tag{4-7}$$

$$s_{kr} - \left(s_{kj} + \sum_{l\in L}\frac{p_{kj}}{v_l}x_{kji}^{l}\right) + Q\left(3 - y_{kjr} - \sum_{l\in L}x_{kji}^{l} - \sum_{l\in L}x_{kri}^{l}\right) \ge 0$$
$$\forall j,r\in J: j<r, k\in M, i\in I_k \tag{4-8}$$

$$\theta_{ki} = C_{max} - \sum_{j\in J}\sum_{l\in L}\frac{p_{kj}}{v_l}x_{kji}^{l} \qquad \forall k\in M, i\in I_k \tag{4-9}$$

$$TEC = \sum_{j\in J}\sum_{k\in M}\sum_{i\in I_k}\sum_{l\in L}\frac{\beta_{ki}p_{kj}\lambda_l}{60v_l}x_{kji}^{l} + \sum_{k\in M}\sum_{i\in I_k}\frac{\alpha_{ki}\beta_{ki}}{60}\theta_{ki} \tag{4-10}$$

$s_{kj} \geq 0 \quad \forall k \in M, j \in J, \; y_{kjr} \in \{0,1\} \quad \forall j, r \in J, k \in M$

$x_{kji}^{l} \in \{0,1\} \, \forall k \in M, j \in J, i \in I_k, l \in L$ (4-11)

The objective functions (4-1) and (4-2) minimize the makespan (C_{max}) and the TEC, respectively. Constraint set (4-3) imposes a single speed level for a job through the stages. Constraint set (4-4) determines the maximum completion time. Constraint set (4-5) ensures that each job passes through all stages and is assigned to exactly one machine at every stage. Through constraint set (4-6), the next operation of a job can be started after its preceding operation is completed. Constraint sets (4-7) and (4-8) determine the sequence of the jobs on each machine, where Q is a large integer. For two jobs assigned to the same machine, the next job can only be started after the preceding job is processed. Constraint set (4-9) computes the idle time on each machine, while constraint set (4-10) calculates the total energy consumption in kilowatt-hours, as proposed in (Mansouri et al., 2016). Both the processing time and the idle time energy consumption are reflected in the calculation of TEC. Finally, the constraint set (4-11) defines the decision variables.

The MILP model of the EHFSP-V2, in which the speed of a job can change from stages to stages, is given below. This model is identical to the above one except that the constraint set (4-3), which imposes a single speed level for a job through the stages, is omitted.

Minimize (4-1) and (4-2)

Subject to:

(4-4) - (4-11).

4.2 Constraint Programming Models for EHFSP-V1 and EHFSP-V2

In this subsection, the CP models are presented for the EHFSP-V1 and EHFSP-V2. Before presenting the CP models, a brief introduction to the constraint programming technique is provided in the following subsection (Section 4.2.1). Then, the CP models are provided in Section 4.2.2.

4.2.1 Constraint Programming

Constraint programming is an efficient approach for modeling and solving combinatorial optimization problems. CP employs *global constraints* as well as the traditional mathematical programming constraints. Global constraints are more effective than the traditional constraints, as they express the relations between variables more easily and employ specialized filtering algorithms due to the problem's structure.

The variables and global constraints used in this thesis are formally defined as below (IBM ILOG CPLEX, 2017):

***Variables*:**

Interval Variable: It denotes an interval of time whose position in a schedule is not pre-determined. An interval is symbolized by a starting time, an ending time, and a duration. Interval variables can be *optional*, meaning that the existence of them in the final solution schedule is part of the decisions in the problem. The optional concept is very useful when activities can be performed on several different resources.

Sequence Variable: A sequence variable indicates a sequence for a set of interval variables. For a given set of interval variables $\{t_1, t_2, t_3, t_4\}$, the value of the sequence variable can be $(t_1,\ t_3,\ t_4,\ t_2)$.

***Global Constraints*:**

$alternative\ (a, \{b_1, .., b_n\})$: This constraint selects an exclusive alternative from a set of optional interval variables $\{b_1, .., b_n\}$. If interval a exists, then exactly one of intervals $\{b_1, .., b_n\}$ exists, and a starts and finishes together with the selected one. If a does not exist, then all b intervals do not exist. This constraint is generally useful to model the choice of one resource among a set of candidate resources and to model alternative performing modes for activities.

$noOverlap\ (p)$: This constraint is used to prevent overlapping of intervals in a sequence variable p. It assures that the sequence is formed by a series of non-overlapping intervals, i.e., any interval in the sequence is finished before the starting time of the next interval in the sequence. This constraint is generally used for formulating disjunctive resources.

$endBeforeStart(predecessor, successor)$: This constraint ensures that, if both interval variables $predecessor$ and $successor$ exist, then $successor$ cannot start before $predecessor$ has been finished.

4.2.2 CP Models for EHFSP-V1 and EHFSP-V2

The CP models are presented for the EHFSP-V1 and EHFSP-V2, using the OPL API of CP Optimizer. Interval variables are defined for representing the operations of the jobs in each stage. Furthermore, optional interval variables represent the performing of job *j* in stage *k* on machine $i \in I_k$ with speed level *l*. The model also declares several sequence variables associated with each machine *i* in stage *k*. Each sequence constraint gathers all the optional interval variables associated with a specific machine. Decision variables are listed below for the proposed CP model:

t_{kj} : Interval variable for the operation of job *j* in stage *k*

z_{ijk}^{l}: Optional interval variable for the operation of job *j* in stage *k* on machine $i \in I_k$ with speed level *l* and duration of $\left({p_{kj}}/{v_l}\right)$

ms_{ik}: Sequence variable for machine $i \in I_k$ in stage *k* over $\{z_{ijk}^{l} \mid j \in J, l \in L\}$

The calculations of C_{max}, TEC and θ_{ki} (idle time on the machine *i* at stage *k*) are handled by defining below expressions:

$C_{max} = max_{j\in J}\ (endOf\ (t_{mj}))$

$\theta_{ki} = C_{max} - \sum_{j\in J}\sum_{l\in L}\frac{p_{kj}}{v_l}\ presenceOf(z_{ijk}^{l})$

$TEC = \sum_{j\in J}\sum_{k\in M}\sum_{i\in I_k}\sum_{l\in L}\frac{\beta_{ki}p_{kj}\lambda_l}{60v_l}\ presenceOf\left(z_{ijk}^{l}\right) + \sum_{k\in M}\sum_{i\in I_k}\frac{\alpha_{ki}\beta_{ki}}{60}\theta_{ki}$

Then, the CP model is given below for the EHFSP-V1:

Minimize C_{max} (4-12)

Minimize TEC (4-13)

Subject to:

$alternative\ \left(t_{kj}, all\ (i\ in\ I_k, l\ in\ L) z_{ijk}^{l}\right) \quad \forall j \in J, k \in M$ (4-14)

$noOverlap(ms_{ik}) \quad \forall k \in M, i \in I_k$ (4-15)

$endBeforeStart(t_{kj}, t_{k+1,j}) \quad \forall j \in J, (k, k+1) \in M$ (4-16)

$\sum_{i\in I_k} presenceOf(z_{ijk}^{l}) = \sum_{i\in I_{k+1}} presenceOf(z_{ij,k+1}^{l})$

$\forall j \in J, (k, k+1) \in M, l \in L$ (4-17)

The objective functions (4-12) and (4-13) minimize the makespan (C_{max}) and the TEC, respectively. Constraint set (4-14) ensures that each operation of job j is assigned to exactly one machine at each stage, and one speed level is chosen for each operation. Constraint set (4-15) states that each machine can perform only one operation at a time. For two jobs assigned to the same machine, the next job can be started after the preceding job is completed. Through constraint set (4-16), the next operation of a certain job in stage $k+1$ can be started after its preceding operation in stage k is finished. Constraint set (4-17) imposes a single speed level for a job through the stages.

The CP model of the EHFSP-V2, in which the speed of a job can vary from stages to stages, is given below. This model is identical to the above one except that the constraint set (4-17) is omitted.

Minimize (4-12) and (4-13)

Subject to:

(4-14) - (4-16).

The processing times (p_{kj}) are defined as integer values in this thesis. However, when a processing time p_{kj} is divided by a speed factor v_l, the resulting duration can be a floating number. As mentioned in Chapter 7, there are three processing speed levels for the machines in this thesis, and the corresponding processing speed factors are $v_l = \{1.2, 1.0, 0.8\}$. In OPL API of CP Optimizer, interval variables cannot have a duration with a floating value. Hence, a transformation procedure has been applied to define integer processing times for interval variables. Note that the operations in Eq. (4-18) are equivalent. Namely dividing the processing times (p_{kj}) by $v_l = \{1.2, 1.0, 0.8\}$ is equivalent to multiply them by $v'_l = \{\frac{10}{12}, \frac{12}{12}, \frac{15}{12}\}$. Hence, we can obtain integer processing time values by multiplying the processing time (p_{kj}/v_l) expressions by a constant value 12, i.e., multiplying the processing times (p_{kj}) by the vc_l ={10, 12, 15}. Accordingly, each (p_{kj}/v_l) expression in the above formulation is replaced by an expression ($p_{kj} * vc_l$).

$$(l=1)\,\frac{p_{kj}}{1.2} = p_{kj} * \frac{10}{12}, \quad (l=2)\,\frac{p_{kj}}{1.0} = p_{kj} * \frac{12}{12}, \quad (l=3)\,\frac{p_{kj}}{0.8} = p_{kj} * \frac{15}{12} \qquad (4\text{-}18)$$

Consequently, the resulting interval variable z^l_{ijk} is defined as below:

z_{ijk}^{l}: Optional interval variable for the operation of job j in stage k on machine $i \in I_k$ with speed level l and duration of $(p_{kj} * vc_l)$

Since the original processing time (p_{kj}/v_l) expressions are multiplied by 12 during the transformation procedure, the resulting C_{max} and TEC values will be 12 times the original C_{max} and TEC values. Therefore, the C_{max}, TEC and θ_{ki} expressions are also modified by dividing them 12, as follows:

$$C_{max} = (max_{j \in J}\ (endOf\ (t_{mj})))/12$$

$$\theta_{ki} = C_{max} - \sum_{j \in J} \sum_{l \in L} \frac{p_{kj} * vc_l}{12}\ presenceOf(z_{ijk}^{l})$$

$$TEC = \sum_{j \in J} \sum_{k \in M} \sum_{i \in I_k} \sum_{l \in L} \frac{\beta_{ki} \lambda_l (p_{kj} * vc_l)}{60 * 12}\ presenceOf\left(z_{ijk}^{l}\right) + \sum_{k \in M} \sum_{i \in I_k} \frac{\alpha_{ki} \beta_{ki}}{60} \theta_{ki}$$

4.3 Conflicting Objectives

In order to show the conflict between minimizing C_{max} and TEC, the Pareto frontiers are obtained for a small problem with five jobs and five stages. For the same instance, Figure 4.1 demonstrates the Pareto frontier for the EHFSP-V1 and Figure 4.2 shows the Pareto frontier for the EHFSP-V2. As seen from the figures, the number of Pareto-optimal solutions in EHFSP-V2 is much more than the number of Pareto-optimal solutions in EHFSP-V1. Therefore, the EHFSP-V2 is relatively more complex to solve than the EHFSP-V1.

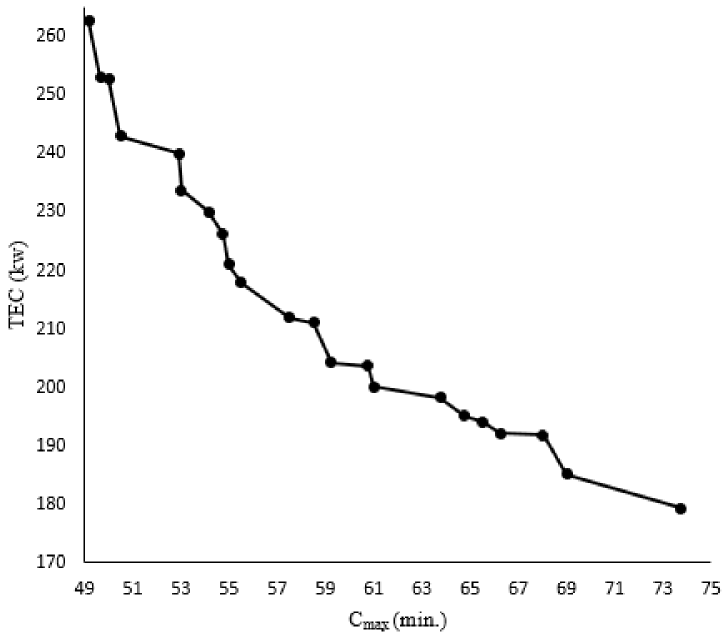


Figure 4. 1. Pareto Frontier of a Small Problem (EHFSP-V1)

As shown in Figures 4.1 and 4.2, the two objectives are conflicting, and they cannot be optimized concurrently. Hence, multi-objective optimization techniques should be employed to solve the bi-objective EHFSP. The multi-objective optimization technique employed in this thesis is explained in Chapter 5 as well as the related terminology.

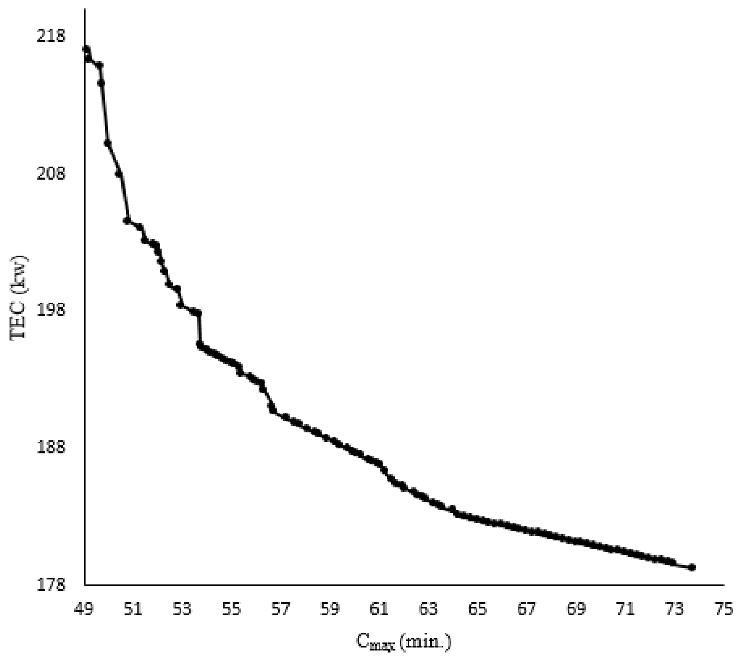


Figure 4. 2. Pareto Frontier of a Small Problem (EHFSP-V2)

CHAPTER 5
MULTI-OBJECTIVE OPTIMIZATION

In this section, common solution techniques to solve multi-objective optimization problems are explained as well as the related terminology. Then, the multi-objective optimization method used in this thesis is explained in detail.

5.1 Terminology

A multi-objective optimization problem (MOP) includes several conflicting objective functions to be optimized simultaneously. As these objective functions conflict with each other, i.e., improvement of one objective function may cause to worsening of another, there is no single optimal solution for these problems. In this case, a set of most preferred solutions, namely Pareto-optimal solutions, is important for the decision-maker. Therefore, the optimality concept is replaced with the concept of Pareto-optimality for the MOPs. The Pareto-optimality and the dominance relation concepts are formally defined for a minimization MOP as follows (Okabe et al. 2003):

MOP: minimize $F(\mathrm{x}) = (f_1(\mathrm{x}), . . . , f_c(\mathrm{x}))^C$

s.t. $\mathrm{x} \in \Omega$,

Dominance: A solution x_i dominates another solution x_j if the two following conditions are satisfied (denoted as $x_i \prec x_j$):

- $\forall c \in 1,..,C; f_c(x_i) \leq f_c(x_j)$
- $\exists c \in 1,..,C; f_c(x_i) < f_c(x_j)$

Weakly Dominance: A solution x_i weakly dominates another solution x_j (denoted as $x_i \preccurlyeq x_j$) if :

- $\forall c \in 1,..,C; f_c(x_i) \leq f_c(x_j)$

***Pareto-Optimality*:** A solution x is named as Pareto-optimal (efficient) if $\nexists\ y \in \Omega;\ \mathrm{y} \prec \mathrm{x}$.

***Pareto-Optimal Frontier (Pareto-Optimal Set)*:** The union of all Pareto optimal solutions $x \in \Omega$ is called as Pareto-optimal frontier (P_{true}).

***Pareto-Optimal Solution Set*:** A finite number of Pareto-optimal solutions that belong to P_{true}, are named as Pareto Optimal Solution Set (P), where $P \subseteq P_{true}$. As it is generally impossible to obtain P_{true}, P is commonly used as an approximation to P_{true}.

***Solution Set and Non-dominated Solution Set*:** The set of solutions obtained by an algorithm is named as Solution Set (S). The solutions in S that are not dominated by others in the set form the Non-dominated Solution Set (S_N). As only non-dominated solutions are generated in S for most of the cases, S_N is usually defined with S.

***Reference Set*:** In general, the Pareto optimal set is unknown for most of the cases. In these cases, the desired reference set (R) is designed with pre-defined solutions. This reference set is usually formed by combining the best-known solutions from several algorithms.

5.2 Solution Methods

As mentioned in Chapter 4, the studied EHFSP in this thesis is a bi-objective optimization problem. Hence, no single optimal solution exists due to the conflict between TEC and C_{max} objectives. Yet, a set of Pareto-optimal solutions can be found by handling the trade-off between these objectives. As mentioned above, a Pareto-optimal solution cannot be improved in one objective without deteriorating the other one, and it is not dominated by any other feasible solution. Since the two objectives of the EHFSP are conflicting and cannot be optimized concurrently, multi-objective solution methods must be employed, which are explained as follows.

Generally, the solution methods for solving MOPs are divided into three main categories due to the impact of the decision-maker (Mavrotas, 2009):

Priori methods:

At the beginning of the solution process, the decision-maker defines its preferences by either determining goals or weights for the objective functions. In the former one, a certain numeric goal is determined for each objective, and the (weighted) sum of deviations of the objective functions from their respective goals is minimized. In the latter one, weights are defined for each objective, and the weighted combination of the objectives is optimized. The drawback of these methods is that it is hard to define the decision maker's preferences precisely in terms of goals or weights in advance.

Interactive methods:

The decision-maker is involved in the whole solution process interactively, meaning that he/she gradually lead the process with his/her preferences to the most preferred solution. The disadvantage of this method is that the decision-maker never knows the Pareto-optimal solution set and the most preferred solution is chosen among the solutions obtained so far.

Posteriori (generation) methods:

The non-dominated solutions are obtained at the beginning, and then the decision-maker selects the one solution among them. The disadvantage of these methods is that they require high computational effort. Nevertheless, they also have advantages. They are preferable whenever the decision-maker is not available, as they find all possible alternatives in advance. Furthermore, the confidence of the decision-maker on these methods is generally high, as they are able to find all potential solutions.

The most commonly used generation methods are the weighting method and the ε-constraint method. It is known that the ε-constraint method has several advantages as compared with the weighting method (Mavrotas, 2009). In this thesis, we use the augmented ε-constraint method to solve the proposed bi-objective MILP and CP models. The augmented ε-constraint method is an extension of the well-known ε-constraint method, and it avoids obtaining weakly Pareto-optimal solutions (Mavrotas, 2009). Note that, one main disadvantage of the traditional ε-constraint method is about the generation of the weakly Pareto-optimal solutions. In order to overcome this shortcoming, the augmented ε-constraint method was proposed by Mavrotas (2009).

Similar to the ε-constraint method, one of the objective functions is optimized in the augmented ε-constraint method employing the other objective functions as constraints. Then, a series of single-objective models are optimally solved by systematically changing the right-hand side values of the objective function constraints. However, in the augmented ε-constraint method, the objective function constraints are transformed into equalities by including the appropriate slack/surplus variables. Then, these slack/surplus variables are used as a second term in the objective function with lower weights in a lexicographic manner, to ensure that only Pareto-optimal solutions are generated. Consequently, the augmented ε-constraint method can be used to obtain an exact Pareto-optimal solution set by appropriately conducting a parametric search on the right-hand side values of the objective function constraints, since it guarantees the Pareto-optimality of the obtained solutions. The details of the weighting method, ε-constraint method, and augmented ε-constraint method are explained in the following subsections.

5.3 Weighting Method

In the weighting method, a weight (w_c) is assigned to each objective function c and the weighted sum of the objectives is minimized. Several objective functions are combined into a single objective function as follows, where $\sum_{c=1}^{C} w_c = 1$:

minimize $\sum_{c=1}^{C} w_c f_c(x)$ s.t. x ∈Ω,

In this approach, non-dominated solutions are obtained by trying different weights for the objectives. The weighting method is also extended as a weighting method with normalization. In this case, the objective functions are normalized and they take values between 0 and 1.

As pointed out by Mavrotas (2009), there can be many redundant runs in the weighting method, as there can be a lot of combinations of weights that generate the same efficient solution. Furthermore, in the weighting method, the scaling of the objective functions may affect the results. Thus, the objective functions should be normalized before defining the weighted sum. However, in the ε -constraint method, this is not required. Moreover, the number of generated Pareto-optimal solutions can be controlled in the ε-constraint method by defining a proper ε level. On the other hand, it is not easy to control this number in the weighting method.

5.4 ε-Constraint and Augmented ε-Constraint Methods

In the ε-constraint method, one of the objectives is optimized and the other objective functions are defined as constraints. Therefore, the aforementioned minimization MOP is reformulated as follows:

minimize $f_1(\mathrm{x})$

s.t.

$f_2(\mathrm{x}) \leq e_2, f_3(\mathrm{x}) \leq e_3, \ldots, f_c(\mathrm{x}) \leq e_c$

$\mathrm{x} \in \Omega$

Using this reformulation, solutions are obtained by parametrical changes on the right side of the constrained objective functions (e_c). To properly apply the ε-constraint method, the range of each objective function must be obtained. The most widely used approach for obtaining these ranges is to use *payoff tables* that include the results from the individual optimization of each objective function. In the construction of these tables, it must be guaranteed that the obtained solutions from the individual optimization of the objective functions are certainly Pareto-optimal solutions. In the presence of alternative optimal solutions, the obtained solution may not be a Pareto-optimal solution. In order to handle this issue, lexicographic optimization is commonly employed for each objective function. In lexicographic optimization, objectives are optimized lexicographically. Namely, the primary objective function is initially optimized, and then among the alternative optimal solutions, the second important objective is optimized with the optimal value of the primary objective, and so on.

However, in the standard ε-constraint method, obtained solutions are usually not Pareto-optimal solutions. In order to overcome this shortcoming of the standard ε-constraint method, the augmented ε-constraint method is proposed by Mavrotas (2009). The augmented ε-constraint method is an extended version of the well-known ε-constraint method, and it avoids obtaining weakly Pareto-optimal solutions (Mavrotas, 2009). Similar to the ε-constraint method, one of the objective functions is optimized using the other objective functions as constraints. However, in this method, the objective function constraints are transformed into equalities by including the appropriate slack/surplus variables. These slack/surplus variables are used as a second term in the objective function with a lower weight in a lexicographic manner, to ensure that only Pareto-optimal solutions are generated. In order to avoid any scaling problems, slack/surplus variables are normalized in the objective function by dividing them into the ranges of the respective objective functions. The details of this method can be found in Mavrotas (2009). The formulation of the MOP based on this method is given below, where *eps* is a sufficiently small number and r_c is range of the objective function *c*.

$$\text{minimize } f_1(\text{x}) - eps\left(\frac{s_2}{r_2} + \frac{s_3}{r_3} + \cdots + \frac{s_c}{r_c}\right)$$

s.t.

$$f_2(\text{x}) + s_2 = e_2,\ f_3(\text{x}) + s_3 = e_3, \ldots,\ f_c(\text{x}) + s_c = e_c$$

$$\text{x} \in \Omega$$

The performances of the ε-constraint and the augmented ε-constraint methods (with and without lexicographic optimization in construction of payoff tables) are shown with an illustrative example below. In this bi-objective example ($f_1 = x_2, f_2 = 4x_1 - x_2$), the range of the second objective function is divided into six equal intervals and a constant ε level is determined. Then, the $e_2$value is gradually decreased using this constant ε level. In both Figures 5.1 and 5.2, Pareto-optimal points are marked with a red square.

As shown in Figure 5.1, the ε-constraint method finds 2 Pareto-optimal solutions without lexicographic optimization, while it finds 4 Pareto-optimal solutions with lexicographic optimization. In the former one, 5 points are dominated by other points (B, C, D, and E), while in the latter one, 3 points are dominated by other points (B, C, and D).

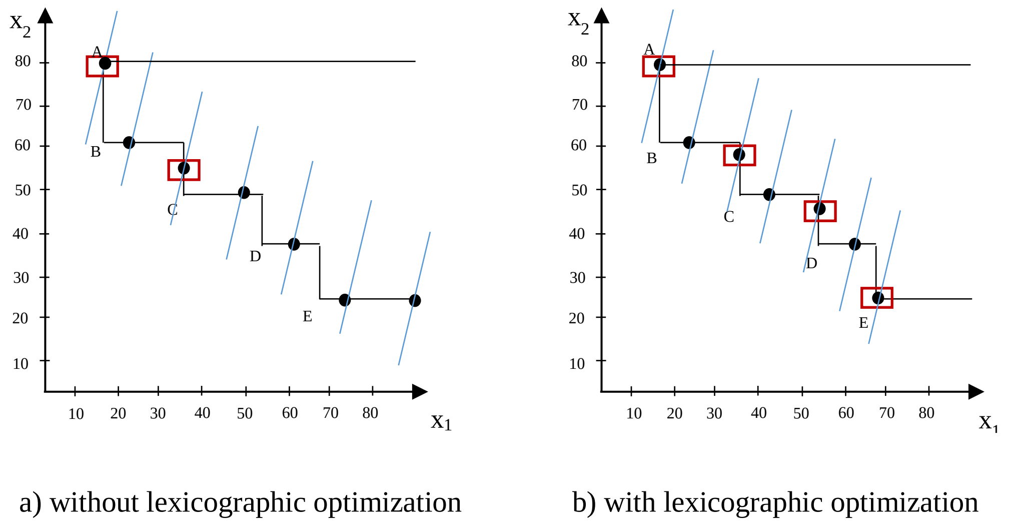
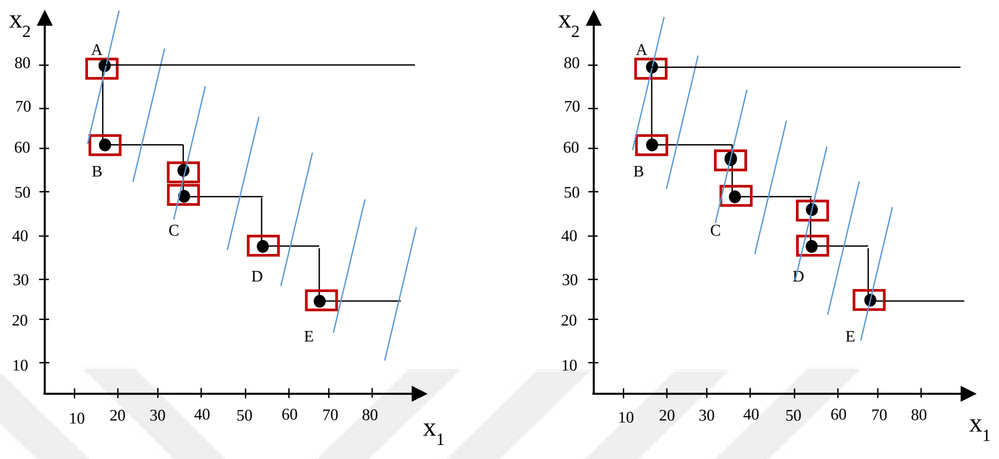


Figure 5. 1. ε-Constraint Method

As shown in Figure 5.2, the augmented ε-constraint method finds 6 Pareto-optimal solutions without lexicographic optimization, while it finds 7 Pareto-optimal solutions with lexicographic optimization. In the former one, point E is obtained twice, while in the latter one, a different Pareto-optimal solution is obtained in each iteration.



a) without lexicographic optimization b) with lexicographic optimization

Figure 5. 2. Augmented ε-Constraint Method

As shown in the above example, the augmented ε-constraint method with lexicographic optimization is an efficient way to generate only Pareto-optimal solutions. Therefore, in this thesis, this combined method is employed to solve bi-objective MILP and CP models for the EHFSP.

The general outline of the augmented ε-constraint method with lexicographic optimization is provided in Figure 5.3 for a bi-objective minimization problem. As shown in Figure 5.3, minimizing f_1 is considered as the objective and the second objective function f_2 is defined as a constraint. Initially, the lexicographic optimization is used for each objective function in order to obtain the payoff table with only Pareto-optimal solutions. Then, starting with an upper bound (f_2^{max}) on f_2, which is found from the payoff table, the single-objective model is iteratively solved optimally by systematically decreasing the right-hand side value (e_2) of the constraint on f_2 with a predetermined ε level, until the minimum value (f_2^{min}) of f_2 is reached.

```
Construct the payoff table using lexicographic optimization for each objective function:
    Payoff Table= $\begin{bmatrix} f_1^{min} & f_2^{max} \\ f_1^{max} & f_2^{min} \end{bmatrix}$
    where $f_1^{min} = \min\{f_1(x)\}, f_2^{min} = \min\{f_2(x)\}$,
    $f_1^{max} = \min\{f_1(x): f_2(x) = f_2^{min}\}, f_2^{max} = \min\{f_2(x): f_1(x) = f_1^{min}\}$.
Add $(f_1^{min}, f_2^{max})$ to the Pareto-optimal solution set
Calculate the range $r_2$ of the second objective function from the payoff table
$e_2 = f_2^{max} - \varepsilon$
While $(e_2 \geq f_2^{min})$do
        Solve the single-objective problem (SOP) optimally:
         SOP:    minimize $f_1$(x)- $eps\ (\frac{s_2}{r_2})$
                   s.t.
                   $f_2(\mathrm{x}) + s_2 = e_2$
                   x $\in \Omega$
         Add the optimal solution value $(f_1^*, f_2^*)$ of the SOP to the Pareto-optimal solution set
         $e_2 = e_2 - \varepsilon$
EndWhile
Report the Pareto-optimal solution set
```

Figure 5.3. General Outline of the Augmented ε-Constraint Method

CHAPTER 6
METAHEURISTIC ALGORITHMS

Since the HFSP is known to be NP-hard (Gupta, 1988) even for the single-objective, the studied bi-objective EHFSP in this thesis is also NP-hard. Hence, energy-efficient bi-objective metaheuristic algorithms are also proposed for the EHFSP in this thesis.

In this section, the proposed seven energy-efficient bi-objective metaheuristic algorithms; namely, two variants of the IG algorithm (E_IG, E_IG_{ALL}), a VBIH algorithm (E_VBIH) and four variants of the ensemble of metaheuristic algorithms ($\text{E\_EM}, \text{E\_EM}_{\text{HFR}}, \text{E\_EM}_{\text{HFN}}, \text{E\_EM}_{\text{HFRN}}$) are explained for the EHFSP-V1. Then, the proposed four energy-efficient bi-objective metaheuristic algorithms; namely, two variants of the IG algorithm (E_IG2, $\text{E\_IG2}_{\text{ALL}}$), a VBIH algorithm (E_VBIH2), and an ensemble of metaheuristic algorithms (E_EM2) are explained for the EHFSP-V2.

6.1 Solution Representation & Fitness Value Calculation

In this thesis, permutation-based encoding is used for fitness value calculation by employing a forward scheduling approach, which is commonly used in the HFSP literature. Initially, the standard forward scheduling approach is explained for the single-objective version of the problem with the makespan criterion in Section 6.1.1. Then, the proposed heuristic fitness calculation approaches are presented in Section 6.1.2 to compensate for the inefficiency of the standard forward scheduling approach. Afterward, the energy-efficient (bi-objective) extensions of the fitness calculation approaches are explained in Sections 6.1.3 and 6.1.4, as well as the solution representations for the EHFSP-V1 and EHFSP-V2.

6.1.1 Standard Forward Scheduling Approach

The permutation-based encoding is an indirect encoding scheme, where the jobs are assigned to the most available machine at the first stage according to the initial permutation π. Then, for the remaining stages, the forward scheduling approach is used to decode a solution, where the jobs are assigned to machines according to their earliest release times at the previous stage. Namely, in each following stage, the jobs are ordered with respect to their release (completion) times from the previous stage and assigned to the most available machine according to that order. In this way, a complete schedule can be obtained for a given initial permutation.

Given an instance in which there are two machines in each stage with processing times $p_{kj} = \begin{pmatrix} 4 & 3 & 5 & 2 & 1 \\ 5 & 4 & 5 & 1 & 4 \end{pmatrix}$ and an initial permutation $\pi = \{1, 2, 3, 4, 5\}$, a complete schedule can be generated as follows: At the first stage, the jobs are assigned to the machines through the initial permutation π. In the second stage, the jobs are ordered with respect to their release times from the first stage, thus resulting in another permutation $\pi^1 = \{2, 1, 4, 5, 3\}$. Then, the jobs are assigned in this order to the machines in the second stage. The Gantt chart for the initial permutation π is illustrated in Figure 6.1 with a makespan $(\mathrm{C_{max}}) = 14$.

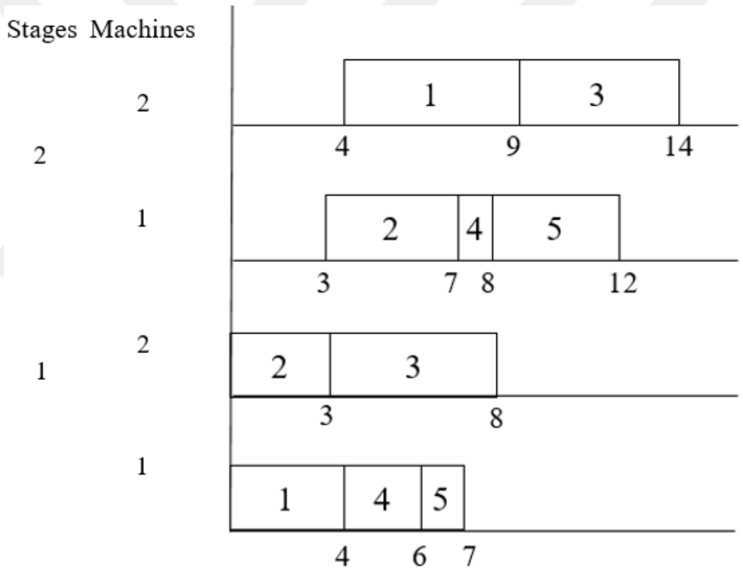


Figure 6. 1. Gantt Chart with $C_{max} = 14$

6.1.2 Heuristic Fitness Calculation Approaches

The aforementioned standard forward scheduling approach is generally an efficient way to obtain a complete schedule. However, some solutions may be unexplored in this approach due to its greedy fashion. In this thesis, two new swap move-based heuristic fitness calculation approaches are proposed, to compensate for the inefficiency of the standard forward scheduling approach and to further enhance the performance of the algorithms. Namely, the standard forward scheduling approach is modified by employing swap moves on the job permutations of some stages, in which jobs are ordered according to their completion times at the previous stage. The aim is to explore the neighboring schedules for a given initial permutation π. Note that, employing a swap move on the job permutation of a stage can lead to a different complete schedule with a different fitness function value.

Consider the above example with $p_{kj} = \begin{pmatrix} 4\,3\,5\,2\,1 \\ 5\,4\,5\,1\,4 \end{pmatrix}$ and an initial permutation $\pi = \{1, 2, 3, 4, 5\}$. As mentioned above, after the jobs have been assigned to the most available machine in the first stage, the resulting permutation is $\pi^1 = \{2, 1, 4, 5, 3\}$ when the jobs are sorted in increasing order of their release times from the first stage. Consequently, by employing the standard forward scheduling approach, we obtain the job sequences {2, 4, 5} and {1, 3} on machine 1 and 2, respectively. The complete Gantt chart for the standard forward scheduling approach is shown in Figure 6.1 and the makespan value is 14.

As can be seen from Figure 6.1, both jobs 3 and 5 are ready to be processed, when job 4 is completed on machine 1 of stage 2. The standard forward scheduling approach chooses job 5, as its release time is shorter than job 3. However, job 3 can also be selected without increasing the idle time of the machine. Thus, the positions of jobs 5 and 3 are swapped in the permutation $\pi^1 = \{2, 1, 4, \mathbf{5}, \mathbf{3}\}$ and a new permutation $\pi^1 = \{2, 1, 4, \mathbf{3}, \mathbf{5}\}$ is generated for stage 2. When the forward scheduling method is applied to stage 2 according to this new permutation, the resulting complete schedule has a smaller makespan, which is equal to 13 as shown in Figure 6.2.

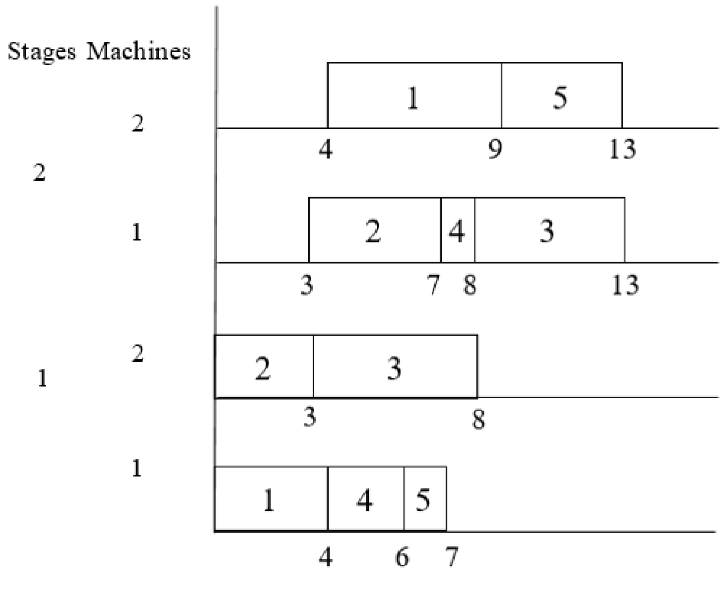


Figure 6. 2. Gantt Chart with $C_{max} = 13$

The drawback of the standard forward scheduling approach has also been addressed by Pan et al. (2014), and a local search procedure has been proposed for a given complete solution with the forward scheduling approach. In the local search procedure of Pan et al. (2014), starting from the second stage, several exchange moves are employed on the permutation of jobs at each stage. Namely, at each stage, each job in the permutation is exchanged with all possible jobs that are ready to be processed, where the complete schedule is computed with a forward scheduling approach for each exchange move. Since the local search procedure of Pan et al. (2014) evaluates the complete schedules for all exchange moves, it requires high computational time. Hence, the authors only applied this local search procedure to the best solution found by their discrete artificial bee colony algorithm. In this thesis, two simple heuristic fitness calculation approaches are proposed by employing only a single swap move in at most $m-1$ stages. As the proposed heuristic fitness calculation approaches do not evaluate the complete schedule at each iteration, they are very fast in terms of computational time.

In the first heuristic fitness calculation approach, namely, heuristic fitness calculation with random swap moves (HFR), a random stage number pt is chosen from the set of stages {2, 3,..., m}. Note that, the first stage is not included in this selection as the swap move is not applied to the first stage. For each stage k, which is greater than or equal to pt ($k \geq pt$), a single swap operation is employed on the resulting permutation of jobs from stage $k-1$ (π^{k-1}), in which jobs are ordered according to their completion times at stage $k-1$. Namely, for each stage k ($k \geq pt$), after we swap two jobs randomly in the resulting permutation π^{k-1}, we employ this new permutation at stage k to apply the forward scheduling method. The outline of the HFR procedure is given in Figure 6.3.

```
Heuristic Fitness Calculation with Random Swap Moves
Schedule the jobs at first stage according to the initial permutation $\pi$
$pt$ = random stage number from $\{2, 3, \ldots, m\}$
for ($k = 2$ to $m$) do
Generate the sequence of jobs $\pi^{k-1}$ according to completion times of the jobs at stage $k$-1
   if ($pt \leq k$) do
        swap two jobs randomly in $\pi^{k-1}$
   end if
   Schedule the jobs at stage $k$ according to the order $\pi^{k-1}$
end for
```

Figure 6. 3. Heuristic Fitness Calculation with Random Swap Moves

In the second heuristic fitness calculation approach, namely, heuristic fitness calculation with neighbor swap moves (HFN), a swapping probability sp is defined to decide whether or not to apply swap operation on the permutation. Then, if it is decided to apply a swap operation, the swap operation is employed only on the neighboring jobs in the permutation. Namely, for each stage k ($k \geq 2$), we employ a swap operation on the resulting permutation of jobs from stage $k-1$ (π^{k-1}) according to the given swapping probability sp. We generate a uniform random number r between 0 and 1, and, if $r \leq sp$, we employ a swap operation on the permutation. Furthermore, in order to apply the swap operation, we choose the two closest neighbor job pairs in the permutation, based on their release times from the previous stage $k-1$. Then, we choose one of these pairs to apply swap operation with a probability np. That is, we generate another uniform random number q between 0 and 1, and, if $q \leq np$, we employ the swap operation on the job pair, which has the minimum difference between the release times at stage $k-1$. In the case of $q > np$, we employ the swap operation on the second closest job pair. After the swap operation on permutation π^{k-1}, we employ the new permutation at stage k to apply a forward scheduling method. The outline of the HFN procedure is provided in Figure 6.4.

```
Heuristic Fitness Calculation with Neighbour Swap Moves
Schedule the jobs at first stage according to the initial permutation $\pi$
for ($k = 2$ to $m$) do
 Generate the sequence of jobs $\pi^{k-1}$ according to completion times of the jobs at stage $k$-1
   if ($r \sim U(0,1) \leq sp$) then do
       choose the two closest neighbor job pairs in $\pi^{k-1}$
       if ($q \sim U(0,1) \leq np$) then do
          employ the swap operation on the **closest** job pair in $\pi^{k-1}$
       else
          employ the swap operation on the **second closest** job pair in $\pi^{k-1}$
       end if
   end if
  Schedule the jobs at stage k according to the order $\pi^{k-1}$
end for
```

Figure 6. 4. Heuristic Fitness Calculation with Neighbor Swap Moves

Suppose that, the job permutation for stage k is $\pi^{k-1}=\{3, 1, 2, 4, 5\}$ with release times $\{10, 12, 13, 16, 19\}$ from stage $k-1$. In HFR, two jobs are swapped randomly in the current permutation, say jobs 2 and 4. Then, the resulting new permutation will be $\pi^{k-1}=\{3, 1, \mathbf{4}, \mathbf{2}, 5\}$. On the other hand, in HFN, two closest neighbor job pairs are defined for the permutation, based on release times, where the first closest pair is

{1,2} and the second closest pair is {3,1}. Then, one of these pairs is selected according to a given probability np, say the job pair {3,1} is chosen, and the jobs in this pair are swapped. Consequently, the resulting job permutation will be π^{k-1} ={**1**, **3**, 2, 4, 5}.

6.1.3 Solution Representation & Fitness Value Calculation for the EHFSP-V1

As mentioned in Chapter 4, a job-based speed scaling strategy is employed in the EHFSP-V1, where the same speed level is used for a job in all stages. For this purpose, a multi-chromosome structure is used for the proposed bi-objective metaheuristics for the EHFSP-V1, which is composed of a permutation of n jobs (π) and a speed vector of three levels (ψ). Note that, the three speed levels refer to fast, normal and slow speeds, respectively. The solution representation for an individual s is given in Figure 6.5, where $\pi_j \in J$ represents the job at position j and $\psi_j \in L$ represents the speed level for the job at position j.

$\boldsymbol{s(\pi,\psi)}$	$\boldsymbol{\pi}$	π_1	π_2	π_3	π_4	π_5	$\dots$	π_n
	$\boldsymbol{\psi}$	ψ_1	ψ_2	ψ_3	ψ_4	ψ_5	$\dots$	ψ_n

Figure 6. 5. Solution Representation for the EHFSP-V1

Since the standard forward scheduling approach is also very effective, the standard forward scheduling approach is employed in all proposed algorithms for the EHFSP-V1. HFR and HFN approaches are only employed in E_EM_{HFR}, E_EM_{HFN}, $\text{E\_EM}_{\text{HFRN}}$ algorithms in order to further improve the performance of the algorithms, as described in Section 6.3.4.

In the energy-efficient version of the HFSP, there are also speed levels for the jobs. Hence, these speed levels should also be considered in the forward scheduling approach. Namely, processing times of the jobs should be calculated according to their speed levels; i.e., processing times of the jobs should be divided by their corresponding speed factors. Furthermore, when the jobs are sorted based on their release times from the previous stages, the corresponding speed levels of the jobs should also be sorted accordingly. Similarly, in the heuristic fitness calculation approaches (HFR and HFN), when two jobs are swapped with each other, their speed levels should also be swapped.

Finally, the TEC value should be computed for the complete schedule, as explained in Chapter 4 as well as the makespan value.

6.1.4 Solution Representation & Fitness Value Calculation for the EHFSP-V2

As mentioned in Chapter 4, a matrix-based speed scaling strategy is employed in the EHFSP-V2, where the speed of a job can vary from stages to stages. Thus, a multi-chromosome structure is used for the proposed bi-objective metaheuristic algorithms for the EHFSP-V2, which is composed of a permutation of n jobs (π) and a speed matrix of three levels (ψ). The solution representation for an individual s is given in Figure 6.6, where $\pi_j \in J$ represents the job at position j and $\psi_{kj} \in L$ represents the speed level for the operation of the job at position j in stage $k \in M$.

	$\boldsymbol{\pi}$	π_1	π_2	π_3	π_4	π_5	$\cdots$	π_n
$\boldsymbol{s(\pi,\psi)}$	$\boldsymbol{\psi}$	ψ_{11}	ψ_{12}	ψ_{13}	ψ_{14}	ψ_{15}	$\cdots$	ψ_{1n}
		ψ_{21}	ψ_{22}	ψ_{23}	ψ_{24}	ψ_{25}	$\cdots$	ψ_{2n}
		ψ_{31}	ψ_{32}	ψ_{33}	ψ_{34}	ψ_{35}	$\cdots$	ψ_{3n}
		$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\cdots$	$\vdots$
		ψ_{m1}	ψ_{m2}	ψ_{m3}	ψ_{m4}	ψ_{m5}	$\cdots$	ψ_{mn}

Figure 6. 6. Solution Representation for the EHFSP-V2

In the proposed bi-objective metaheuristic algorithms for the EHFSP-V2, the standard forward scheduling approach is employed for the fitness value calculation. HFR and HFN approaches are employed only as a local search to further improve the performance of the algorithms, as described in Section 6.4.5. As mentioned in the previous subsection, in the EHFSP, there are also speed levels for the jobs. Hence, these speed levels should also be considered in the forward scheduling approach and heuristic fitness calculation approaches (HFR and HFN), as explained in Section 6.1.3. Finally, the TEC value should be computed for the complete schedule, as explained in Chapter 4 as well as the makespan value.

6.2 Constructive Heuristic & Single-Objective Algorithms for the HFSP with Makespan Criterion

In the proposed energy-efficient bi-objective metaheuristics for the EHFSP-V1 and EHFSP-V2, single-objective versions of the IG, IG$_{\text{ALL}}$, and VBIH algorithms with only makespan criterion are initially employed to obtain a good starting solution, i.e., a job permutation. Then, the initial population is formed by assigning speed levels to each job of the starting solution. Hence, in this section, the single-objective versions of the IG, IG_{ALL}, and VBIH algorithms with only makespan criterion are explained as well as a proposed constructive heuristic.

6.2.1 Constructive Heuristic for the HFSP with Makespan Criterion

In this thesis, a new constructive heuristic NEH_M(x), i.e., a modified NEH heuristic with x solutions, is proposed for the HFSP with the makespan criterion, modifying the well-known NEH heuristic (Nawaz et al., 1983). The pseudo-code of the NEH heuristic is given in Figure 6.7. Initially, the sum of the processing times on all stages (P_j) is calculated for each job $j \in J$ and jobs are sorted in decreasing order of P_j. Then, the first job in ρ (ρ_1) is chosen to obtain a partial solution with a size one. Consequently, the remaining jobs in ρ are sequentially inserted into the partial solution one by one until a complete solution with n jobs is obtained.

```
$NEH\ Heuristic$
$\forall j \in J, P_j = \sum_{k=1}^{m} p_{kj}$
Step1. $\rho = Sort\ the\ jobs\ in\ decreasing\ order\ of\ P_j$
Step2. $\pi = \{\rho_1\}$
    $for\ (i = 2\ to\ n)\ do$
        $Take\ job\ \rho_i\ from\ \rho$
        $Test\ job\ \rho_i\ in\ all\ of\ the\ i\ possible\ positions\ of\ solution\ \pi$
        $Insert\ job\ \rho_i\ to\ the\ best\ position\ in\ solution\ \pi\ with\ the\ minimum\ fitness\ value$
    $end\ for$
$return\ \pi$
```

Figure 6. 7. NEH Heuristic

The proposed NEH_M(x) procedure is given in Figure 6.8. Similar to the NEH, jobs are initially sorted in decreasing order of P_j to define the initial order (π^0). As the first job has an impact on the waiting time of other jobs and the total idle time, the first job of the partial solution should be defined carefully. In NEH_M(x), x new solutions are generated from the same initial order π^0, by choosing a different job as the first job. Namely, in the h^{th} iteration, the job at position h is chosen as the first job of the initial partial solution, and then, the NEH insertion procedure is applied to this initial partial solution, where $h = 1, \dots, x$. As shown in Figure 6.8, initially, the first job of the initial order (π_1^0) is defined as the first job, and the NEH insertion procedure is applied to generate a new solution (π^1). Next, the second job of the initial order (π_2^0) is defined as the first job, and the NEH insertion procedure is employed to generate another new solution (π^2). This process is repeated x times to obtain x new solutions. Finally, the best one of these x solutions is selected as the final solution. In this thesis, we define $x = n$.

```
$NEH\_M(x)$ $Heuristic$
$\forall j \in J, P_j = \sum_{k=1}^{m} p_{kj}$
Step1. $\pi^0 = Sort\ the\ jobs\ in\ decreasing\ order\ of\ P_j$
Step2. $for\ (h = 1\ to\ x)\ do$
        $\pi' = \pi^0$
        $take\ \pi'_h\ as\ the\ first\ job:$
          $remove\ \pi'_h\ and\ insert\ it\ to\ the\ first\ position\ of\ \pi'$
        $\pi'' = Apply\ NEH\ insertion\ procedure, as\ follows:$
            $\pi'' = \{\pi'_1\}$
            $for\ (i = 2\ to\ n)\ do$
                $Take\ job\ \pi'_i\ from\ \pi'$
                $Test\ job\ \pi'_i\ in\ all\ of\ the\ i\ possible\ positions\ of\ solution\ \pi''$
                $Insert\ job\ \pi'_i\ to\ the\ best\ position\ in\ \pi''$
            $end\ for$
        $\pi^h = \pi''$
        $end\ for$
Step3. $Return\ the\ best\ solution\ among\ \{\pi^1, \pi^2, \dots, \pi^x\}$
```

Figure 6. 8. NEH_M(x) Heuristic

6.2.2 Single-Objective Algorithms with Makespan Criterion

In this section, the single-objective IG, IG_{ALL}, and VBIH algorithms with only makespan minimization are explained. As mentioned before, these algorithms are employed in the proposed bi-objective metaheuristics to obtain a good initial solution.

The IG algorithm is developed by Ruiz and Stützle (2007). The IG algorithm has four main parts: the initial solution, destruction-construction (DC) procedure, local search, and the acceptance criterion. The initial solution is generated by a constructive heuristic. Then, the DC procedure is employed to generate new solutions. The destruction phase removes κ jobs randomly from the current solution. Then, these κ jobs are reinserted into partial solutions in the construction phase. A local search is applied to the complete solution after the DC procedure. Consequently, an acceptance criterion is used to accept the new solution after the local search. These steps are repeated until a stopping criterion is met.

The proposed IG algorithm in this thesis is outlined in Figure 6.9, where $rand$ is a uniform random number between 0 and 1. As shown in Figure 6.9, NEH_M(x) is employed as a constructive heuristic in the proposed IG algorithm. Note that, the NEH heuristic is used for this purpose in the original IG algorithm (Ruiz and Stützle, 2007). Then, in the destruction step, κ jobs are randomly chosen and removed from the solution π^0. This procedure results in two partial solutions: the partial solution π^c with n - κ jobs and the partial solution π^d with κ jobs. Note that, π^d includes the jobs that will be reinserted into π^c, in the order in which they were removed from π^0. The construction process begins with a partial solution π^c and applies κ steps, in which the jobs in π^d are reinserted into π^c. That is, it starts with π^c and inserts the first job of $\pi^d(\pi_1^d)$ into all possible $n - \kappa + 1$ positions of π^c. Then, the best position for π_1^d, which has the minimum makespan, is chosen and π_1^d is inserted in that position of π^c. These steps are repeated for all jobs in π^d until π^d is empty. The complete solution is obtained once the last removed job is inserted into n positions, and the best insertion is chosen.

```
$Procedure\ IG\ Algorithm(\kappa, \tau P)$
$\pi^0 = NEH\_M(n),\ \pi^{best} = \pi^0$
$while\ (time\ limit\ is\ not\ exceeded)\ do$
 $\pi^d, \pi^c = Destruction\ (\pi^0, \kappa)$
 $\pi^1 = Construction\ (\pi^d, \pi^c)$
 $\pi^2 = First - Improvement\ Insertion\ Neighborhood\ (\pi^1)$  %local search to the complete solution
   $if\ f(\pi^2) < f(\pi^0)\ then$
        $\pi^0 = \pi^2$
        $if\ f(\pi^2) < f(\pi^{best})\ then$
          $\pi^{best} = \pi^2$
        $end\ if$
   $else\ if\ (rand < exp\{-(f(\pi^2) - f(\pi^0))/T\})\ then$
        $\pi^0 = \pi^2$
   $end\ if$
$end\ while$
$return\ \pi^{best}\ and\ f(\pi^{best})$
```

Figure 6. 9. IG Algorithm

The first-improvement insertion neighborhood structure is used as a local search, after the construction phase of the IG algorithm. As shown in Figure 6.10, job π_k at position k is randomly chosen from the current solution π without repetition and inserted into all possible positions of the solution. When the best insertion position is found by improving the makespan, the job π_k is inserted into that position. This procedure is repeated for all jobs. This insertion local search procedure is named as *first-improvement* insertion neighborhood structure since it employs a first-improvement type pivoting rule for updating the solution. According to this rule, after inserting the removed job into the best position of the current solution that leads to the smallest makespan, the current solution is replaced with the new one if there is an improvement, without testing the other jobs at all possible positions. Then, this updated solution is used for the next iteration to test another job. Namely, at any iteration, after performing the best insertion for the chosen job, the current solution is updated if there is an improvement, before testing the other jobs at all positions.

```
$First - Improvement\ Insertion\ Neighborhood\ (\pi)$
$for\ i = 1\ to\ n\ do$
  $remove\ job\ \pi_k\ from\ solution\ \pi\ randomly\ (without\ repetition)$
  $\pi^* = Insert\ job\ \pi_k\ in\ best\ position\ of\ \pi$
  $if\ \big(f(\pi^*) < f(\pi)\big)\ then\ do$
      $\pi = \pi^*$
  $end\ if$
$end\ for$
$return\ \pi\ and\ f(\pi)$
```

Figure 6. 10. First-Improvement Insertion Neighborhood Structure

After the local search step, it is decided whether to accept the new solution as the incumbent solution for the next iteration or not. If the new solution is better than the current one, the IG algorithm always accepts the new solution as the incumbent solution. However, if the new solution is worse than the current one, it accepts the new solution with a probability. For this purpose, a simple simulated annealing-type acceptance criterion is used with a constant temperature, which is suggested by Osman and Potts (1989):

$$T = \frac{\sum_{j=1}^{n}\sum_{k=1}^{m} p_{kj}}{10nm} \times \tau P, \tag{6-1}$$

where n is the number of jobs, m is the number of stages and τP is a parameter to be adjusted.

Recently, a new version of the IG algorithm, namely IG_{ALL}, is presented in the literature for the permutation flowshop with the makespan criterion (Dubois-Lacoste et al., 2017). The difference between the two algorithms is that IG_{ALL} applies an additional local search to partial solutions after the destruction in order to enhance solution quality. The proposed IG_{ALL} algorithm outlined in Figure 6.11 employs

NEH_M(x) as a constructive heuristic. Note that, IG$_{\text{ALL}}$ applies the first-improvement neighborhood structure to the partial solutions with n - κ jobs before the construction phase, and if any improvement has been found, the local search is applied again until a local optimum is obtained. The rest of the procedure is the same as the aforementioned IG algorithm.

```
$Procedure\ IG_{ALL}\ Algorithm(\kappa, \tau P)$
$\pi^0 = NEH\_M(n),\ \pi^{best} = \pi^0$
$while\ (time\ limit\ is\ not\ exceeded)\ do$
 $\pi^d, \pi^c = Destruction\ (\pi^0, \kappa)$
 $\pi^c = First - Improvement\ Insertion\ Neighborhood\ (\pi^c)$ %local search to the partial solution
 $\pi^1 = Construction\ (\pi^d, \pi^c)$
 $\pi^2 = First - Improvement\ Insertion\ Neighborhood\ (\pi^1)$ %local search to the complete solution
    $if\ f(\pi^2) < f(\pi^0)\ then$
          $\pi^0 = \pi^2$
          $if\ f(\pi^2) < f(\pi^{best})\ then$
             $\pi^{best} = \pi^2$
          $end\ if$
   $else\ if\ (rand < exp\{-(f(\pi^2) - f(\pi^0))/T\})\ then$
          $\pi^0 = \pi^2$
   $end\ if$
$end\ while$
$return\ \pi^{best}\ and\ f(\pi^{best})$
```

Figure 6. 11. IG$_{\text{ALL}}$ Algorithm

A block move is proposed by Xu et al. (2014), where a block with bs consecutive jobs is removed from the solution and inserted into another position. Based on this idea, VBIH algorithms are proposed by Tasgetiren et al. (2016, 2017), where the block size bs changes during the procedure. In this thesis, a similar VBIH algorithm is developed, as outlined in Figure 6.12. Note that NEH_M(x) heuristic is used in the construction of the initial solution, as in the proposed IG algorithms. As shown in Figure 6.12, a block of jobs with size bs is initially removed from the current solution, where bs is set in between a minimum block size (bs_{min}) and a maximum block size (bs_{max}). Then, as in IG$_{\text{ALL}}$, the first-improvement neighborhood structure is employed on the partial solution with n - bs jobs; and if any improvement has been found, the local search is applied again until a local optimum is reached. Afterward, the best block insertion is performed by testing the removed block in all possible positions of the partial solution. Then, the first-improvement neighborhood structure is employed on the obtained complete solution. Finally, the acceptance criterion given in Eq. (6-1) is employed. This process is iterated until the bs attains the bs_{max}.

```
$Procedure\ VBIH\ (bs_{max}, \tau P)$
$\pi^0 = NEH\_M(n), \pi^{best} = \pi^0,\ bs_{min} = 2$
$while\ (time\ limit\ is\ not\ exceeded)\ do$
$bs = bs_{min}$
$do\{$
 $\pi^1 = Remove\ a\ block\ with\ size\ bs\ from\ \pi^0$
 $\pi^2 = First - Improvement\ Insertion\ Neighborhood\ (\pi^1)$    %local search to the partial solution
 $\pi^3 = Insert\ the\ block\ into\ the\ best\ position\ in\ \pi^2$
 $\pi^4 = First - Improvement\ Insertion\ Neighborhood\ (\pi^3)$   %local search to the complete solution
        $if\ (f(\pi^4) < f(\pi^0))\ then\ do$
              $\pi^0 = \pi^4$
              $if\ (f(\pi^4) < f(\pi^{best}))\ then\ do$
                  $\pi^{best} = \pi^4$
              $endif$
        $else$
              $if\ (rand < exp\{-(f(\pi^4) - f(\pi^0))/T\})$
                  $\pi^0 = \pi^4$
              $endif$
        $endif$
        $bs = bs + 1$
$\}while(bs \leq bs_{max})$
$endwhile$
```

Figure 6. 12. Variable Block Insertion Algorithm

6.3 Energy-efficient Bi-Objective Metaheuristic Algorithms for the EHFSP-V1

In this section, the proposed seven energy-efficient bi-objective metaheuristic algorithms; namely, two variants of the IG algorithm (E_IG, E_IG$_{\text{ALL}}$), a VBIH algorithm (E_VBIH) and four variants of the ensemble of metaheuristic algorithms (E_EM, E_EM$_{\text{HFR}}$, E_EM$_{\text{HFN}}$, E_EM$_{\text{HFRN}}$) are explained for the EHFSP-V1.

6.3.1 Initial Population

As mentioned in Section 6.2, in the proposed energy-efficient bi-objective metaheuristics for the EHFSP-V1, single-objective versions of the IG, IG$_{\text{ALL}}$, and VBIH algorithms with only makespan criterion are initially employed to obtain a good starting solution, i.e., a job permutation. Then, the initial population is formed by assigning a random speed level to each job of the starting solution. Namely, the initial population with size PS is constructed as follows: Firstly, an initial solution is obtained by the NEH_M(x) constructive heuristic, which is explained in Section 6.2.1. Then, the resulting solution is taken as the initial solution for the single-objective IG, IG$_{\text{ALL}}$, or VBIH algorithms for the makespan minimization, which is explained in Section 6.2.2. In order to start with a good job permutation, 25% of the total CPU time is devoted to the following: the single-objective IG in E_IG, E_EM, E_EM$_{\mathrm{HFR}}$, E_EM$_{\mathrm{HFN}}$, E_EM$_{\mathrm{HFRN}}$ algorithms; the single-objective IG$_{\mathrm{ALL}}$ in E_IG$_{\mathrm{ALL}}$ algorithm; and the single-objective VBIH in the E_VBIH algorithm. Once the best solution π^{best} is found by one of these single-objective algorithms, the first three individuals in the population are generated by assigning fast, normal and slow speed levels to all jobs in π^{best}. The rest of the individuals in the population are constructed by assigning a random speed level to each job in π^{best}. The archive set (AS) is initially empty and filled with non-dominated solutions from the initial population.

6.3.2 E_IG and E_IG$_{\mathrm{ALL}}$ Algorithms

As mentioned in Section 6.3.1, the initial population is formed in E_IG using the good starting solution found by the single-objective IG algorithm with only makespan criterion, whereas the single-objective IG$_{\mathrm{ALL}}$ algorithm with makespan criterion is used for the initial population generation in E_IG$_{\mathrm{ALL}}$ algorithm. After the generation of the initial population, in E_IG and E_IG$_{\mathrm{ALL}}$ algorithms, destruction-construction and local search procedures are applied to each individual in the population, while the time limit is not exceeded. Note that, these procedures are similar to the procedures in single-objective IG algorithms; except that, in E_IG and E_IG$_{\mathrm{ALL}}$ algorithms, speed levels are also regarded and the solutions are assessed according to the dominance rules due to the bi-objective nature of the EHFSP. Namely, in the destruction step, κ jobs are randomly removed from the solution as well as their speed levels, and random speed levels are assigned to these removed κ jobs. Before the construction, in the E_IG$_{\mathrm{ALL}}$ algorithm, the energy-efficient first-improvement insertion neighborhood structure (Figure 6.13) is applied to the partial solution regarding the speed levels, while it is not applied in the E_IG algorithm. Then, these κ jobs are reinserted into the partial solution with their respective speed levels following the best insertion policy, in the order they were removed, until a complete solution of n jobs is established. As the problem is bi-objective, the dominance rule $(\prec)$ is used when two solutions are compared, where the partial solutions are assessed based on the partial dominance rule.

After the destruction-construction procedures, the energy-efficient first-improvement insertion neighborhood structure is applied to the complete solution, as shown in

Figure 6.13. Job π_i and speed ψ_i are removed from position i of solution s, and a new speed level is randomly assigned to this job. Then, the local search inserts this job-speed pair (π_i, ψ_i) into all possible positions of the incumbent solution $s(\pi, \psi)$. After the best insertion position is found, (π_i, ψ_i) is inserted into that position. If the new solution s^* dominates the incumbent solution s, then the current solution is updated. This is repeated for all job-speed pairs. In the case of an improving solution, the local search is repeated until no more improving solutions can be obtained. Note that, the archive set AS is also updated during the procedure, whenever a new non-dominated solution is found.

```
$improve = true$
$while\ (improve = true)\ do$
 $improve = false$
 $for\ k = 1\ to\ n\ do$
  $(\pi_i, \psi_i) =$
  $Remove\ job\ \pi_i\ and\ its\ speed\ level\ \psi_i\ from\ solution\ s(\pi, \psi)\ randomly(without\ repetition)$
  $\psi_i = assign\ a\ random\ speed\ level\ from\ \{1,2,3\}$
  $s^*(\pi^*, \psi^*) = Insert\ job\ (\pi_i, \psi_i)\ into\ best\ position\ of s(\pi, \psi)$
  $if\ (s^* \prec s)\ then\ do$
     $s = s^*$
     $improve = true$
  $end\ if$
 $end\ for$
$end\ while$
```

Figure 6. 13. Energy-Efficient First-Improvement Insertion Neighborhood Structure for the EHFSP-V1

6.3.3 E_VBIH Algorithm

As mentioned in Section 6.3.1, the initial population is generated for the E_VBIH algorithm using the good starting solution found by the single-objective VBIH algorithm with only a makespan criterion. After the generation of the initial population, in the E_VBIH algorithm, block insertion and local search procedures are applied to each individual in the population, while the time limit is not exceeded. Note that these procedures are similar to the procedures in the single-objective VBIH algorithm, except that, in the E_VBIH algorithm, speed levels are also considered and the solutions are assessed according to the dominance rules. That is, a block of jobs with size bs is randomly removed from the solution together with their speeds, where bs is in between a minimum block size $(bs_{min} = 2)$ and a maximum block size (bs_{max}). Random speed levels are assigned to these removed jobs. Then, the energy-efficient first-improvement insertion neighborhood structure (Figure 6.13) is applied to the partial solution considering the speed levels. Afterward, the block of jobs with size bs is inserted into all possible positions of the partial solution with their respective speeds, and the best block insertion is realized.

After the block insertion procedure, the energy-efficient first-improvement insertion neighborhood structure is applied to the complete solution, as shown in Figure 6.13. The archive set AS is also updated during this procedure, in the case of a new non-dominated solution is found. Note that, as in E_IG algorithms, the dominance rule is used to evaluate solutions, where the partial solutions are evaluated based on the partial dominance rule. This process is repeated until the bs reaches the bs_{max}.

6.3.4 Ensemble of Metaheuristic Algorithms ($\mathbf{E\_EM}$, $\mathbf{E\_EM_{HFR}}$, $\mathbf{E\_EM_{HFN}}$, $\mathbf{E\_EM_{HFRN}}$)

In order to improve solution quality, the aforementioned E_IG, E_IG$_{\text{ALL}}$ and E_VBIH algorithms are combined in an energy-efficient ensemble of metaheuristic algorithms (E_EM). Note that, the ensemble idea in heuristic optimization, which combines several heuristic procedures effectively, is initially presented in (Mallipedi et al., 2011; Tasgetiren et al., 2010; Mallipedi and Suganthan, 2010). After the generation of the initial population, in the E_EM algorithm, a random algorithm strategy is assigned to each individual in the population. In this way, a different algorithm is applied to each individual according to the assigned strategy, while the time limit is not exceeded. There are four algorithmic strategies in the E_EM algorithm: E_IG algorithm, E_IG$_{\text{ALL}}$ algorithm, E_VBIH algorithm, and crossover local search. Note that, the crossover local search is also included in the E_EM algorithm as one of the strategies since it enhances the solution quality (Öztop et al., 2018; Tasgetiren et al., 2018b).

The crossover local search initially employs a uniform crossover operator only on speed levels while keeping the permutation the same. For an individual s_p in the population, another individual s_q is selected from the population randomly. Then, a new solution is generated by taking the speed levels either from s_p or s_q, depending on the crossover probability CR. Namely, for each position j in s_p, a uniform random number r_j is generated between 0 and 1, and if $r_j < CR$, speed level at position j in s_p is kept the same. In the case of $r_j \geq CR$, the speed level at position j in s_p is replaced by the speed level at the same position in individual s_q. Note that, the crossover probability CR is drawn from the uniform distribution between 0.4 and 0.6. After the crossover, the energy-efficient first-improvement insertion neighborhood structure (Figure 6.13) is applied to the solution on hand. The archive set AS is updated during the procedure, whenever a new non-dominated solution is found.

In order to enhance the performance of the E_EM algorithm and to avoid the shortcomings of the standard forward scheduling approach as mentioned in Section 6.1.2, HFR and HFN heuristic fitness calculation approaches are also employed on each individual in the population. Namely, the complete schedule is re-computed for each individual in the population by employing HFR and HFN approaches, in order to explore the neighboring schedules. As mentioned in Section 6.1.2, HFR and HFN approaches employ swap moves on the job permutations of some stages, during forward scheduling procedure. Note that, employing a swap move on the job permutation of a stage can lead to a different complete schedule with different fitness function values.

Consequently, after implementing the E_EM algorithm, the HFR approach is employed on each individual in the population in the E_EM$_{\text{HFR}}$ algorithm, while the HFN approach is employed in E_EM$_{\text{HFN}}$ algorithm. Furthermore, as a fourth variant of the E_EM algorithm, both HFR and HFN approaches are employed with an equal probability in the E_EM$_{\text{HFRN}}$ algorithm. Note that, HFR and HFN approaches are applied $\lfloor n/2 \rfloor$ times to each individual in E_EM$_{\text{HFR}}$, E_EM$_{\text{HFN}}$ and E_EM$_{\text{HFRN}}$ algorithms. As mentioned in Section 6.1.3, speed levels of the jobs are also regarded in the heuristic fitness calculation approaches. The archive set AS is updated during the procedure if a new non-dominated solution is found.

The general outline of the energy-efficient ensemble of metaheuristic algorithms (E_EM, E_EM$_{\text{HFRN}}$, E_EM$_{\text{HFR}}$, E_EM$_{\text{HFN}}$) is provided in Figure 6.14, where r is a uniform random number between 0 and 1. Note that, as mentioned in Section 6.3.1, the initial population is generated for the ensemble of metaheuristic algorithms using the good starting solution found by the single-objective IG algorithm with only the makespan criterion.

```
Procedure $E\_EM, E\_EM_{HFR}, E\_EM_{HFN}, E\_EM_{HFRN}$
Generate the initial population with size $PS$:
  • Apply the single objective IG algorithm using $NEH\_M(x)$ constructive heuristic
    for 25% of the total time budget to find $\pi^{best}$
  • Form the initial population using $\pi^{best}$
for $p = 1$ to $PS$
    Assign a random algorithm strategy $str_p$ to individual $s_p$
end for
while (time limit is not exceeded) do
for $p = 1$ to $PS$
if $(str_p = 1)$ Apply E_IG to individual $s_p$ (Destruction-Construction & Local Search)
else if $(str_p = 2)$ Apply $E\_IG_{ALL}$ to individual $s_p$ (Destruction-Construction & Local Search)
else if $(str_p = 3)$ Apply E_VBIH to individual $s_p$ (Variable Block Insertion & Local Search)
else if $(str_p = 4)$ Apply crossover local search to individual $s_p$
end if
    % HFR & HFN approaches (only in $E\_EM_{HFR}, E\_EM_{HFN}, E\_EM_{HFRN}$ algorithms)
    for $k = 1$ to $\lfloor n/2 \rfloor$
        ($\mathbf{E\_EM_{HFR}}$): recompute the complete schedule for individual $s_p$ using HFR approach
        ($\mathbf{E\_EM_{HFN}}$): recompute the complete schedule for individual $s_p$ using HFN approach
        ($\mathbf{E\_EM_{HFRN}}$):
            if $(r \sim U(0,1) \leq 0.5)$
            recompute the complete schedule for individual $s_p$ using HFR approach
            else
            recompute the complete schedule for individual $s_p$ using HFN approach
            end if
    end for
end for
end while
```

Figure 6. 14. General Outline of the Energy-Efficient Ensemble of Metaheuristic Algorithms for the EHFSP-V1

6.4 Energy-efficient Bi-Objective Metaheuristic Algorithms for the EHFSP-V2

In this section, the proposed four energy-efficient bi-objective metaheuristic algorithms; namely, two variants of the IG algorithm (E_IG2, E_IG2$_{\text{ALL}}$), a VBIH algorithm (E_VBIH2), and an ensemble of metaheuristic algorithms (E_EM2) are explained for the EHFSP-V2.

6.4.1 Initial Population

In the proposed bi-objective metaheuristic algorithms for the EHFSP-V2, the initial population with size PS is constructed as follows: Firstly, an initial solution is obtained by the NEH_M(x) constructive heuristic, which is explained in Section 6.2.1. Then, the resulting solution is taken as the initial solution for the single-objective IG, IG_{ALL} or VBIH algorithms for makespan minimization, which is explained in Section 6.2.2. In order to start with a good job permutation, $Inp\%$ of the total CPU time is devoted to the following: the single-objective IG in E_IG2 and E_EM2 algorithms; the single-objective IG_{ALL} in the E_IG2_{ALL} algorithm; and the single-objective VBIH in the E_VBIH2 algorithm, where $Inp\%$ is 20% for large instances and $Inp\%$ is 10% for small and medium instances.

Once the best solution π^{best} is found by one of these single-objective algorithms, the first individual in the population is generated by assigning fast speed levels to all operations of the jobs in π^{best}. The rest of the individuals in the population are constructed by assigning slow and normal speed levels randomly to the operations of jobs in π^{best}. The archive set (AS) is initially empty and filled with non-dominated solutions from the initial population. In order to start with more energy-efficient solutions, a mutation strategy is also applied to each individual in the population. Namely, for each individual $s_p(\pi, \psi) \in PS$, we mutate the speed level ψ_{kj} of each operation of a job by assigning a slow or normal speed level randomly. Then, the archive set AS is updated after mutating the speed level ψ_{kj} of each operation of a job, in the case of a new non-dominated solution is found.

6.4.2 E_IG2 and E_IG2_{ALL} Algorithms

E_IG2 and E_IG2_{ALL} algorithms are very similar to the E_IG and E_IG_{ALL} algorithms in Section 6.3.2. Namely, E_IG2 is the extended version of the E_IG and E_IG2_{ALL} is the extended version of the E_IG_{ALL} to the matrix-based speed scaling strategy. Similar to the E_IG and E_IG_{ALL} algorithms, after the generation of the initial population; destruction, construction and local search procedures are applied to each individual in E_IG2 and E_IG2_{ALL} algorithms. However, when applying these procedures, a job is removed/inserted from/into the solution as well as its speed column (speed levels of its all operations) in the E_IG2 and E_IG2_{ALL} algorithms due to the matrix-based speed scaling structure. Additionally, when changing the speed levels of a removed job, random speed levels are assigned to all operations. Namely, in the destruction phase, κ jobs are randomly removed from the solution as well as their speed levels, and random speed levels are assigned to the operations of these removed κ jobs. Before the construction, in the E_IG2_{ALL} algorithm, the energy-efficient first-improvement insertion neighborhood structure (Figure 6.15) is applied to the partial solution, while it is not applied in the E_IG2 algorithm. Then, these κ jobs are reinserted into the partial solution with their respective speed levels following the best insertion policy, in the order of removal.

```
$improve = true$
$while\ (improve = true)\ do$
 $improve = false$
 $for\ q = 1\ to\ n\ do$
 $(\pi_i, \overrightarrow{\psi_i}) =$
 $Remove\ job\ \pi_i\ and\ its\ speed\ column\ \overrightarrow{\psi_i}\ from\ solution\ s(\pi, \psi)\ randomly(without\ repetition)$
   $for\ k = 1\ to\ m\ do$
     $\overrightarrow{\psi_i} = assign\ a\ random\ speed\ level\ from\ \{1,2,3\}\ to\ each\ operation\ k\ of\ job\ \pi_i$
   $end\ for$
   $s^*(\pi^*, \psi^*) = Insert\ job\ (\pi_i, \overrightarrow{\psi_i})\ into\ best\ position\ of s(\pi, \psi)$
   $if\ (s^* \prec s)\ then\ do$
      $s = s^*$
      $improve = true$
   $end\ if$
 $end\ for$
$end\ while$
```

Figure 6. 15. Energy-Efficient First-Improvement Insertion Neighborhood Structure for the EHFSP-V2

After destruction-construction procedures, the energy-efficient first-improvement insertion neighborhood structure is applied to the complete solution, as shown in Figure 6.15. Note that, this local search (Figure 6.15) is the extended version of the local search in Figure 6.13 to the matrix-based speed scaling strategy. Namely, job π_i and its speed column $\overrightarrow{\psi_i}$ (i.e., speed levels of all operations of the job π_i) are removed from position i of solution s, and new speed levels are randomly assigned to the operations of this removed job. Then, the local search inserts this job-speed column pair $(\pi_i, \overrightarrow{\psi_i})$ into all possible positions of the incumbent solution $s(\pi, \psi)$, and chooses the best insertion. If the new solution s^* dominates the incumbent solution s, then the current solution is updated. This is repeated for all job-speed column pairs. In the case of an improving solution, the local search is restarted until no more improving solutions can be obtained. The archive set AS is also updated during the procedure, whenever a new non-dominated solution is found.

Unlike the E_IG and E_IG$_{\text{ALL}}$ algorithms, after implementing the aforementioned E_IG2 and E_IG2$_{\text{ALL}}$ procedures, a mutation strategy and heuristic fitness calculation approaches are also employed on each individual, as explained in Section 6.4.5.

6.4.3 E_VBIH2 Algorithm

E_VBIH2 algorithm is very similar to the E_VBIH algorithm in Section 6.3.3. Namely, E_VBIH2 is the extended version of the E_VBIH algorithm to the matrix-based speed scaling strategy. Similar to the E_VBIH algorithm, after the generation of the initial population, block insertion and local search procedures are applied to each individual in the E_VBIH2 algorithm. However, when applying these procedures, a job is removed/inserted from/into the solution as well as its speed column (speed levels of its all operations) due to the matrix-based speed scaling structure. Furthermore, when changing the speed levels of a removed job, random speed levels are assigned to its all operations. Namely, a block of jobs with size bs is randomly removed from the solution together with their speed levels, where $bs_{min} \leq bs \leq bs_{max}$, and random speed levels are assigned to the operations of these removed jobs. Note that, we set bs_{min}= 2. Then, the energy-efficient first-improvement insertion neighborhood structure (Figure 6.15) is applied to the partial solution. Afterward, the removed block of jobs is inserted into all possible positions of the partial solution with their respective speeds, and the best block insertion is realized. After block insertion procedure, the energy-efficient first-improvement insertion neighborhood structure is applied to the complete solution, as shown in Figure 6.15. The archive set AS is also updated during this procedure, in the case of a new non-dominated solution is found. This process is repeated until the bs reaches the bs_{max}.

Unlike the E_VBIH algorithm, after implementing the aforementioned E_VBIH2 procedures, a mutation strategy and heuristic fitness calculation approaches are also employed on each individual, as explained in Section 6.4.5.

6.4.4. Ensemble of Metaheuristic Algorithms (E_EM2)

As a fourth algorithm, the aforementioned E_IG2, E_IG2$_{\text{ALL}}$ and E_VBIH2 algorithms are combined in an energy-efficient ensemble of metaheuristic algorithms (E_EM2). After the generation of the initial population, in the E_EM2 algorithm, a random algorithm strategy is assigned to each individual in the population. There are three algorithmic strategies in the E_EM2 algorithm: E_IG2 algorithm, E_IG2$_{\text{ALL}}$ algorithm and E_VBIH2 algorithm. In this way, a different algorithm is applied to each individual according to the assigned strategy similar to the E_EM algorithm described in Section 6.3.4. The general outline of the energy-efficient ensemble of metaheuristic algorithms (E_EM2) is provided in Figure 6.16, where r is a uniform random number between 0 and 1. As shown in Figure 6.16, after implementing the E_EM2 procedures, a mutation strategy and heuristic fitness calculation approaches are also employed on each individual, as explained in Section 6.4.5.

```
Procedure E_EM2
Generate the initial population with size PS:
  • Apply the single objective IG algorithm using NEH_M(x) constructive heuristic
    for Inp% of the total time budget to find $\pi^{best}$
  • Form the initial population using $\pi^{best}$
  • Apply a mutation strategy on speed levels of the individuals in the initial population
for $p = 1$ to PS
    Assign a random algorithm strategy $str_p$ to individual $s_p$
end for
while (time limit is not exceeded) do
for $p = 1$ to PS
if $(str_p = 1)$ Apply E_IG2 to individual $s_p$ (Destruction-Construction & Local Search)
else if $(str_p = 2)$ Apply E_IG2$_{ALL}$ to individual $s_p$ (Destruction-Construction & Local Search)
else if $(str_p = 3)$ Apply E_VBIH2 to individual $s_p$ (Variable Block Insertion & Local Search)
end if
    Apply a mutation strategy on the speed levels of the individual $s_p$
    for $k = 1$ to $n$
            if $(r \sim U(0,1) \leq 0.5)$
            recompute the complete schedule for individual $s_p$ using HFR approach
            else
            recompute the complete schedule for individual $s_p$ using HFN approach
            end if
    end for
end for
end while
```

Figure 6. 16. General Outline of the Energy-Efficient Ensemble of Metaheuristic Algorithms for the EHFSP-V2

6.4.5. Heuristic Fitness Calculation & Mutation Operators

As mentioned in Section 6.1.4, the standard forward scheduling approach is employed in all proposed bi-objective metaheuristic algorithms for the EHFSP-V2, since the standard forward scheduling approach is very effective. HFR and HFN approaches are employed only as a local search to further improve the performance of the algorithms.

So as to obtain more energy-efficient solutions, a mutation strategy is also applied to each individual in the population, after the aforementioned E_IG2, E_IG2$_{\mathrm{ALL}}$, E_VBIH2, and E_EM2 algorithms. Namely, for each individual, we mutate the speed level ψ_{kj} of each operation of a job by assigning a slow or normal speed level randomly. Then, the archive set AS is updated after mutating the speed level ψ_{kj} of each operation of a job, in the case of a new non-dominated solution is found.

In order to enhance the performance of the aforementioned E_IG2, E_IG2$_{\mathrm{ALL}}$, E_VBIH2 and E_EM2 algorithms further and to avoid the shortcomings of the standard forward scheduling approach as mentioned in Section 6.1.2, HFR and HFN heuristic fitness calculation approaches are also employed on each individual in the population, as a local search. Namely, the complete schedule is re-computed for each individual in the population by employing HFR and HFN approaches, in order to explore the neighboring schedules. As mentioned in Section 6.1.2, HFR and HFN approaches employ swap moves on the job permutations of some stages, during the forward scheduling procedure. Consequently, after implementing the E_IG2, E_IG2$_{\mathrm{ALL}}$, E_VBIH2 or E_EM2 algorithm, both HFR and HFN approaches are employed on each individual with an equal probability. Note that, HFR and HFN approaches are applied *n* times to each individual. As mentioned in Section 6.1.4, speed levels of the jobs are also regarded in the heuristic fitness calculation approaches. The archive set AS is updated during the procedure if a new non-dominated solution is found.

6.5 Archive Set Update Procedure

In all aforementioned energy efficient bi-objective metaheuristic algorithms for both EHFSP-V1 and EHFSP-V2, an archive set AS is used to store non-dominated solutions. When a new non-dominated solution found, it is included in the archive set AS and any member dominated by the new non-dominated solution is removed. In order to update the archive set AS, an effective update procedure is employed, which is proposed by Pan et al. (2009), as shown in Figure 6.17.

For including a new non-dominated solution *x* to the archive set *AS*, a straightforward way is to compare it with each solution in the *AS* to check whether it is dominated by any solution in the *AS*. Since this straightforward method makes comparisons with all solutions in the set, it requires a high computational time. However, the update procedure of Pan et al. (2009) employs a faster update procedure without evaluating all comparisons, using the storage structure of *AS*. Basically, it finds the most suitable position for the new solution x to be inserted in a smarter way regarding the storage structure of the *AS*. Note that, the non-dominated solutions in AS are stored in increasing order of their first objective function values, where their second objective function values will be in decreasing order.

As shown in Figure 6.17, when comparing $f_1(x)$ with $f_1(\vartheta_j)$, following cases may arise:

Case 1. If $f_1(x) = f_1(\vartheta_j)$ and if $f_2(x) < f_2(\vartheta_j)$, x is a new non-dominated solution and it dominates the solution ϑ_j. Therefore, *x* should be inserted to position *j* (*pos*=*j*) and ϑ_j should be replaced by *x*. Otherwise, if $f_1(x) = f_1(\vartheta_j)$ and if $f_2(x) \geq f_2(\vartheta_j)$, *x* is dominated by ϑ_j or has the same objective function values as ϑ_j.

Case 2. If $f_1(x) < f_1(\vartheta_j)$:

If $j = 1$, *x* is a new non-dominated solution; it should be inserted to the first position (*pos*=*j*) in *AS* and the archive size *u* should be incremented by one.

If $j > 1$ and if $f_2(x) < f_2(\vartheta_{j-1})$, x is a new non-dominated solution; it should be inserted to the position *j* (*pos*=*j*) and the archive size *u* should be incremented by one. Otherwise, if $j > 1$ and if $f_2(x) \geq f_2(\vartheta_{j-1})$, *x* is dominated by ϑ_{j-1}.

Case 3. If $f_1(x) > f_1(\vartheta_j)$ and if $f_2(x) < f_2(\vartheta_j)$, *x* is a new non-dominated solution; it should be inserted to the position *j+1* (*pos*=*j+1*) and the archive size *u* should be incremented by one. Otherwise, if $f_1(x) > f_1(\vartheta_j)$ and if $f_2(x) \geq f_2(\vartheta_j)$, x is dominated by ϑ_j.

If any of the above cases is met, the solution x is added to position pos and all solutions dominated by x in AS are removed as explained in Figure 6.17.

1. Archive size is $u = |AS|$ and $AS = \{\vartheta_1, \vartheta_2, .., \vartheta_u\}$. Initially, AS is empty and the first non-dominated solution x will be added to the first position in AS. Let $k = 1$.
2. Find a most suitable position pos for the next individual x in the archive set AS as follows:

```
$do\{$
   $j = \lfloor (k+u)/2 \rfloor$
   $if\ \left(f_1(x) = f_1(\vartheta_j)\right)\ then\ j = j,\ break$
   $elseif\ \left(f_1(x) < f_1(\vartheta_j)\right)\ then\ u = j - 1$
   $else\ \ k = j + 1$
$while(k \le u)$
```

3. When comparing $f_1(x)$ with $f_1(\vartheta_j)$, following cases may arise:

```
$Case\ 1.\ if\ \left(f_1(x) = f_1(\vartheta_j)\right)\ and\ if\ \left(f_2(x) < f_2(\vartheta_j)\right) then\ pos = j$
$Case\ 2.\ if\ \left(f_1(x) < f_1(\vartheta_j)\right)$
      $if\ \ j = 1\ \ then\ pos = j\ and\ u = u + 1$
      $if\ \ j > 1\ and\ \ if\ \left(f_2(x) < f_2(\vartheta_{j-1})\right) then\ pos = j\ and\ u = u + 1$
$Case\ 3.\ if\ \left(f_1(x) > f_1(\vartheta_j)\right)\ and\ if\ \left(f_2(x) < f_2(\vartheta_j)\right) then\ pos = j + 1\ and\ u = u + 1$
```

If any of the above cases is met, the solution x is added to position pos and all solutions dominated by x in AS are removed. The below procedure removes the dominated solutions from AS:

```
Step 1.  $If\ (pos = u)\ then\ go\ to\ Step\ 4$
Step 2.  $Let\ pos = pos + 1.\ If\ f_2(\vartheta_{pos}) \ge f_2(x)\ then\ remove\ \vartheta_{pos}; otherwise\ go\ to\ Step\ 4$
Step 3.  $if(pos < u)\ then\ go\ to\ Step\ 2$
Step 4.  $AS = report\ non - dominated\ solutions$
```

Figure 6. 17. Archive Set Update Procedure

CHAPTER 7
COMPUTATIONAL RESULTS

For evaluating the performance of the proposed solution approaches, the well-known HFSP benchmark set (Carlier and Neron, 2000; Liao et al., 2012; Öztop et al., 2019) is modified, where each instance is represented by the number of jobs, the number of stages and the machine layout at the stages.

The machine layout is defined by the letters *a*, *b*, *c*, and *d*:

a: There is a single machine in the middle stage, and there are three machines in other stages.

b: There is a single machine in the first stage, and there are 3 machines in other stages.

c: There are two machines in the middle stage, and three machines in other stages.

d: There are three machines in all stages.

For example, the notation *j*10*c*5*b*1 indicates a problem with 10 jobs (*j*10), 5 stages (*c*5) and *b* type layout, where the last number 1 is the problem index for a specific type. Originally, there are 77 instances in the benchmark set (Carlier and Neron, 2000), where the number of jobs is either 10 or 15 and the number of stages is either 5 or 10. Furthermore, additional 40 large instances with 30, 40, 50 and 60 jobs and 5 stages are proposed in (Liao et al., 2012; Öztop et al., 2019), where the number of machines in each stage is randomly generated between 3 and 5, and the processing times of jobs are uniform in the range [1,100]. The instances for the bi-objective EHFSP are generated by adding energy-related parameters to these benchmarks. Due to the computationally challenging nature of the bi-objective EHFSP, additional small instances with 5 jobs & 5 stages are also created by truncating the instances with 10 jobs & 5 stages and 15 jobs & 5 stages. Consequently, 47 small instances (*j*5*c*5), 77 medium instances (23 *j*10*c*5 instances, 24 *j*15*c*5 instances, 18 *j*10*c*10 instances, 12 *j*15*c*10 instances) and 40 large instances (10 *j*30*c*5*e* instances, 10 *j*40*c*5*e* instances, 10 *j*50*c*5*e* instances, 10 *j*60*c*5*e* instances) are generated for the EHFSP-V1 by adding energy-related parameters to the HFSP benchmarks.

Due to the complex nature of the EHFSP-V2, only 12 small instances with 5 jobs & 5 stages are used, which are created by truncating the instances with 10 jobs & 5 stages. Actually, there are 23 instances with 10 jobs & 5 stages. However, the Pareto-optimal solutions can be obtained for the EHFSP-V2 only for the truncated versions of the first 12 of these instances in reasonable computational time. Hence, only the truncated versions of the first 12 of these 23 instances are employed for the EHFSP-V2. Consequently, 12 small instances (*j*5*c*5), 77 medium instances (23 $j10c5$ instances, 24 $j15c5$ instances, 18 $j10c10$ instances, 12 $j15c10$ instances) and 40 large instances (10 $j30c5e$ instances, 10 $j40c5e$ instances, 10 $j50c5e$ instances, 10 $j60c5e$ instances) are generated for the EHFSP-V2 by adding energy-related parameters to the HFSP benchmarks.

In the calculation of TEC, the speed and energy parameters proposed by Mansouri et al. (2016) are used. As mentioned before, there are three processing speed levels for the machines: fast, normal and slow, and the corresponding processing speed factors are $v_l = \{1.2, 1.0, 0.8\}$. The conversion factors, which are used to estimate the energy consumption during processing are λ_l = {1.5, 1.0, 0.6} for fast, normal and slow processing speeds, respectively. It is assumed that the machines have the same power ($\beta_{ki} = 60\,kW \;\forall\, i \in I_k, k \in M$) with the same conversion factor for idle times ($\alpha_{ki} = 0.05 \;\forall\, i \in I_k, k \in M$).

As mentioned in Section 5.4, the augmented ε-constraint method is used to solve the proposed bi-objective MILP and CP models. All instances are solved through the augmented ε-constraint method using IBM ILOG CPLEX 12.8 on a Core i5, 2.80 GHz, 8 GB RAM computer. Note that, CP Optimizer suite of IBM ILOG CPLEX 12.8 is used to solve the CP model. Minimizing C_{max} is considered as the objective and TEC as a constraint. The lexicographic optimization is used for each objective function in order to obtain the payoff table with only Pareto-optimal solutions. Starting with an upper bound on TEC, which is found from the payoff table, the single-objective model is iteratively solved optimally by decreasing the constraint on TEC with a predetermined ε level, until the minimum value of TEC is reached.

In general, it is impossible to find all solutions on the continuous Pareto-optimal frontiers. Therefore, the Pareto-optimal frontiers are approximated for the small-sized instances using a very small ε level (10^{-3}). These finite number of Pareto-optimal solutions are referred to Pareto-optimal solution set (P). In order to avoid redundant iterations, the information about the objective space is used as soon as it is available. At any iteration, when the slack variable is larger than the ε level, it indicates that the same solution will be found in the next iteration, with the only difference being the slack variable. Therefore, a jump strategy is employed at each iteration by decreasing the constraint on TEC to the TEC value of the optimal solution found in the previous iteration. The Pareto-optimal solution sets are obtained for every small-sized instance using the augmented ε-constraint method with this acceleration strategy.

For medium instances, non-dominated solution sets are found by dividing the range of TEC objective function to 20 equal intervals and using this value as ε level. Due to the exponentially increasing solution times of the single-objective model, a 9-minute time limit is set for each iteration (3 hours time limit in total). Since the optimality of the solution is not guaranteed in each iteration, the aforementioned acceleration strategy is not used during this search process.

The proposed algorithms are coded in C++ programming language on Microsoft Visual Studio 2012 and all instances are solved on a Core i5, 2.80 GHz, 8 GB RAM computer. For the EHFSP-V1, thirty replications are carried out for each instance. In each replication, the algorithm is run for $25nm$ milliseconds for small instances, $50nm$ milliseconds for medium instances and $100nm$ milliseconds for large instances, where n denotes the number of jobs and m represents the number of stages. For the EHFSP-V2, twenty replications are carried out for each instance, where the algorithm is run for $100nm$ milliseconds in each replication. The population size of PS=100 is employed in all algorithms for the EHFSP-V1 (E_IG, E_IG$_{\mathrm{ALL}}$, E_VBIH, E_EM, E_EM$_{\mathrm{HFR}}$, E_EM$_{\mathrm{HFN}}$, E_EM$_{\mathrm{HFRN}}$). The population size of PS=30 is employed in all algorithms for the EHFSP-V2 (E_IG2, E_IG2$_{\mathrm{ALL}}$, E_VBIH2, E_EM2).

According to Okabe et al. (2003), there are three main aspects to assess the quality of a non-dominated solution set: the cardinality (i.e., the number of solutions), the convergence (i.e., the closeness to the Pareto-optimal frontier) and the diversity (i.e., the distribution and spread of the solutions). The performances of the proposed algorithms are evaluated with respect to these three main aspects in the following subsections.

Since very close approximations to Pareto-optimal frontiers (P) are obtained for small instances, the below performance metrics are used to assess the quality of the non-dominated solution set (S) found by a metaheuristic algorithm, together with the cardinality metric (number of non-dominated solutions found).

- Ratio of the Pareto-optimal solutions found (C_p) $= |S \cap P| / |P|$
- Inverted Generational Distance (IGD) = $\frac{\sum_{v \in P} d(v,S)}{|P|}$,

where $d(v,S)$ denotes the minimum Euclidean distance between v and the points in S. Lower IGD value is required to ensure that the set S is very close to the set P (Coello et al., 2007).

- Distribution Spacing (DS) (Tan et al., 2006)

DS(S) = $\frac{\left[\frac{1}{|S|}\sum_{i \in S}(d_i - \bar{d})^2\right]^{1/2}}{\bar{d}}$, where $\bar{d} = \frac{\sum_{i \in S} d_i}{|S|}$

and d_i denotes the minimum Euclidean distance between solution i and its nearest neighbor in S. The smaller value of spacing indicates that the solutions in S are more evenly distributed.

For medium and large instances, the non-dominated solution sets of time-limited MILP, time-limited CP and metaheuristic algorithms are compared with each other in terms of the aforementioned cardinality, C_p, IGD and DS metrics. As the Pareto-optimal solution sets (P) are not known for these instances, the reference sets (R) are used in C_p and IGD metrics. Note that the reference set includes only the high-quality non-dominated solutions, which are obtained by selecting all the non-dominated solutions found by the metaheuristic algorithms, time-limited MILP and CP approaches.

7.1 Parameter Calibration of the Algorithms

7.1.1 Parameter Calibration of the IG, IG$_{\text{ALL}}$ and VBIH Procedures

In order to calibrate the parameters of the algorithms, a design of experiment (DOE) (Montgomery, 2008) is carried out for the IG, IG$_{\text{ALL}}$ and VBIH algorithms with the makespan criterion. For this purpose, random instances with 5 stages are generated using the same methodology proposed in (Liao et al., 2012; Öztop et al., 2019), where the number of machines in each stage is randomly generated between 3 and 5 and the processing times of the jobs are uniform in the range [1,100]. Consequently, instances with 30, 40, 50 and 60 jobs are generated, each size having 5 instances, summing up to 20 instances. Both algorithms are coded in C++ programming language on Microsoft Visual Studio 2012, and a full factorial design of experiments is carried out for each algorithm on a Core i7, 2.60 GHz, 8 GB RAM computer.

The IG algorithm has two parameters; the destruction size (κ) and the temperature adjustment parameter (τP). After pilot experiments, we set κ to three levels as $\kappa \in (2,3,4)$; and τP to three levels as $\tau P \in (0.1, 0.3, 0.5)$ resulting in 9 treatments. There are 20 instances, and each instance is run for 9 treatments with a maximum CPU time $T_{max} = 20 \times n \times m$ milliseconds. The relative percentage deviation (RPD) is computed as a response variable of the experiment. Namely, we calculate the RPD for each instance-treatment pair as follows:

$$RPD(C_{max}^{i}) = \left(\frac{C_{max}^{i} - C_{max}^{min}}{C_{max}^{min}}\right) * 100 \qquad (7\text{-}1)$$

where C_{max}^{i} is the C_{max} value generated by a given heuristic in treatment *i*, and C_{max}^{min} is the minimum C_{max} found among 9 treatments for that instance.

An ANOVA procedure is performed after determining the RPD values for each instance, and the results are given in Figure 7.1. As shown in Figure 7.1, different levels for the κ and τP parameters do not result in statistically significant differences in the RPD values, as the *p* -values of these parameters are greater than the significance level, α= 0.05. This indicates that the IG performs rather robustly with respect to various levels of these parameters. Furthermore, no statistically significant interaction effect exists between parameters as the *p*-value of the parameter interaction effect is higher than the significance level.

$Source$	DF	$Seq\ SS$	$Adj\ SS$	$Adj\ MS$	$F-Ratio$	$p-value$
κ	2	0.08558	0.08558	0.04279	0.53	0.588
tP	2	0.26988	0.26988	0.13494	1.68	0.189
$\kappa * tP$	4	0.26735	0.26735	0.06684	0.83	0.506
$Error$	171	13.72465	13.72465	0.08026		
$Total$	179	14.34746				

Figure 7. 1. ANOVA Results for Parameters of the IG

As there is no significant interaction effect, the main effects plots of the parameters are also provided in Figure 7.2. Even though there is no statistically significant difference between various levels of κ and τP parameters, the plots demonstrate that settings of κ=4 and τP = 0.5 provide better RPD value than others. As shown in the figure, small τP levels result in worse RPD values and decreased algorithm performance, i.e., a higher setting of τP value provides better results. Note that, according to the detailed design of the experiments of Ruiz and Stützle (2007) for the original IG algorithm, κ=4 setting was also provided better performance than the other κ values between 2 and 8. Consequently, we set κ=4 for the IG, E_IG, and E_IG2 algorithms; and τP=0.5 in IG.

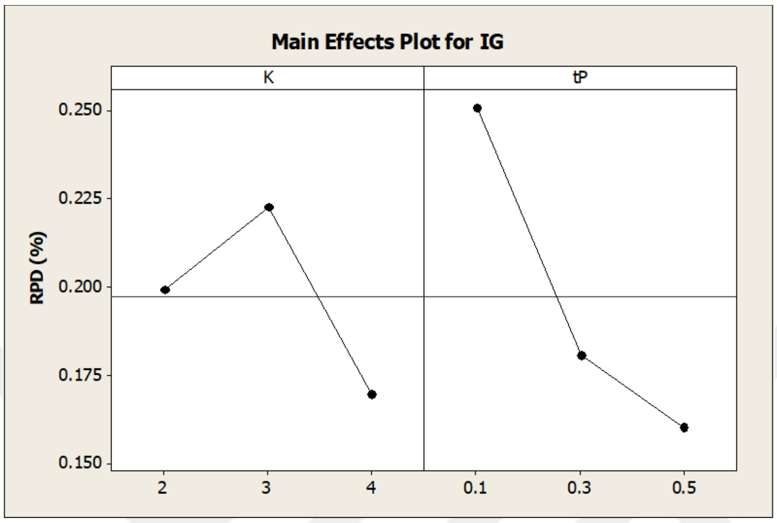


Figure 7. 2. Main Effects Plot for Parameters of the IG

Similar to the IG, a full factorial design is conducted for the parameter settings of IG$_{ALL}$. The IG$_{ALL}$ algorithm has two parameters; the destruction size (κ) and the temperature adjustment parameter (τP). We set κ to three levels as $\kappa \in (2,3,4)$; and τP to three levels as $\tau P \in (0.1, 0.3, 0.5)$ resulting in 9 treatments, similar to the IG. The RPD values are determined by employing the same method mentioned above, and the ANOVA results are given in Figure 7.3.

$Source$	DF	$Seq\ SS$	$Adj\ SS$	$Adj\ MS$	$F-Ratio$	$p-value$
κ	2	0.05038	0.05038	0.02519	0.39	0.675
tP	2	0.02959	0.02959	0.0148	0.23	0.794
$\kappa * tP$	4	0.06477	0.06477	0.01619	0.25	0.907
$Error$	171	10.9311	10.9311	0.06392		
$Total$	179	11.07584				

Figure 7. 3. ANOVA Results for Parameters of the IG$_{ALL}$

As shown in Figure 7.3, different levels for the κ and τP parameters do not result in statistically significant differences in the RPD values for the IG$_{ALL}$, as the p -values of these parameters are greater than the significance level α= 0.05. This states that the IG$_{ALL}$ performs rather robustly with respect to various levels of these parameters.

Additionally, no statistically significant interaction effect exists between parameters as the *p*-value of the parameter interaction effect is higher than the significance level.

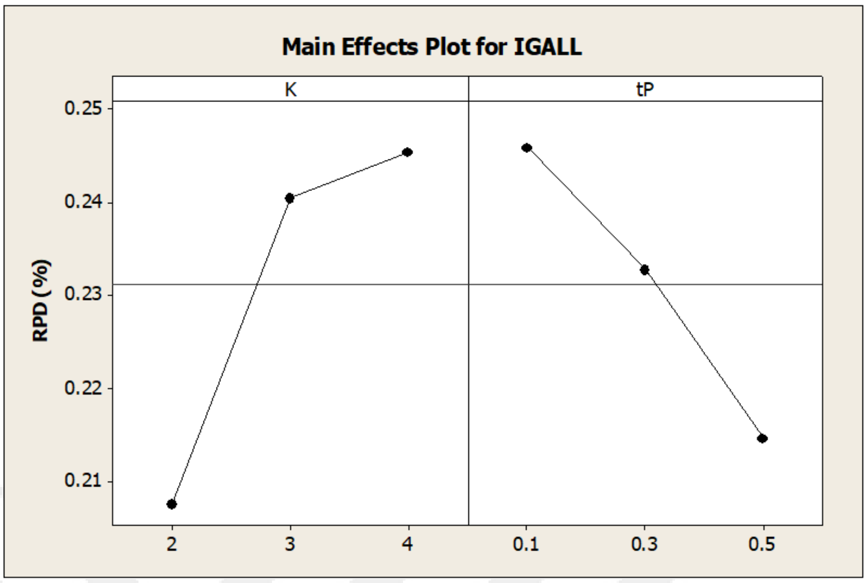


Figure 7. 4. Main Effects Plot for Parameters of the IG$_{\text{ALL}}$

Since there is no significant interaction effect, the main effects plots of the parameters for the IG$_{\text{ALL}}$ are also provided in Figure 7.4. Even though there is no statistically significant difference between various levels of κ and τP parameters, the plots demonstrate that settings of κ=2 and τP = 0.5 provide better RPD value than others. As shown in the figure, a higher setting of τP value provides better results, similar to the IG. Furthermore, as shown in the figure, high κ levels result in worse RPD values and decreased algorithm performance. Note that, according to the comprehensive experimental parameter tunings of Dubois-Lacoste et al. (2017) for the original IG$_{\text{ALL}}$ algorithm, κ=2 setting also provided better performance than the other κ values. Consequently, we set κ=2 for the IG$_{\text{ALL}}$, E_IG$_{\text{ALL}}$ and E_IG2$_{\text{ALL}}$ algorithms; and τP=0.5 in IG$_{\text{ALL}}$.

Similarly, a full factorial design is also conducted for the parameter settings of VBIH. The VBIH algorithm has two parameters; the maximum block size (bs_{max}) and the temperature adjustment parameter (τP). We set bs_{max} to three levels as $bs_{max} \in (2,3,4)$; and τP to three levels as $\tau P \in (0.1, 0.3, 0.5)$ resulting in 9 treatments. The RPD values are determined by employing the same method mentioned above, and the ANOVA results are given in Figure 7.5.

$Source$	DF	$Seq\ SS$	$Adj\ SS$	$Adj\ MS$	$F - Ratio$	$p - value$
bs_{max}	2	0.03153	0.03153	0.01576	0.16	0.849
tP	2	0.39535	0.39535	0.19767	2.05	0.131
$bs_{max} * tP$	4	0.09013	0.09013	0.02253	0.23	0.919
$Error$	171	16.46368	16.46368	0.09628		
$Total$	179	16.98068				

Figure 7. 5. ANOVA Results for Parameters of the VBIH

As shown in Figure 7.5, different levels for the bs_{max} and τP parameters do not result in statistically significant differences in the RPD values for the VBIH, as the p -values of these parameters are greater than the significance level, α= 0.05. This indicates that the VBIH performs rather robustly with respect to various levels of these parameters. Furthermore, no statistically significant interaction effect exists between parameters.

Since there is no significant interaction effect, the main effects plots of the parameters for the VBIH are also provided in Figure 7.6. Even though there is no statistically significant difference between various levels of bs_{max} and τP parameters, the plots demonstrate that settings of bs_{max}= 3 and τP = 0.5 provide better RPD value than others. As shown in the figure, a higher setting of τP value provides better results, similar to the IG and IG_{ALL} algorithms. Consequently, we set bs_{max}= 3 for the VBIH, E_VBIH and E_VBIH2 algorithms; and τP=0.5 in VBIH.

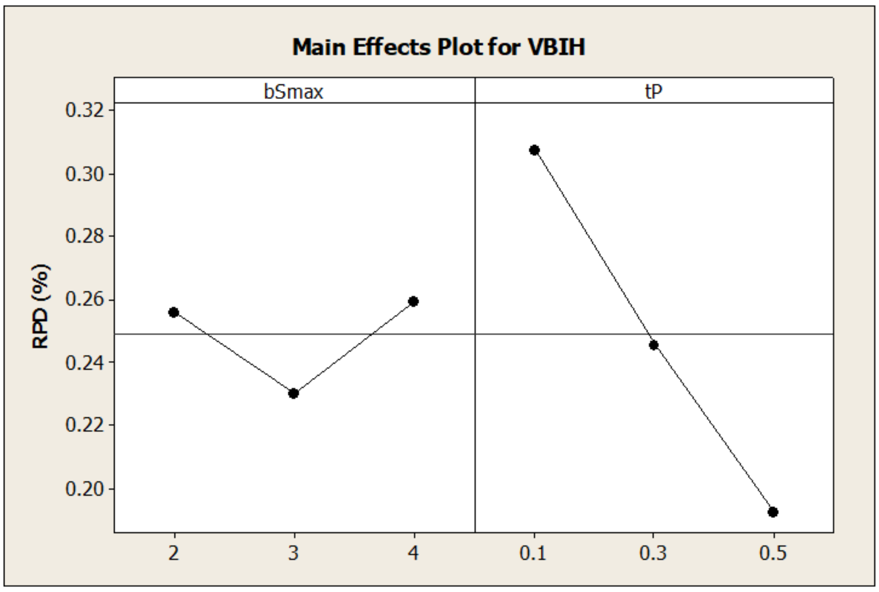


Figure 7. 6. Main Effects Plot for Parameters of the VBIH

7.1.2 Parameter Calibration of the HFN Approach

The HFN approach is employed in only two extensions of the E_EM algorithm for the EHFSP-V1, i.e., E_EM_{HFN} and E_EM_{HFRN} algorithms. Therefore, according to the preliminary experiments, we set the following parameters for the HFN approach of the $\mathrm{E\_EM_{HFN}}$ and $\mathrm{E\_EM_{HFRN}}$ algorithms: $sp = 0.60$ and $np = 0.60$ for small and medium instances, $sp = 0.40$ and $np = 0.60$ for large instances.

On the other hand, the HFN approach is employed in all algorithms for the EHFSP-V2 due to the more complex nature of the problem. Consequently, in order to calibrate the sp and np parameters of the algorithms for the EHFSP-V2, a design of experiment (DOE) is carried out for the E_IG2, $\mathrm{E\_IG2_{ALL}}$ and E_VBIH2 algorithms. For this purpose, random instances with 5 stages are generated using the same methodology proposed in (Liao et al., 2012; Öztop et al., 2019), where the number of machines in each stage is randomly generated between 3 and 5 and the processing times of the jobs are uniform in the range [1,100]. Consequently, instances with 30, 40, 50 and 60 jobs are generated, each size having 4 instances, summing up to 16 instances. Both algorithms are coded in C++ programming language on Microsoft Visual Studio 2012, and a full factorial design of experiments is carried out for each algorithm on a Core i7, 2.60 GHz, 8 GB RAM computer.

The HFN approach has two parameters; the swapping probability sp and the neighbor selection probability np. After pilot experiments, we set sp to three levels as $sp \in (0.4, 0.6, 08)$; and np to three levels as $np \in (0.4, 0.6, 08)$ resulting in 9 treatments. Each instance is run for 9 treatments with a maximum CPU time $T_{max} = 100 \times n \times m$ milliseconds. The IGD metric is computed as a response variable of the experiment. Namely, we calculate the IGD for each instance-treatment pair as follows:

$$\mathrm{IGD}\,(S_i) = \frac{\sum_{v \in P} d(v, S_i)}{|P|}, \tag{7-2}$$

where S_i is the non-dominated solution set generated by a given heuristic in treatment i, P is the reference set obtained from 9 treatments for that instance, and $d(v, S_i)$ denotes the minimum Euclidean distance between v and the points in S_i. Note that, the reference set includes only the high-quality non-dominated solutions, which are obtained by selecting all the non-dominated solutions found by the heuristic under all 9 treatments. A full factorial design of experiments is carried out for each algorithm using this methodology.

ANOVA results are given for the E_IG2 algorithm in Figure 7.7. As shown in Figure 7.7, different levels for the sp and np parameters do not result in statistically significant differences in the IGD values, as the p -values of these parameters are greater than the significance level, α= 0.05. It can be said that the E_IG2 performs rather robustly with respect to various levels of these parameters. Furthermore, no statistically significant interaction effect exists between parameters as the p-value of the parameter interaction effect is higher than the significance level.

$Source$	DF	$Seq\,SS$	$Adj\,SS$	$Adj\,MS$	$F-Ratio$	$p-value$
sp	2	92.3	92.3	46.2	0.37	0.691
np	2	444.3	444.3	222.1	1.78	0.172
$sp*np$	4	353.1	353.1	88.3	0.71	0.587
$Error$	135	16816.8	16816.8	124.6		
$Total$	143	17706.4				

Figure 7. 7. ANOVA Results for Parameters of the E_IG2

Since there is no significant interaction effect, the main effects plots of the parameters are also provided in Figure 7.8. Even though there is no statistically significant difference between various levels of sp and np parameters, the plots demonstrate that settings of sp=0.6 and np = 0.6 provide better IGD value than others. As shown in the figure, np=0.4 and np=0.8 settings have similar IGD values, but, the np=0.6 setting provides better IGD value than these settings. Consequently, we set sp=0.6 and np = 0.6 for the HFN procedure of the E_IG2 algorithm.

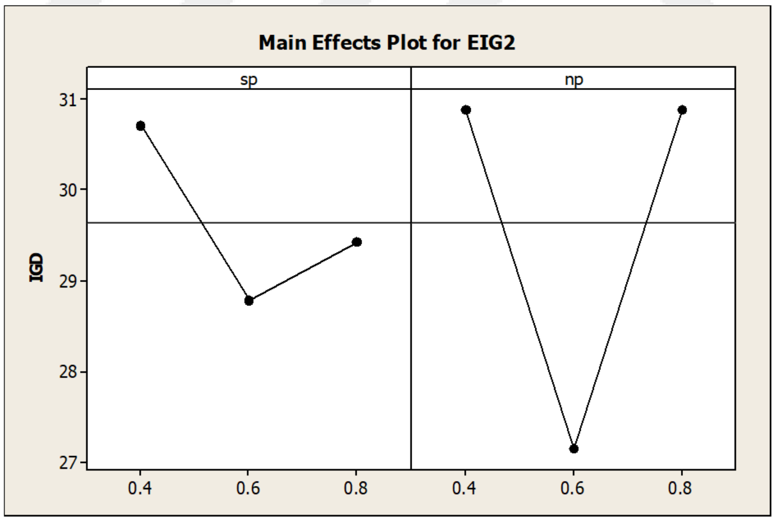


Figure 7. 8. Main Effects Plot for Parameters of the E_IG2

ANOVA results are given for the E_IG2$_{\text{ALL}}$ algorithm in Figure 7.9. The results show that the sp parameter is very significant at the significance level, α=0.05. However, different levels for the np parameter do not result in statistically significant differences in the IGD values, as the p-value of this parameter is greater than the significance level, α= 0.05. It can be said that the E_IG2$_{\text{ALL}}$ performs rather robustly with respect to various levels of the np parameter. Furthermore, no statistically significant interaction effect exists between parameters as the p-value of the parameter interaction effect is higher than the significance level.

$Source$	DF	$Seq\ SS$	$Adj\ SS$	$Adj\ MS$	$F-Ratio$	$p-value$
sp	2	1434	1434	717	5.15	0.007
np	2	401.3	401.3	200.6	1.44	0.240
$sp * np$	4	547.6	547.6	136.9	0.98	0.418
$Error$	135	18776.9	18776.9	139.1		
$Total$	143	21159.8				

Figure 7. 9. ANOVA Results for Parameters of the E_IG2_{ALL}

Since there is no significant interaction effect, the main effects plots of the parameters are provided in Figure 7.10. As shown in the figure, sp=0.6 setting has the minimum IGD value, where sp=0.8 setting has the worst IGD value. Even though there is no statistically significant difference between various levels of the np parameter, the plot demonstrates that settings of np = 0.4 and np = 0.6 provide better IGD value than the other one. Consequently, we set sp=0.6 and np = 0.6 for the HFN procedure of the $\text{E\_IG2}_{\text{ALL}}$ algorithm.

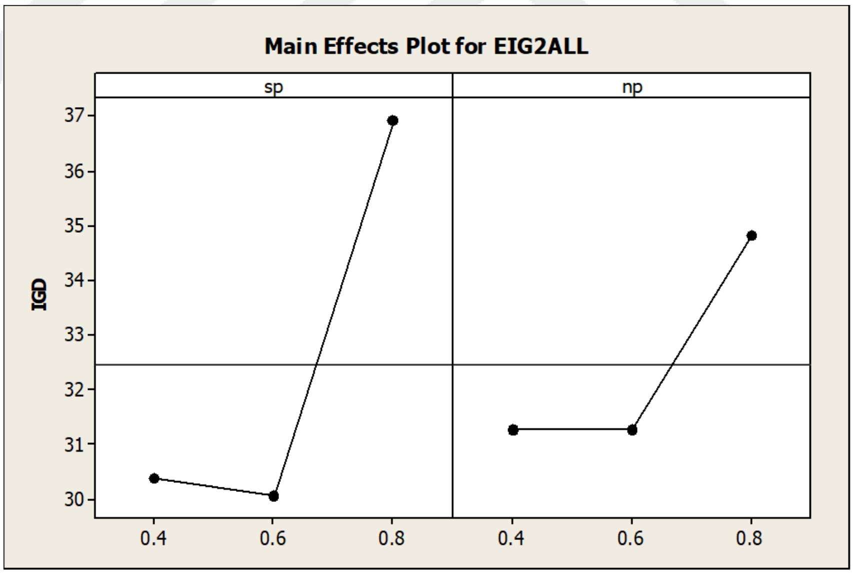


Figure 7. 10. Main Effects Plot for Parameters of the E_IG2_{ALL}

ANOVA results are given for the E_VBIH2 algorithm in Figure 7.11. As shown in Figure 7.11, different levels for the sp and np parameters do not result in statistically significant differences in the IGD values, as the p-values of these parameters are greater than the significance level, α = 0.05. It can be said that the E_VBIH2 performs rather robustly with respect to various levels of these parameters. Furthermore, no statistically significant interaction effect exists between parameters as the p-value of the parameter interaction effect is higher than the significance level.

$Source$	DF	$Seq\ SS$	$Adj\ SS$	$Adj\ MS$	$F-Ratio$	$p-value$
sp	2	298.4	298.4	149.2	0.57	0.569
np	2	519.4	519.4	259.7	0.99	0.376
$sp * np$	4	214.3	214.3	53.6	0.20	0.936
$Error$	135	35535.4	35535.4	263.2		
$Total$	143	36567.4				

Figure 7. 11. ANOVA Results for Parameters of the E_VBIH2

As there is no significant interaction effect, the main effects plots of the parameters are also provided in Figure 7.12. Even though there is no statistically significant difference between various levels of sp and np parameters, the plots demonstrate that settings of sp=0.6 and np = 0.6 provide better IGD value than others. Consequently, we set sp=0.6 and np = 0.6 for the HFN procedure of the E_VBIH2 algorithm. Since the E_EM2 algorithm is the combined version of E_IG2, E_IG2$_{\mathrm{ALL}}$ and E_VBIH2 algorithms, we also set sp=0.6 and np = 0.6 for the HFN procedure of the E_EM2 algorithm.

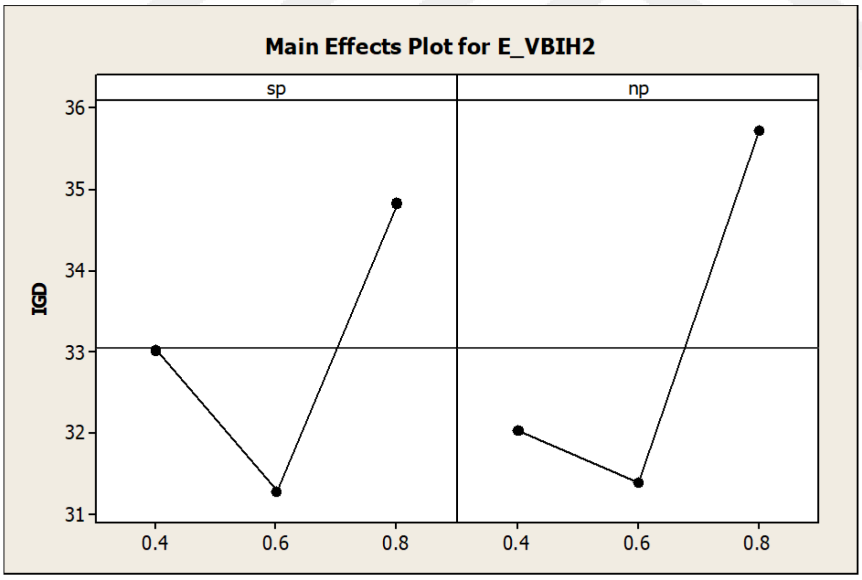


Figure 7. 12. Main Effects Plot for Parameters of the E_VBIH2

7.2 Comparison of Constructive Heuristics based on C_{max} Criterion

Before proceeding to the computational results for the bi-objective EHFSP, the impact of the proposed NEH_M(x) heuristic is initially analyzed on the solution quality. As mentioned before, all proposed algorithms in this thesis employ NEH_M(x) to generate the initial solution. In this section, the performance of NEH_M(x) is compared with the well-known NEH heuristic in order to demonstrate its efficiency. For evaluating the performances of these two constructive heuristics, the large instances proposed by (Liao et al., 2012; Öztop et al., 2019) with 30, 40, 50 and 60 jobs and 5 stages are used. All constructive heuristics are run for these instances on a Core i7, 2.60 GHz, 8 GB RAM computer. Note that, the comparisons are made considering only the makespan objective.

Table 7. 1. Comparison of Constructive Heuristics

Instance	NEH		NEH_M(x)		Instance	NEH		NEH_M(x)	
	RPD (%)	**CPU (ms)**	**RPD (%)**	**CPU (ms)**		**RPD (%)**	**CPU (ms)**	**RPD (%)**	**CPU (ms)**
j30c5e1	6.28	0	3.90	63	j50c5e1	2.37	0	1.34	234
j30c5e2	6.49	0	2.27	47	j50c5e2	2.41	15	0.40	219
j30c5e3	8.94	0	4.22	31	j50c5e3	6.03	0	1.94	234
j30c5e4	9.41	0	5.15	31	j50c5e4	2.20	16	0.77	250
j30c5e5	4.67	0	1.83	47	j50c5e5	3.80	0	2.24	235
j30c5e6	9.67	0	3.83	31	j50c5e6	8.81	0	2.38	250
j30c5e7	7.67	0	3.51	32	j50c5e7	4.74	15	0.75	235
j30c5e8	12.31	0	5.49	31	j50c5e8	2.80	0	1.17	250
j30c5e9	5.14	15	4.52	31	j50c5e9	8.73	16	4.44	250
j30c5e10	17.28	0	7.68	31	j50c5e10	6.19	0	2.48	266
j40c5e1	3.63	0	1.31	110	j60c5e1	6.16	16	1.23	500
j40c5e2	3.52	0	2.48	109	j60c5e2	5.65	0	3.26	531
j40c5e3	8.24	0	2.25	109	j60c5e3	4.92	15	1.99	516
j40c5e4	7.36	0	3.81	110	j60c5e4	5.58	0	3.30	562
j40c5e5	10.52	16	3.93	109	j60c5e5	3.48	16	2.45	516
j40c5e6	3.20	0	0.77	94	j60c5e6	5.92	16	1.91	547
j40c5e7	2.86	0	0.95	109	j60c5e7	0.90	0	0.57	484
j40c5e8	4.36	0	0.75	110	j60c5e8	1.26	15	0.00	500
j40c5e9	3.10	16	0.95	93	j60c5e9	0.61	0	0.17	485
j40c5e10	6.59	0	3.49	94	j60c5e10	3.91	16	0.87	515
					Average	**5.69**	**5.08**	**2.42**	**225.03**

Table 7.1 reports the results for each constructive heuristic. In the table, the relative percentage deviation (RPD) between the solution C_{max} and BC_{Max} is computed as follows:

$$\text{RPD} = \frac{C_{max} - BC_{Max}}{BC_{Max}} * 100, \tag{7-3}$$

where BC_{Max} is the best-known solution reported by Öztop et al. (2019). For each constructive heuristic, the total CPU time is also reported for each instance. Note that zero values in CPU times indicate negligible solution times.

As shown in Table 7.1, NEH is very fast, with a 5.08 ms average CPU time. However, its average RPD is 5.69%. Although NEH_M(x) takes 225.03 ms on average, its average RPD is 2.42% from the best-known solutions reported in Öztop et al. (2019). It is clear from Table 7.1 that NEH_M(x) significantly outperforms the NEH heuristic.

In order to further analyze the effect of the proposed NEH_M(x) constructive heuristic on the solution quality, we also run the single-objective algorithms (IG, IG_{ALL} and VBIH) for the two different versions depending on the initial solution generation. Namely, the initial solution is generated with the standard NEH in the first version of the algorithms, whereas the proposed NEH_M(x) is used for the initial solution generation in the second version of the algorithms. For evaluating the performances of these two versions of the single-objective algorithms, the aforementioned 40 large instances are used. All single-objective algorithms are run for these instances on a Core i7, 2.60 GHz, 8 GB RAM computer considering only the makespan criterion. For each algorithm, 15 independent replications are carried out for each instance. All algorithms are run for 20nm milliseconds in each replication, where n denotes the number of jobs and m represents the number of stages.

Table 7.2 reports the RPD results for each version of the algorithms, where RPD values are computed for each instance with respect to the best-known solutions as in Eq. (7-3). The average RPD values over 15 replications are reported for each algorithm, as well as the maximum and the minimum values.

As shown in Table 7.2, the second version of each algorithm (with NEH_M(x)) outperforms the first version of that algorithm (with NEH) in terms of both the average, minimum and maximum RPDs. Particularly, in terms of average RPDs, using the NEH_M(x) heuristic to generate the initial solution instead of NEH, improves the performance of the IG, IG_{ALL} and VBIH algorithms by 0.07%, 0.09% and 0.09% respectively, on the overall average.

In terms of minimum RPDs, IG_NEH_M(x) (0.15%), IG_{ALL}_NEH_M(x) (0.19%) and VBIH_ NEH_M(x) (0.17%) perform much better than IG_NEH (0.28%), IG_{ALL}_NEH (0.29%) and VBIH_NEH (0.27%). Similarly, in terms of maximum RPDs, IG_NEH_M(x) (0.40%), IG_{ALL}_NEH_M(x) (0.43%) and VBIH_ NEH_M(x) (0.43%) perform much better than IG_NEH (0.54%), IG_{ALL}_NEH (0.62%) and VBIH_NEH (0.67%). Consequently, it can be said that employing NEH_M(x) as a constructive heuristic significantly improves the performance of the algorithms.

Table 7. 2. Comparison of Single-Objective Algorithms

Instance	IG_NEH			IG_NEH_M(x)			IG$_{ALL}$_NEH			IG$_{ALL}$_NEH_M(x)			VBIH_NEH			VBIH_NEH_M(x)		
	RPD (%)			RPD (%)			RPD (%)			RPD (%)			RPD (%)			RPD (%)		
	Avg	Min	Max	Avg	Min	Max	Avg	Min	Max	Avg	Min	Max	Avg	Min	Max	Avg	Min	Max
j30c5e1	1.46	1.30	1.73	1.10	0.87	1.30	1.24	1.08	1.52	1.20	0.87	1.30	1.34	1.08	1.73	1.00	0.65	1.30
j30c5e2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
j30c5e3	0.93	0.84	1.01	0.87	0.51	1.01	0.93	0.84	1.01	0.89	0.67	1.01	0.97	0.84	1.18	0.83	0.51	1.01
j30c5e4	1.18	0.89	1.42	1.05	0.71	1.24	1.29	1.07	1.60	1.09	0.89	1.24	1.09	0.89	1.42	0.94	0.53	1.24
j30c5e5	0.62	0.50	0.83	0.56	0.33	0.67	0.67	0.50	0.83	0.59	0.50	0.83	0.74	0.50	1.00	0.64	0.50	0.83
j30c5e6	0.70	0.50	1.00	0.52	0.17	0.83	0.79	0.50	1.17	0.51	0.17	0.83	0.83	0.50	1.17	0.56	0.17	0.83
j30c5e7	0.09	0.00	0.16	0.04	0.00	0.16	0.11	0.00	0.32	0.05	0.00	0.16	0.11	0.00	0.32	0.03	0.00	0.16
j30c5e8	0.36	0.15	0.74	0.22	0.00	0.30	0.40	0.30	0.59	0.28	0.00	0.45	0.40	0.15	0.59	0.17	0.00	0.45
j30c5e9	0.54	0.47	0.78	0.57	0.16	0.78	0.62	0.47	0.78	0.52	0.16	0.78	0.63	0.47	0.93	0.58	0.47	0.78
j30c5e10	1.70	1.57	1.92	1.40	0.52	1.75	1.68	1.57	2.09	1.55	1.22	1.92	1.59	1.22	2.09	1.37	1.05	1.57
j40c5e1	0.15	0.15	0.15	0.14	0.00	0.15	0.15	0.15	0.15	0.14	0.00	0.15	0.15	0.15	0.15	0.15	0.15	0.15
j40c5e2	0.19	0.13	0.39	0.10	0.00	0.13	0.19	0.13	0.52	0.10	0.00	0.26	0.16	0.13	0.26	0.10	0.00	0.26
j40c5e3	0.05	0.00	0.25	0.00	0.00	0.00	0.07	0.00	0.25	0.00	0.00	0.00	0.05	0.00	0.25	0.03	0.00	0.25
j40c5e4	0.45	0.27	0.68	0.37	0.14	0.68	0.48	0.27	0.68	0.37	0.14	0.54	0.39	0.14	0.68	0.27	0.00	0.54
j40c5e5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
j40c5e6	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
j40c5e7	0.14	0.14	0.14	0.14	0.14	0.14	0.14	0.14	0.14	0.14	0.14	0.14	0.15	0.14	0.41	0.14	0.14	0.14
j40c5e8	0.19	0.12	0.25	0.16	0.00	0.25	0.21	0.12	0.25	0.22	0.12	0.25	0.22	0.12	0.25	0.20	0.12	0.25
j40c5e9	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.00	0.12	0.00	0.00	0.00	0.02	0.00	0.12	0.00	0.00	0.00
j40c5e10	0.48	0.39	0.65	0.40	0.26	0.52	0.49	0.39	0.65	0.44	0.13	0.65	0.54	0.39	0.78	0.53	0.26	0.65
j50c5e1	0.24	0.10	0.31	0.21	0.00	0.31	0.26	0.21	0.31	0.20	0.10	0.31	0.32	0.21	0.62	0.19	0.00	0.31
j50c5e2	0.07	0.00	0.20	0.02	0.00	0.10	0.16	0.00	0.40	0.05	0.00	0.20	0.14	0.00	0.20	0.11	0.00	0.20
j50c5e3	0.43	0.29	0.68	0.37	0.19	0.49	0.43	0.29	0.87	0.41	0.29	0.58	0.52	0.29	1.07	0.36	0.19	0.49
j50c5e4	0.01	0.00	0.11	0.00	0.00	0.00	0.04	0.00	0.22	0.00	0.00	0.00	0.01	0.00	0.22	0.01	0.00	0.11
j50c5e5	0.40	0.29	0.58	0.18	0.10	0.29	0.48	0.19	0.88	0.25	0.10	0.39	0.45	0.19	0.78	0.25	0.00	0.39
j50c5e6	0.36	0.36	0.36	0.36	0.36	0.36	0.39	0.36	0.60	0.37	0.36	0.48	0.49	0.36	0.71	0.43	0.36	0.60
j50c5e7	0.11	0.11	0.11	0.10	0.00	0.11	0.14	0.11	0.43	0.09	0.00	0.11	0.14	0.11	0.54	0.10	0.00	0.11
j50c5e8	0.03	0.00	0.12	0.00	0.00	0.00	0.02	0.00	0.12	0.03	0.00	0.12	0.02	0.00	0.12	0.02	0.00	0.12
j50c5e9	0.86	0.61	1.23	0.71	0.31	1.07	0.98	0.77	1.23	0.64	0.31	0.92	1.05	0.77	1.68	0.66	0.31	0.92
j50c5e10	0.21	0.00	0.41	0.10	0.00	0.31	0.37	0.10	1.14	0.15	0.00	0.31	0.33	0.10	1.34	0.09	0.00	0.21
j60c5e1	0.11	0.09	0.18	0.09	0.09	0.09	0.12	0.09	0.26	0.09	0.09	0.09	0.13	0.09	0.18	0.07	0.00	0.09
j60c5e2	1.01	0.76	1.30	0.81	0.43	0.98	0.96	0.76	1.20	0.77	0.54	0.98	0.83	0.65	1.09	0.75	0.43	0.98
j60c5e3	0.00	0.00	0.00	0.00	0.00	0.00	0.15	0.00	0.69	0.06	0.00	0.35	0.34	0.00	1.47	0.14	0.00	0.35
j60c5e4	0.85	0.51	2.03	0.60	0.38	0.89	0.90	0.63	1.27	0.58	0.38	0.76	0.84	0.63	1.40	0.62	0.38	0.76
j60c5e5	0.30	0.19	0.47	0.12	0.00	0.28	0.35	0.09	0.56	0.19	0.00	0.38	0.31	0.19	0.47	0.21	0.00	0.47
j60c5e6	0.41	0.29	0.67	0.38	0.19	0.57	0.45	0.29	0.95	0.40	0.29	0.48	0.43	0.29	0.76	0.45	0.19	0.57
j60c5e7	0.00	0.00	0.00	0.00	0.00	0.00	0.03	0.00	0.24	0.00	0.00	0.00	0.03	0.00	0.24	0.00	0.00	0.00
j60c5e8	0.02	0.00	0.10	0.00	0.00	0.00	0.05	0.00	0.10	0.00	0.00	0.00	0.05	0.00	0.10	0.00	0.00	0.00
j60c5e9	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
j60c5e10	0.30	0.22	0.65	0.22	0.22	0.22	0.25	0.22	0.65	0.22	0.22	0.22	0.28	0.22	0.65	0.22	0.22	0.22
Average	**0.37**	**0.28**	**0.54**	**0.30**	**0.15**	**0.40**	**0.40**	**0.29**	**0.62**	**0.31**	**0.19**	**0.43**	**0.40**	**0.27**	**0.67**	**0.31**	**0.17**	**0.43**

7.3 Small Instances for the EHFSP-V1

In the following tables, E_{HFRN}, E_{HFR}, E_{HFN}, E, IG, IG_{ALL} and $VBIH$ represent $\mathrm{E\_EM_{HFRN}}$, $\mathrm{E\_EM_{HFR}}$, $\mathrm{E\_EM_{HFN}}$, E_EM, E_IG, $\mathrm{E\_IG_{ALL}}$ and E_VBIH algorithms, respectively. Table 7.3 reports the results of *Cp*, IGD and DS performance metrics for each metaheuristic algorithm on the first set of small instances, which are obtained by truncating the instances with 10 jobs and 5 stages. $\mathrm{E\_EM_{HFRN}}$ finds 85%; $\mathrm{E\_EM_{HFR}}$ finds 83%; $\mathrm{E\_EM_{HFN}}$ finds 80%; E_EM, E_IG, $\mathrm{E\_IG_{ALL}}$ and E_VBIH find 64% of the Pareto-optimal solutions on the overall average. Note that, $\mathrm{E\_EM_{HFRN}}$ and $\mathrm{E\_EM_{HFN}}$ find all Pareto-optimal solutions for 10 out of 23 instances, where $\mathrm{E\_EM_{HFR}}$ finds all Pareto-optimal solutions for 9 instances; E_EM, E_IG, $\mathrm{E\_IG_{ALL}}$ and E_VBIH find all Pareto-optimal solutions for 5 instances. In terms of convergence, $\mathrm{E\_EM_{HFRN}}$ is the best performer with 0.23 IGD value in overall average whereas $\mathrm{E\_EM_{HFR}}$ also has a very small (0.27) IGD value. However, it can be said that all algorithms provide very close approximations to the Pareto-optimal solution set *P*, as the maximum of their average IGD values is 0.85. In terms of distribution spacing, solutions obtained by the metaheuristic algorithms are evenly distributed due to their low DS values.

Table 7.4 reports the results of *Cp*, IGD and DS performance metrics for each metaheuristic algorithm on the second set of small instances, which are obtained by truncating the instances with 15 jobs and 5 stages. As shown in the table, $\mathrm{E\_EM_{HFRN}}$ finds 88%; $\mathrm{E\_EM_{HFR}}$ finds 87%; $\mathrm{E\_EM_{HFN}}$ finds 82%; E_EM, E_IG, $\mathrm{E\_IG_{ALL}}$ and E_VBIH find 70% of the Pareto-optimal solutions in the overall average. $\mathrm{E\_EM_{HFRN}}$ finds all Pareto-optimal solutions for 14 out of 24 instances, where $\mathrm{E\_EM_{HFR}}$ finds all Pareto-optimal solutions for 13 instances; $\mathrm{E\_EM_{HFN}}$ finds all Pareto-optimal solutions for 8 instances; E_EM, E_IG, $\mathrm{E\_IG_{ALL}}$ and E_VBIH find all Pareto-optimal solutions for 7 instances. In terms of IGD values, $\mathrm{E\_EM_{HFRN}}$ (0.21) and $\mathrm{E\_EM_{HFR}}$ (0.23) are the best performer ones on the overall average, whereas $\mathrm{E\_EM_{HFN}}$ also has a very small (0.33) IGD value. All algorithms provide very close approximations to the Pareto-optimal solution set *P*, as the maximum of their average IGD values is 0.58. In terms of the spread of the solutions, solutions obtained by the metaheuristic algorithms are evenly distributed, as they have very low DS values.

Finally, as the ensembles of metaheuristic algorithms with HFR/HFN approaches have higher *Cp* and lower IGD values, it can be said that HFR and HFN approaches substantially improve the solution quality for these small instances. When the ensembles of metaheuristic algorithms are compared with each other, it can be said that E_EM$_{HFRN}$ and E_EM$_{HFR}$ perform slightly better.

In order to visualize the performance of the algorithms, the Pareto frontiers obtained by the algorithms are provided for an instance with five jobs and five stages in Figure 7.13. As shown in Figure 7.13, the ensembles of metaheuristic algorithms with HFR/HFN approaches (E_EM$_{HFRN}$, E_EM$_{HFR}$, E_EM$_{HFN}$) outperform the other metaheuristics (E_EM, E_IG, E_IG$_{ALL}$ and E_VBIH), as they provide better approximations to the Pareto-optimal frontier. Note that, E_EM$_{HFRN}$ and E_EM$_{HFR}$ perform slightly better than the E_EM$_{HFN}$ algorithm, as seen in Figure 7.13.

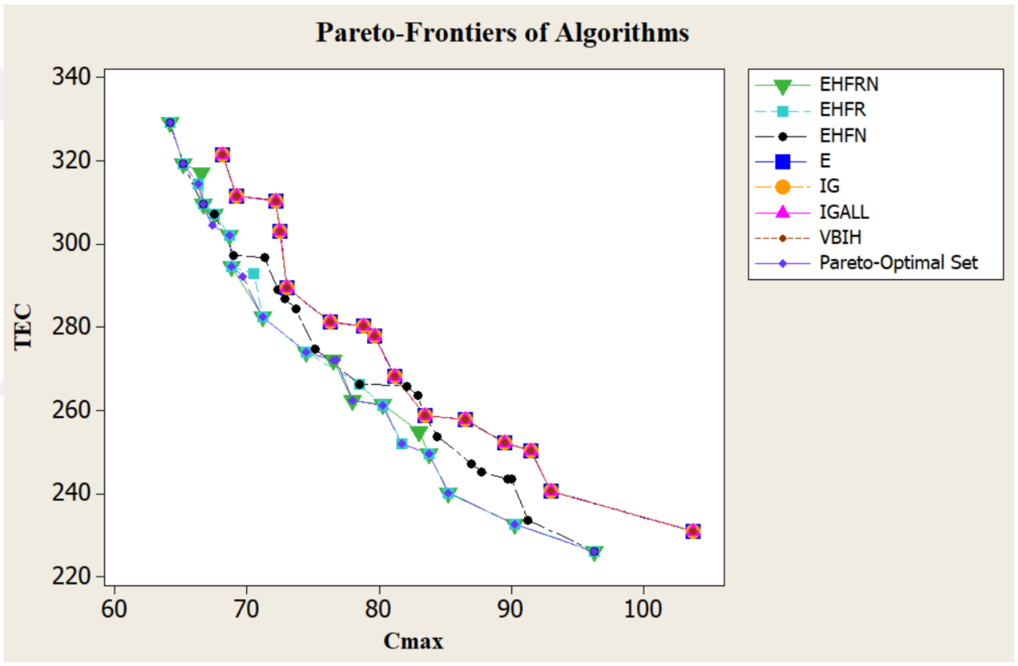


Figure 7. 13. Comparison of Algorithms for an Instance with 5 Jobs

Table 7. 3. Performance Comparison of Algorithms on Small Instances (Set 1) for the EHFSP-V1

	Cardinality							**Ratio of Pareto-optimal Solutions Found (C_p)**						
Instance	**EHFRN**	**EHFR**	**EHFN**	**E**	**IG**	**IGALL**	**VBIH**	**EHFRN**	**EHFR**	**EHFN**	**E**	**IG**	**IGALL**	**VBIH**
1_j5c5a2	24	24	25	24	24	24	24	0.55	0.59	0.50	0.23	0.23	0.23	0.23
1_j5c5a3	26	25	23	21	21	21	21	0.48	0.48	0.48	0.38	0.38	0.38	0.38
1_j5c5a4	18	19	14	18	18	18	18	0.47	0.53	0.35	0.35	0.35	0.35	0.35
1_j5c5a5	17	17	21	15	15	15	15	0.68	0.74	0.21	0.00	0.00	0.00	0.00
1_j5c5a6	25	25	25	24	24	24	24	0.96	0.96	1.00	0.88	0.88	0.88	0.88
1_j5c5b1	25	25	25	24	24	24	24	1.00	1.00	1.00	0.96	0.96	0.96	0.96
1_j5c5b2	18	18	18	17	17	17	17	1.00	1.00	1.00	0.94	0.94	0.94	0.94
1_j5c5b3	23	23	23	23	23	23	23	1.00	1.00	1.00	1.00	1.00	1.00	1.00
1_j5c5b4	20	20	20	22	22	22	22	1.00	0.95	1.00	0.90	0.90	0.90	0.90
1_j5c5b5	29	29	29	29	29	29	29	1.00	1.00	1.00	1.00	1.00	1.00	1.00
1_j5c5b6	14	14	14	14	14	14	14	1.00	1.00	1.00	1.00	1.00	1.00	1.00
1_j5c5c1	17	17	17	15	15	15	15	0.88	0.88	0.88	0.82	0.82	0.82	0.82
1_j5c5c2	22	21	20	22	22	22	22	0.67	0.63	0.63	0.21	0.21	0.21	0.21
1_j5c5c3	15	15	14	16	16	16	16	0.93	0.93	0.87	0.60	0.60	0.60	0.60
1_j5c5c4	13	13	13	13	13	13	13	1.00	1.00	1.00	1.00	1.00	1.00	1.00
1_j5c5c5	16	16	17	20	20	20	20	0.61	0.50	0.44	0.17	0.17	0.17	0.17
1_j5c5c6	13	14	14	16	16	16	16	0.83	0.58	0.75	0.42	0.42	0.42	0.42
1_j5c5d1	13	13	13	12	12	12	12	1.00	1.00	1.00	0.69	0.69	0.69	0.69
1_j5c5d2	18	17	18	15	15	15	15	0.95	0.79	0.84	0.53	0.53	0.53	0.53
1_j5c5d3	16	17	16	16	16	16	16	0.67	0.72	0.72	0.61	0.61	0.61	0.61
1_j5c5d4	21	20	24	21	21	21	21	0.91	0.87	0.78	0.65	0.65	0.65	0.65
1_j5c5d5	21	21	19	19	19	19	19	1.00	1.00	0.90	0.38	0.38	0.38	0.38
1_j5c5d6	13	13	13	13	13	13	13	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Average	**19.00**	**18.96**	**18.91**	**18.65**	**18.65**	**18.65**	**18.65**	**0.85**	**0.83**	**0.80**	**0.64**	**0.64**	**0.64**	**0.64**

Table 7. 3. (Cont'd) Performance Comparison of Algorithms on Small Instances (Set 1) for the EHFSP-V1

	IGD							**Distribution Spacing (DS)**						
Instance	**EHFRN**	**EHFR**	**EHFN**	**E**	**IG**	**IGALL**	**VBIH**	**EHFRN**	**EHFR**	**EHFN**	**E**	**IG**	**IGALL**	**VBIH**
1_j5c5a2	0.88	0.78	1.06	2.01	2.01	2.01	2.01	0.63	0.78	0.49	0.56	0.56	0.56	0.56
1_j5c5a3	0.69	0.69	0.67	0.92	0.92	0.92	0.92	1.27	1.35	1.18	1.15	1.15	1.15	1.15
1_j5c5a4	0.95	0.87	1.17	1.88	1.88	1.88	1.88	0.93	0.99	0.66	0.93	0.93	0.93	0.93
1_j5c5a5	0.60	0.62	2.69	5.93	5.93	5.93	5.93	0.55	0.48	0.76	0.66	0.66	0.66	0.66
1_j5c5a6	0.01	0.01	0.00	0.14	0.14	0.14	0.14	0.52	0.52	0.52	0.53	0.53	0.53	0.53
1_j5c5b1	0.00	0.00	0.00	0.01	0.01	0.01	0.01	1.07	1.07	1.07	0.97	0.97	0.97	0.97
1_j5c5b2	0.00	0.00	0.00	0.04	0.04	0.04	0.04	1.07	1.07	1.07	0.95	0.95	0.95	0.95
1_j5c5b3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.83	0.83	0.83	0.83	0.83	0.83	0.83
1_j5c5b4	0.00	0.07	0.00	0.14	0.14	0.14	0.14	0.80	0.79	0.80	1.02	1.02	1.02	1.02
1_j5c5b5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.88	0.88	0.88	0.88	0.88	0.88	0.88
1_j5c5b6	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.71	0.71	0.71	0.71	0.71	0.71	0.71
1_j5c5c1	0.09	0.09	0.09	0.15	0.15	0.15	0.15	1.17	1.17	1.17	0.95	0.95	0.95	0.95
1_j5c5c2	0.41	0.40	0.44	1.17	1.17	1.17	1.17	1.00	1.03	1.00	1.06	1.06	1.06	1.06
1_j5c5c3	0.02	0.02	0.19	0.37	0.37	0.37	0.37	0.59	0.59	0.57	0.64	0.64	0.64	0.64
1_j5c5c4	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.54	0.54	0.54	0.54	0.54	0.54	0.54
1_j5c5c5	0.30	0.49	0.60	1.89	1.89	1.89	1.89	0.86	0.97	1.01	1.15	1.15	1.15	1.15
1_j5c5c6	0.66	0.91	0.53	1.22	1.22	1.22	1.22	0.40	0.40	0.38	0.46	0.46	0.46	0.46
1_j5c5d1	0.00	0.00	0.00	0.19	0.19	0.19	0.19	0.85	0.85	0.85	0.76	0.76	0.76	0.76
1_j5c5d2	0.11	0.40	0.28	1.09	1.09	1.09	1.09	0.71	0.71	0.68	0.61	0.61	0.61	0.61
1_j5c5d3	0.58	0.38	0.40	0.61	0.61	0.61	0.61	0.94	0.92	0.83	0.93	0.93	0.93	0.93
1_j5c5d4	0.04	0.43	0.39	0.79	0.79	0.79	0.79	0.73	0.73	0.87	0.86	0.86	0.86	0.86
1_j5c5d5	0.00	0.00	0.22	0.96	0.96	0.96	0.96	0.97	0.97	0.91	0.90	0.90	0.90	0.90
1_j5c5d6	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.59	0.59	0.59	0.59	0.59	0.59	0.59
Average	**0.23**	**0.27**	**0.38**	**0.85**	**0.85**	**0.85**	**0.85**	**0.81**	**0.82**	**0.80**	**0.81**	**0.81**	**0.81**	**0.81**

Table 7. 4. Performance Comparison of Algorithms on Small Instances (Set 2) for the EHFSP-V1

	Cardinality							**Ratio of Pareto-optimal Solutions Found (C_p)**						
Instance	**EHFRN**	**EHFR**	**EHFN**	**E**	**IG**	**IGALL**	**VBIH**	**EHFRN**	**EHFR**	**EHFN**	**E**	**IG**	**IGALL**	**VBIH**
2_j5c5a1	14	14	13	15	15	15	15	0.87	0.87	0.87	0.73	0.73	0.73	0.73
2_j5c5a2	20	20	20	20	20	20	20	1.00	1.00	1.00	1.00	1.00	1.00	1.00
2_j5c5a3	20	20	23	19	19	19	19	0.40	0.40	0.27	0.00	0.00	0.00	0.00
2_j5c5a4	28	25	27	23	23	23	23	0.14	0.21	0.03	0.00	0.00	0.00	0.00
2_j5c5a5	37	38	37	35	35	35	35	0.83	0.67	0.64	0.28	0.28	0.28	0.28
2_j5c5a6	16	16	16	16	16	16	16	0.62	0.57	0.29	0.19	0.19	0.19	0.19
2_j5c5b1	30	30	31	29	29	29	29	1.00	1.00	0.97	0.77	0.77	0.77	0.77
2_j5c5b2	23	23	23	24	24	24	24	1.00	1.00	0.96	0.87	0.87	0.87	0.87
2_j5c5b3	21	21	21	21	21	21	21	1.00	1.00	1.00	1.00	1.00	1.00	1.00
2_j5c5b4	18	18	18	18	18	18	18	1.00	1.00	1.00	1.00	1.00	1.00	1.00
2_j5c5b5	47	47	47	47	47	47	47	1.00	1.00	1.00	1.00	1.00	1.00	1.00
2_j5c5b6	28	28	28	28	28	28	28	1.00	1.00	1.00	1.00	1.00	1.00	1.00
2_j5c5c1	18	18	17	17	17	17	17	1.00	1.00	0.94	0.78	0.78	0.78	0.78
2_j5c5c2	18	17	16	16	16	16	16	0.89	0.84	0.84	0.58	0.58	0.58	0.58
2_j5c5c3	20	20	20	21	21	21	21	0.90	0.90	0.90	0.86	0.86	0.86	0.86
2_j5c5c4	17	17	19	18	18	18	18	0.67	0.67	0.61	0.50	0.50	0.50	0.50
2_j5c5c5	12	12	12	11	11	11	11	0.92	0.92	0.92	0.77	0.77	0.77	0.77
2_j5c5c6	16	16	16	16	16	16	16	1.00	1.00	0.94	0.75	0.75	0.75	0.75
2_j5c5d1	19	19	19	19	19	19	19	1.00	1.00	1.00	1.00	1.00	1.00	1.00
2_j5c5d2	13	13	13	13	13	13	13	1.00	1.00	1.00	1.00	1.00	1.00	1.00
2_j5c5d3	15	15	14	15	15	15	15	0.88	0.81	0.63	0.31	0.31	0.31	0.31
2_j5c5d4	19	19	18	17	17	17	17	1.00	0.95	0.84	0.74	0.74	0.74	0.74
2_j5c5d5	15	15	15	16	16	16	16	1.00	1.00	0.93	0.73	0.73	0.73	0.73
2_j5c5d6	23	23	23	23	23	23	23	1.00	1.00	1.00	0.96	0.96	0.96	0.96
Average	**21.13**	**21.00**	**21.08**	**20.71**	**20.71**	**20.71**	**20.71**	**0.88**	**0.87**	**0.82**	**0.70**	**0.70**	**0.70**	**0.70**

Table 7. 4. (Cont'd) Performance Comparison of Algorithms on Small Instances (Set 2) for the EHFSP-V1

	IGD							Distribution Spacing (DS)						
Instance	**EHFRN**	**EHFR**	**EHFN**	**E**	**IG**	**IGALL**	**VBIH**	**EHFRN**	**EHFR**	**EHFN**	**E**	**IG**	**IGALL**	**VBIH**
2_j5c5a1	0.10	0.10	0.12	0.24	0.24	0.24	0.24	0.92	0.92	0.75	0.96	0.96	0.96	0.96
2_j5c5a2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.93	0.93	0.93	0.93	0.93	0.93	0.93
2_j5c5a3	1.15	1.15	1.38	3.12	3.12	3.12	3.12	0.71	0.72	0.73	0.69	0.69	0.69	0.69
2_j5c5a4	2.32	2.17	2.92	3.57	3.57	3.57	3.57	0.83	0.77	0.90	0.59	0.59	0.59	0.59
2_j5c5a5	0.22	0.47	0.44	1.35	1.35	1.35	1.35	0.89	0.95	0.92	1.13	1.13	1.13	1.13
2_j5c5a6	0.48	0.59	0.86	1.69	1.69	1.69	1.69	0.84	0.86	0.86	0.82	0.82	0.82	0.82
2_j5c5b1	0.00	0.00	0.02	0.23	0.23	0.23	0.23	1.38	1.38	1.44	1.32	1.32	1.32	1.32
2_j5c5b2	0.00	0.00	0.02	0.10	0.10	0.10	0.10	1.00	1.00	0.97	1.00	1.00	1.00	1.00
2_j5c5b3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.27	1.27	1.27	1.27	1.27	1.27	1.27
2_j5c5b4	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.08	1.08	1.08	1.08	1.08	1.08	1.08
2_j5c5b5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.17	1.17	1.17	1.17	1.17	1.17	1.17
2_j5c5b6	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.73	0.73	0.73	0.73	0.73	0.73	0.73
2_j5c5c1	0.00	0.00	0.01	0.16	0.16	0.16	0.16	0.96	0.96	0.86	0.87	0.87	0.87	0.87
2_j5c5c2	0.19	0.27	0.32	0.64	0.64	0.64	0.64	0.86	0.83	0.79	0.86	0.86	0.86	0.86
2_j5c5c3	0.03	0.03	0.03	0.06	0.06	0.06	0.06	0.81	0.81	0.81	0.83	0.83	0.83	0.83
2_j5c5c4	0.29	0.32	0.34	0.60	0.60	0.60	0.60	0.99	0.91	1.21	1.08	1.08	1.08	1.08
2_j5c5c5	0.08	0.08	0.08	0.17	0.17	0.17	0.17	0.41	0.41	0.41	0.05	0.05	0.05	0.05
2_j5c5c6	0.00	0.00	0.01	0.28	0.28	0.28	0.28	0.72	0.72	0.72	0.77	0.77	0.77	0.77
2_j5c5d1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.24	1.24	1.24	1.24	1.24	1.24	1.24
2_j5c5d2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.41	0.41	0.41	0.41	0.41	0.41	0.41
2_j5c5d3	0.27	0.28	1.18	1.17	1.17	1.17	1.17	0.60	0.68	0.53	0.56	0.56	0.56	0.56
2_j5c5d4	0.00	0.03	0.19	0.26	0.26	0.26	0.26	0.94	0.94	0.97	0.92	0.92	0.92	0.92
2_j5c5d5	0.00	0.00	0.01	0.17	0.17	0.17	0.17	0.74	0.74	0.75	0.80	0.80	0.80	0.80
2_j5c5d6	0.00	0.00	0.00	0.07	0.07	0.07	0.07	1.34	1.34	1.34	1.28	1.28	1.28	1.28
Average	**0.21**	**0.23**	**0.33**	**0.58**	**0.58**	**0.58**	**0.58**	**0.91**	**0.91**	**0.91**	**0.89**	**0.89**	**0.89**	**0.89**

7.4 Medium & Large Instances for the EHFSP-V1

As mentioned in the beginning of Section 7, for medium and large instances, the non-dominated solution sets of time-limited MILP, time-limited CP and metaheuristic algorithms are compared with each other in terms of the aforementioned cardinality, C_p, IGD and DS metrics. As the Pareto-optimal solution sets (P) are not known for these instances, the reference sets (R) are used in C_p and IGD metrics. Note that the reference set includes only the high-quality non-dominated solutions, which are obtained by selecting all the non-dominated solutions found by the seven metaheuristic algorithms, time-limited MILP and CP approaches.

In order to make the computational results statistically convincing, a series of Wilcoxon signed-rank tests is also conducted at the significance level of $\alpha = 0.05$. Note that, Wilcoxon signed-rank test is a non-parametric test to compare two related groups, which is based on the differences between paired observations. This test is employed to decide whether there is a statistically significant difference between the two solution approaches in terms of a certain performance metric. Let m_D denotes the median of the difference between two different algorithms for a certain metric, the null hypothesis is defined by H_0: $m_D = 0$ indicating that there is no difference between the two algorithms in terms of that metric, whereas the alternative hypothesis is defined by H_1: $m_D \neq 0$ indicating that there is a difference between the two algorithms. For each pair of the algorithms, the p-value results of the Wilcoxon signed-rank tests are reported for all performance metrics. Note that there is a statistically significant difference between the two algorithms in terms of a certain performance metric if the corresponding p-value is smaller than $\alpha = 0.05$.

Table 7.5 reports the performances of the time-limited MILP (MILP), time-limited CP (CP), E_EM$_{\text{HFRN}}$(E_{HFRN}), E_EM$_{\text{HFR}}$ (E_{HFR}), E_EM$_{\text{HFN}}$ (E_{HFN}), E_EM(E), E_IG (IG), E_IG$_{\text{ALL}}$ (IG_{ALL}) and E_VBIH ($VBIH$) algorithms on medium instances with 10 jobs. Furthermore, Table 7.6 reports the p-value results of the Wilcoxon signed-rank tests for these instances. As shown in Table 7.5, each metaheuristic algorithm finds approximately three times as many non-dominated solutions as the time-limited MILP and CP, in exceptionally fewer computation times. The statistical results reported in Table 7.6 also confirm that all metaheuristics perform significantly better than the MILP and CP in terms of the cardinality metric. Note that there is no statistically significant difference between the metaheuristics in terms of cardinality. As shown in Table 7.5, $\mathrm{E\_EM_{HFRN}}$ and $\mathrm{E\_EM_{HFN}}$ find 69%, $\mathrm{E\_EM_{HFR}}$ finds 67%; E_EM, $\mathrm{E\_IG_{ALL}}$ and E_VBIH find 62%; E_IG finds 61%; time-limited CP finds 27%, and time-limited MILP finds 14% of the reference solutions on the overall average. Note that, $\mathrm{E\_EM_{HFN}}$ finds all reference solutions for 12 out of 41 instances, where E_VBIH finds all reference solutions for 7 instances; $\mathrm{E\_EM_{HFRN}}$, $\mathrm{E\_EM_{HFR}}$, E_EM, $\mathrm{E\_IG_{ALL}}$ and E_IG find all reference solutions for 8 instances. According to the p-value results reported in Table 7.6, all metaheuristics perform significantly and statistically better than the MILP and CP in terms of C_p metric. Note that, $\mathrm{E\_EM_{HFRN}}$ and $\mathrm{E\_EM_{HFN}}$ algorithms are statistically equivalent in terms of C_p metric and they perform statistically better than the other metaheuristics. There is no statistically significant difference between E_EM, E_IG, $\mathrm{E\_IG_{ALL}}$ and E_VBIH algorithms in terms of C_p metric.

In terms of convergence, $\mathrm{E\_EM_{HFRN}}$ (0.48), $\mathrm{E\_EM_{HFR}}$ (0.50) and $\mathrm{E\_EM_{HFN}}$ (0.49) are the best performer ones on the overall average, whereas other metaheuristic algorithms also have small IGD values around 0.59. In terms of IGD metric, all metaheuristic algorithms outperform the time-limited MILP and CP, while ensembles of metaheuristic algorithms with HFR/HFN approaches slightly outperform the other metaheuristic algorithms. This statement is also consistent with the Wilcoxon signed-rank test results reported in Table 7.6. As shown in Table 7.6, $\mathrm{E\_EM_{HFRN}}$, $\mathrm{E\_EM_{HFR}}$ and $\mathrm{E\_EM_{HFN}}$ algorithms are statistically equivalent in terms of IGD metric, and they perform statistically better than the other solution approaches. Note that, there is no statistically significant difference between E_EM, E_IG, $\mathrm{E\_IG_{ALL}}$ and E_VBIH algorithms in terms of IGD metric, except the E_IG vs. E_VBIH pair.

For the comparison of distribution spacing metric, time-limited MILP and CP approaches have smaller DS values than the metaheuristic algorithms, which implies that the solutions generated by MILP and CP approaches are spread more uniformly in their own discovered frontiers. This statement is also consistent with the Wilcoxon signed-rank test results reported in Table 7.6. This result is expected, as a constant ε level is used through the augmented ε-constraint method in the time-limited MILP and CP approaches. Nevertheless, the metaheuristic algorithms also have low DS values, indicating even dispersions.

Table 7. 5. Performance Comparison of Algorithms on Medium Instances with 10 Jobs for the EHFSP-V1

	Cardinality									**Ratio of Reference Solutions Found (C_p)**								
Instance	**MILP**	**CP**	**EHFRN**	**EHFR**	**EHFN**	**E**	**IG**	**IGALL**	**VBIH**	**MILP**	**CP**	**EHFRN**	**EHFR**	**EHFN**	**E**	**IG**	**IGALL**	**VBIH**
j10c5a2	21	21	79	79	79	79	79	79	79	0.27	0.27	0.99	0.99	1.00	1.00	1.00	1.00	1.00
j10c5a3	21	21	88	89	89	89	89	89	87	0.24	0.24	0.99	1.00	1.00	1.00	1.00	1.00	0.98
j10c5a4	21	21	98	98	98	98	98	98	98	0.21	0.21	1.00	1.00	1.00	1.00	1.00	1.00	1.00
j10c5a5	21	21	118	118	118	118	118	118	118	0.18	0.18	1.00	1.00	1.00	1.00	1.00	1.00	1.00
j10c5a6	21	21	85	82	87	81	82	83	83	0.24	0.24	0.84	0.75	0.78	0.66	0.62	0.74	0.62
j10c5b1	21	21	77	77	77	77	77	77	77	0.27	0.27	1.00	1.00	1.00	1.00	1.00	1.00	1.00
j10c5b2	21	21	72	72	73	72	72	72	72	0.29	0.29	0.99	0.99	1.00	0.99	0.99	0.99	0.99
j10c5b3	21	21	91	91	91	93	93	93	93	0.21	0.23	0.99	0.99	0.99	0.93	0.93	0.93	0.93
j10c5b4	21	21	89	89	89	89	89	89	89	0.24	0.24	1.00	1.00	1.00	0.98	0.98	0.98	0.98
j10c5b5	21	21	96	96	96	96	96	96	96	0.22	0.22	1.00	1.00	1.00	1.00	1.00	1.00	1.00
j10c5b6	21	21	97	97	97	97	97	97	97	0.22	0.22	1.00	1.00	1.00	1.00	1.00	1.00	1.00
j10c5c1	20	21	68	71	71	68	76	62	71	0.10	0.36	0.51	0.36	0.49	0.41	0.24	0.32	0.36
j10c5c2	19	21	78	72	71	82	69	73	66	0.09	0.36	0.31	0.34	0.38	0.34	0.33	0.29	0.38
j10c5c3	19	21	75	75	81	71	73	76	75	0.10	0.51	0.34	0.32	0.39	0.27	0.22	0.24	0.27
j10c5c4	20	18	55	61	56	59	60	59	60	0.02	0.20	0.67	0.60	0.70	0.62	0.50	0.47	0.48
j10c5c5	19	20	71	75	67	75	71	78	70	0.07	0.33	0.46	0.41	0.48	0.30	0.28	0.36	0.34
j10c5c6	15	21	67	71	69	64	69	64	69	0.07	0.30	0.57	0.52	0.58	0.48	0.39	0.48	0.43
j10c5d1	21	21	54	55	57	60	54	59	67	0.09	0.30	0.48	0.45	0.41	0.45	0.35	0.42	0.33
j10c5d2	17	21	64	58	65	70	54	68	64	0.03	0.19	0.48	0.58	0.50	0.41	0.39	0.39	0.45
j10c5d3	18	21	66	68	63	68	63	73	71	0.03	0.11	0.49	0.51	0.53	0.36	0.47	0.40	0.36
j10c5d4	19	19	63	68	66	70	71	65	66	0.00	0.08	0.40	0.45	0.50	0.32	0.36	0.33	0.37
j10c5d5	19	19	61	58	65	55	58	56	53	0.02	0.11	0.43	0.46	0.52	0.41	0.29	0.25	0.43
j10c5d6	20	21	57	54	54	61	57	52	53	0.04	0.24	0.56	0.49	0.56	0.36	0.40	0.47	0.44
j10c10a1	18	21	63	52	55	58	56	59	61	0.26	0.42	0.42	0.48	0.34	0.44	0.42	0.44	0.44
j10c10a2	21	21	81	81	88	76	77	86	82	0.12	0.36	0.33	0.31	0.26	0.17	0.28	0.17	0.21
j10c10a3	21	21	68	68	74	76	76	76	73	0.19	0.29	0.55	0.54	0.58	0.52	0.46	0.54	0.49
j10c10a4	20	21	51	49	48	40	46	37	48	0.35	0.81	0.27	0.27	0.27	0.27	0.31	0.27	0.23
j10c10a5	19	21	58	67	60	56	63	59	54	0.22	0.46	0.46	0.52	0.50	0.39	0.46	0.52	0.57
j10c10a6	19	21	57	64	58	53	59	61	54	0.39	0.51	0.59	0.56	0.61	0.51	0.49	0.56	0.56
j10c10b1	21	21	86	86	86	86	86	86	86	0.24	0.24	1.00	1.00	1.00	1.00	1.00	1.00	1.00
j10c10b2	21	21	59	59	59	60	60	60	60	0.27	0.38	0.96	0.91	0.95	0.89	0.89	0.89	0.89
j10c10b3	21	21	96	95	96	95	95	95	95	0.15	0.22	0.99	0.99	1.00	0.99	0.99	0.99	0.99
j10c10b4	20	20	78	81	78	81	81	81	81	0.06	0.23	0.99	0.95	0.99	0.95	0.95	0.95	0.95
j10c10b5	20	21	81	80	80	82	82	82	82	0.13	0.26	0.97	0.92	0.96	0.92	0.92	0.92	0.92
j10c10b6	21	21	64	64	64	64	64	64	64	0.23	0.33	1.00	0.98	1.00	0.98	0.98	0.97	0.98
j10c10c1	17	20	57	60	57	56	52	57	59	0.00	0.13	0.59	0.49	0.56	0.31	0.36	0.43	0.46
j10c10c2	16	18	59	62	56	65	75	64	63	0.02	0.25	0.41	0.34	0.41	0.34	0.23	0.25	0.27
j10c10c3	19	19	54	54	56	57	59	55	65	0.00	0.12	0.57	0.48	0.48	0.53	0.32	0.42	0.42
j10c10c4	16	18	50	54	56	49	49	53	55	0.00	0.07	0.46	0.41	0.48	0.21	0.30	0.31	0.33
j10c10c5	19	18	57	54	58	54	58	60	53	0.03	0.19	0.53	0.43	0.40	0.29	0.40	0.41	0.29
j10c10c6	14	20	49	52	49	53	55	56	55	0.02	0.15	0.56	0.50	0.58	0.44	0.50	0.48	0.38
Average	**19.51**	**20.46**	**71.63**	**72.10**	**72.12**	**72.02**	**72.15**	**72.37**	**72.29**	**0.14**	**0.27**	**0.69**	**0.67**	**0.69**	**0.62**	**0.61**	**0.62**	**0.62**

Table 7. 5. (Cont'd) Performance Comparison of Algorithms on Medium Instances with 10 Jobs for the EHFSP-V1

	IGD									Distribution Spacing (DS)								
Instance	**MILP**	**CP**	**E_{HFRN}**	**E_{HFR}**	**E_{HFN}**	**E**	**IG**	**IG_{ALL}**	**VBIH**	**MILP**	**CP**	**E_{HFRN}**	**E_{HFR}**	**E_{HFN}**	**E**	**IG**	**IG_{ALL}**	**VBIH**
j10c5a2	2.05	2.05	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.20	0.20	1.01	1.01	1.01	1.01	1.01	1.01	1.01
j10c5a3	1.80	1.80	0.02	0.00	0.00	0.00	0.00	0.00	0.05	0.21	0.21	1.24	1.23	1.23	1.23	1.23	1.23	1.31
j10c5a4	2.50	2.57	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.12	0.12	1.17	1.17	1.17	1.17	1.17	1.17	1.17
j10c5a5	2.38	2.38	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.11	0.11	1.44	1.44	1.44	1.44	1.44	1.44	1.44
j10c5a6	2.57	2.52	0.11	0.16	0.17	0.28	0.27	0.23	0.26	0.18	0.16	1.07	1.03	1.07	1.01	1.05	1.05	1.02
j10c5b1	2.32	2.32	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.08	0.08	1.09	1.09	1.09	1.09	1.09	1.09	1.09
j10c5b2	2.06	2.06	0.03	0.03	0.00	0.03	0.03	0.03	0.03	0.11	0.11	1.29	1.29	1.27	1.29	1.29	1.29	1.29
j10c5b3	2.03	1.98	0.01	0.01	0.01	0.02	0.02	0.02	0.02	0.14	0.20	1.05	1.05	1.05	1.04	1.04	1.04	1.04
j10c5b4	2.62	2.59	0.00	0.00	0.00	0.01	0.01	0.01	0.01	0.24	0.20	1.00	1.00	1.00	1.00	1.00	1.00	1.00
j10c5b5	2.53	2.53	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.13	0.13	1.14	1.14	1.14	1.14	1.14	1.14	1.14
j10c5b6	2.20	2.20	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.10	0.10	1.55	1.55	1.55	1.55	1.55	1.55	1.55
j10c5c1	2.99	1.91	0.47	0.60	0.47	0.56	0.80	0.65	0.61	0.54	0.27	1.10	1.09	1.10	1.27	1.37	1.08	1.33
j10c5c2	3.50	1.15	0.52	0.60	0.51	0.66	0.67	0.65	0.59	0.34	0.09	1.49	1.68	1.46	1.46	1.56	1.55	1.37
j10c5c3	2.80	1.53	0.78	0.81	0.76	1.16	1.09	1.16	1.10	0.36	0.33	1.00	0.99	1.07	0.89	1.04	0.89	0.93
j10c5c4	3.50	3.29	0.40	0.34	0.39	0.42	0.44	0.51	0.59	0.47	0.47	0.83	0.79	0.89	0.84	0.79	0.85	0.82
j10c5c5	3.35	1.85	0.42	0.51	0.55	0.65	0.71	0.64	0.64	0.15	0.13	0.75	0.79	0.85	0.81	1.09	0.97	0.88
j10c5c6	4.75	1.88	0.43	0.35	0.43	0.49	0.62	0.58	0.55	0.42	0.26	0.86	1.00	0.89	0.74	1.06	0.96	0.98
j10c5d1	2.74	2.06	0.54	0.53	0.90	0.82	0.99	0.62	0.68	0.32	0.23	0.83	0.72	1.10	0.92	1.16	1.04	1.02
j10c5d2	4.91	3.12	0.59	0.59	0.64	0.56	0.87	0.93	0.65	0.52	0.28	1.18	1.23	1.22	1.28	1.04	1.35	1.31
j10c5d3	4.00	2.92	0.52	0.41	0.44	0.53	0.61	0.53	0.51	0.30	0.45	1.02	1.05	1.04	0.96	1.43	1.08	1.07
j10c5d4	4.28	3.28	0.68	0.54	0.51	0.59	0.58	0.70	0.62	0.19	0.37	1.51	1.05	1.61	1.23	1.21	1.71	0.97
j10c5d5	4.07	3.62	0.67	0.62	0.60	0.83	0.84	0.97	0.65	0.29	0.29	1.44	1.54	1.61	1.48	1.60	1.50	1.16
j10c5d6	3.61	2.76	0.45	0.59	0.53	0.62	0.69	0.70	0.62	0.40	0.46	0.98	1.23	1.19	1.27	1.29	1.29	1.20
j10c10a1	6.12	2.96	1.18	1.27	1.66	1.60	1.64	1.38	1.27	0.34	0.24	1.00	0.99	0.90	1.01	1.11	1.14	0.99
j10c10a2	4.14	4.19	1.76	1.84	1.89	1.98	1.89	1.96	2.12	0.14	0.10	0.97	0.99	1.25	0.89	0.99	1.10	1.11
j10c10a3	4.37	3.83	0.90	0.99	0.77	0.90	1.04	0.91	0.96	0.13	0.16	0.96	1.00	0.95	0.94	1.01	1.04	0.95
j10c10a4	4.25	0.74	3.06	2.59	2.78	2.84	3.04	2.94	3.12	0.33	0.31	1.53	1.55	1.55	1.24	1.55	1.21	1.51
j10c10a5	4.45	2.15	1.10	0.86	0.89	1.13	1.13	1.02	0.84	0.25	0.25	1.28	1.40	1.23	1.39	1.44	1.36	1.34
j10c10a6	2.89	1.98	1.14	1.34	1.18	1.25	1.50	1.57	1.34	0.23	0.20	0.95	1.05	1.08	1.12	1.00	1.15	1.21
j10c10b1	3.67	3.62	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.31	0.30	1.78	1.78	1.78	1.78	1.78	1.78	1.78
j10c10b2	3.54	2.68	0.07	0.13	0.08	0.15	0.15	0.15	0.15	0.27	0.20	1.32	1.32	1.34	1.35	1.35	1.35	1.35
j10c10b3	5.16	3.23	0.00	0.01	0.00	0.01	0.01	0.01	0.01	0.29	0.18	1.68	1.68	1.68	1.68	1.68	1.68	1.68
j10c10b4	7.21	3.71	0.01	0.04	0.01	0.04	0.04	0.04	0.04	0.34	0.27	2.05	2.12	2.05	2.12	2.12	2.12	2.12
j10c10b5	5.81	2.80	0.03	0.08	0.05	0.09	0.09	0.09	0.09	0.27	0.30	1.93	1.90	1.93	1.94	1.94	1.94	1.94
j10c10b6	4.30	3.68	0.00	0.02	0.00	0.03	0.03	0.04	0.03	0.29	0.40	1.14	1.14	1.14	1.14	1.14	1.13	1.14
j10c10c1	8.08	6.43	0.50	0.64	0.53	1.04	0.79	0.69	0.61	0.29	0.56	1.34	1.38	1.37	1.39	1.30	1.32	1.38
j10c10c2	10.81	4.70	1.08	1.26	1.12	1.42	1.31	1.26	1.41	0.72	0.42	1.50	1.47	1.50	1.39	1.65	1.46	1.68
j10c10c3	7.85	6.25	0.58	0.92	0.71	0.67	0.82	0.95	0.75	0.23	0.50	1.33	1.81	1.63	1.39	1.50	1.85	1.48
j10c10c4	8.90	8.16	0.56	0.64	0.43	0.85	0.77	0.68	0.69	0.29	0.71	1.61	1.74	1.75	1.58	1.62	1.67	1.64
j10c10c5	6.85	6.76	0.59	0.72	0.71	0.84	0.80	0.88	0.86	0.30	0.28	1.53	1.64	1.58	1.55	1.65	1.74	1.48
j10c10c6	9.28	3.26	0.48	0.62	0.42	0.73	0.57	0.58	0.72	0.30	0.11	1.53	1.66	1.59	1.53	1.65	1.77	1.69
Average	**4.32**	**3.06**	**0.48**	**0.50**	**0.49**	**0.58**	**0.61**	**0.59**	**0.57**	**0.27**	**0.26**	**1.26**	**1.29**	**1.30**	**1.26**	**1.32**	**1.32**	**1.28**

Table 7. 6. *p*-values of Wilcoxon Signed-Rank Tests for Medium Instances with 10 Jobs for the EHFSP-V1

Pairs of Algorithms	Cardinality	C_p	IGD	DS	Pairs of Algorithms	Cardinality	C_p	IGD	DS
MILP vs CP	0.00	0.00	0.00	0.33	E_{HFRN} vs IG	0.59	0.00	0.00	0.00
MILP vs E_{HFRN}	0.00	0.00	0.00	0.00	E_{HFRN} vs IG_{ALL}	0.15	0.00	0.00	0.00
MILP vs E_{HFR}	0.00	0.00	0.00	0.00	E_{HFRN} vs VBIH	0.27	0.00	0.00	0.04
MILP vs E_{HFN}	0.00	0.00	0.00	0.00	E_{HFR} vs E_{HFN}	0.98	0.01	0.19	1.00
MILP vs E	0.00	0.00	0.00	0.00	E_{HFR} vs E	0.89	0.00	0.00	0.16
MILP vs IG	0.00	0.00	0.00	0.00	E_{HFR} vs IG	0.83	0.00	0.00	0.12
MILP vs IG_{ALL}	0.00	0.00	0.00	0.00	E_{HFR} vs IG_{ALL}	0.55	0.00	0.00	0.12
MILP vs VBIH	0.00	0.00	0.00	0.00	E_{HFR} vs VBIH	0.78	0.00	0.00	0.99
CP vs E_{HFRN}	0.00	0.00	0.00	0.00	E_{HFN} vs E	0.99	0.00	0.00	0.05
CP vs E_{HFR}	0.00	0.00	0.00	0.00	E_{HFN} vs IG	0.94	0.00	0.00	0.35
CP vs E_{HFN}	0.00	0.00	0.00	0.00	E_{HFN} vs IG_{ALL}	0.56	0.00	0.00	0.18
CP vs E	0.00	0.00	0.00	0.00	E_{HFN} vs VBIH	0.77	0.00	0.00	0.91
CP vs IG	0.00	0.00	0.00	0.00	E vs IG	0.54	0.44	0.16	0.00
CP vs IG_{ALL}	0.00	0.00	0.00	0.00	E vs IG_{ALL}	0.48	0.65	0.86	0.00
CP vs VBIH	0.00	0.00	0.00	0.00	E vs VBIH	0.42	0.97	0.86	0.09
E_{HFRN} vs E_{HFR}	0.28	0.01	0.05	0.02	IG vs IG_{ALL}	0.76	0.08	0.33	0.85
E_{HFRN} vs E_{HFN}	0.30	0.53	0.76	0.00	IG vs VBIH	0.78	0.15	0.04	0.06
E_{HFRN} vs E	0.44	0.00	0.00	0.73	IG_{ALL} vs VBIH	0.70	0.82	0.23	0.13

Table 7.7 reports the results for each solution approach on the medium instances with 15 jobs. Table 7.8 also reports the *p*-value results of the Wilcoxon signed-rank tests for these instances. As shown in Table 7.7, each metaheuristic algorithm finds approximately seven times as many non-dominated solutions as the time-limited MILP and CP, in very short computation times. The *p*-value results reported in Table 7.8 also verify that all metaheuristics perform significantly better than the MILP and CP in terms of the cardinality metric. Note that, there is no statistically significant difference between the E_EM_{HFRN}, E_EM_{HFR} and E_EM_{HFN} algorithms in terms of cardinality. As shown in Table 7.7, E_IG finds 49%; E_EM_{HFR} finds 46%; E_EM_{HFN} finds 45%; E_EM_{HFRN} finds 44%; E_EM finds 37%; E_VBIH finds 35%; E_IG_{ALL} finds 30%; time-limited CP finds 9% and time-limited MILP finds 3% of the reference solutions on the overall average. As it can be seen in Table 7.8, all metaheuristics perform significantly and statistically better than the MILP and CP in terms of C_p metric, where E_IG, E_EM_{HFR}, E_EM_{HFN} and E_EM_{HFRN} algorithms outperform the other metaheuristics. Note that, all the pairwise differences are statistically significant at the $\alpha = 0.05$ level in terms of cardinality and C_p metrics, except E_EM_{HFRN} vs E_EM_{HFR}, E_EM_{HFRN} vs E_EM_{HFN}, E_EM_{HFR} vs E_EM_{HFN} and E_EM vs. E_VBIH pairs.

In terms of proximity to the reference frontier, E_IG (0.64), E_EM$_{\mathrm{HFN}}$ (0.67), E_EM$_{\mathrm{HFR}}$ (0.67) and E_EM$_{\mathrm{HFRN}}$ (0.69) are the closest ones in overall average, whereas other metaheuristic algorithms also have small IGD values. Similar to the results on instances with 10 jobs, all metaheuristic algorithms outperform the time-limited MILP and CP approaches in terms of IGD metric. These statements are also consistent with the Wilcoxon signed-rank test results reported in Table 7.8. Note that, there is no statistically significant difference between E_EM$_{\mathrm{HFRN}}$, E_EM$_{\mathrm{HFR}}$, E_EM$_{\mathrm{HFN}}$ and E_IG algorithms in terms of IGD metric, except the E_EM$_{\mathrm{HFRN}}$ vs. E_IG pair. In terms of the spread of the solutions, even though metaheuristic algorithms have low DS values, time-limited MILP and CP approaches have smaller DS values than these algorithms due to the usage of a constant ε level, which is also confirmed by the Wilcoxon signed-rank test results reported in Table 7.8. In order to visualize the performance of the algorithms, the Pareto frontiers obtained by the algorithms are provided for an instance with 15 jobs and 5 stages in Figure 7.14. As seen in Figure 7.14, the metaheuristic algorithms outperform the time-limited MILP and CP approaches in terms of both cardinality and convergence, where the time-limited CP performs better than the time-limited MILP.

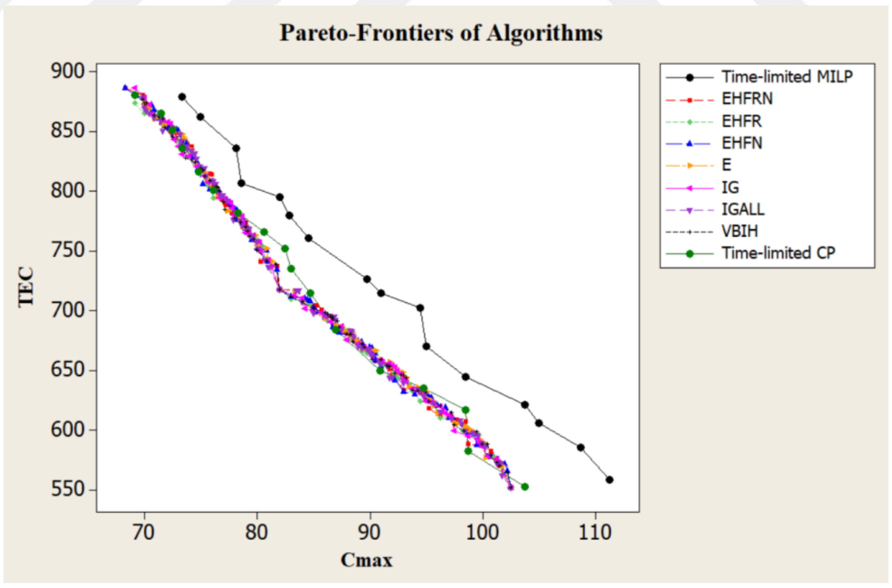


Figure 7. 14. Comparison of Algorithms for an Instance with 15 Jobs

Consequently, it is clear from Tables 7.5 and 7.7, ensembles of metaheuristic algorithms with HFR/HFN approaches perform better than the other metaheuristic algorithms for the medium instances, due to their higher C_p and lower IGD values. Note that, E_IG also performs very well for the instances with 15 jobs. Nevertheless, all metaheuristic algorithms perform much better than the time-limited MILP and CP approaches, in terms of both cardinality and proximity to the reference set.

Table 7. 7. Performance Comparison of Algorithms on Medium Instances with 15 Jobs for the EHFSP-V1

	Cardinality									**Ratio of Reference Solutions Found (C_p)**								
Instance	**MILP**	**CP**	**E_{HFRN}**	**E_{HFR}**	**E_{HFN}**	**E**	**IG**	**IG_{ALL}**	**VBIH**	**MILP**	**CP**	**E_{HFRN}**	**E_{HFR}**	**E_{HFN}**	**E**	**IG**	**IG_{ALL}**	**VBIH**
j15c5a1	21	21	201	202	174	192	199	180	195	0.03	0.09	0.29	0.32	0.36	0.23	0.32	0.25	0.22
j15c5a2	19	21	206	214	221	207	217	181	212	0.03	0.08	0.56	0.58	0.57	0.41	0.73	0.35	0.51
j15c5a3	20	21	149	156	160	147	160	134	146	0.08	0.12	0.40	0.47	0.50	0.39	0.55	0.33	0.33
j15c5a4	19	21	158	156	171	147	171	146	146	0.04	0.10	0.55	0.41	0.52	0.35	0.58	0.27	0.33
j15c5a5	21	21	167	164	159	164	184	160	151	0.07	0.11	0.48	0.52	0.45	0.40	0.63	0.35	0.42
j15c5a6	21	21	187	198	207	189	207	177	191	0.04	0.10	0.58	0.56	0.48	0.43	0.66	0.32	0.44
j15c5b1	20	21	243	256	255	232	244	218	227	0.05	0.07	0.86	0.87	0.86	0.79	0.91	0.57	0.74
j15c5b2	17	21	247	247	241	239	245	227	223	0.06	0.08	0.85	0.89	0.86	0.83	0.88	0.54	0.64
j15c5b3	21	21	197	200	201	203	198	190	195	0.08	0.10	0.89	0.91	0.90	0.85	0.90	0.68	0.86
j15c5b4	18	21	140	142	138	143	137	134	141	0.04	0.14	0.94	0.92	0.90	0.73	0.85	0.78	0.67
j15c5b5	19	21	223	222	218	216	221	204	212	0.01	0.09	0.87	0.80	0.88	0.67	0.92	0.57	0.62
j15c5b6	21	21	239	236	230	224	241	232	236	0.06	0.07	0.87	0.89	0.89	0.71	0.88	0.56	0.66
j15c5c1	17	19	93	77	99	91	88	79	93	0.00	0.13	0.14	0.21	0.19	0.17	0.20	0.17	0.13
j15c5c2	19	22	86	90	78	87	86	95	84	0.00	0.22	0.12	0.14	0.19	0.14	0.17	0.12	0.23
j15c5c3	17	20	95	98	97	90	107	87	93	0.00	0.09	0.19	0.18	0.21	0.11	0.10	0.14	0.18
j15c5c4	18	21	82	90	88	101	102	86	80	0.00	0.12	0.15	0.16	0.15	0.08	0.13	0.13	0.19
j15c5c5	19	17	75	66	68	74	68	80	75	0.00	0.04	0.28	0.17	0.23	0.15	0.21	0.11	0.19
j15c5c6	18	20	87	83	95	93	93	96	93	0.01	0.12	0.24	0.22	0.14	0.11	0.24	0.10	0.19
j15c5d1	20	21	179	187	196	183	184	184	176	0.06	0.09	0.62	0.72	0.67	0.55	0.81	0.43	0.43
j15c5d2	15	19	79	85	88	88	82	84	84	0.00	0.04	0.17	0.21	0.20	0.19	0.16	0.18	0.12
j15c5d3	16	17	79	73	81	81	81	84	71	0.00	0.04	0.17	0.20	0.19	0.14	0.14	0.18	0.17
j15c5d4	14	17	76	78	79	85	82	89	83	0.00	0.06	0.11	0.22	0.19	0.14	0.13	0.18	0.17
j15c5d5	16	21	74	81	86	87	78	83	83	0.00	0.06	0.18	0.19	0.19	0.13	0.27	0.19	0.14
j15c5d6	15	18	87	74	75	77	86	84	84	0.00	0.04	0.07	0.16	0.17	0.21	0.15	0.12	0.20
j15c10a1	20	21	252	254	243	237	262	214	243	0.05	0.08	0.81	0.79	0.75	0.63	0.78	0.41	0.55
j15c10a2	19	21	138	144	134	140	154	133	131	0.02	0.13	0.38	0.37	0.45	0.35	0.51	0.23	0.26
j15c10a3	18	21	176	176	183	166	191	174	177	0.02	0.08	0.43	0.37	0.32	0.27	0.51	0.25	0.20
j15c10a4	21	21	217	219	207	192	229	170	190	0.02	0.07	0.49	0.54	0.48	0.43	0.63	0.31	0.31
j15c10a5	17	21	124	123	117	112	122	118	117	0.02	0.12	0.37	0.43	0.45	0.27	0.42	0.26	0.23
j15c10a6	17	21	147	136	140	137	133	130	132	0.03	0.12	0.42	0.58	0.50	0.38	0.61	0.30	0.30
j15c10b1	19	21	172	150	150	134	152	146	148	0.04	0.05	0.34	0.34	0.33	0.28	0.39	0.25	0.28
j15c10b2	20	21	114	127	114	104	120	103	116	0.02	0.11	0.55	0.44	0.45	0.38	0.44	0.23	0.30
j15c10b3	18	21	146	131	148	129	152	121	118	0.03	0.11	0.48	0.43	0.48	0.40	0.60	0.39	0.43
j15c10b4	20	21	149	168	155	149	175	133	156	0.04	0.09	0.46	0.57	0.48	0.31	0.51	0.28	0.33
j15c10b5	20	20	150	138	132	140	132	126	139	0.01	0.05	0.30	0.29	0.23	0.26	0.30	0.20	0.18
j15c10b6	21	21	143	148	134	136	153	135	133	0.02	0.08	0.37	0.31	0.33	0.29	0.39	0.19	0.34
Average	**18.64**	**20.42**	**149.36**	**149.69**	**148.94**	**144.89**	**153.78**	**139.36**	**143.72**	**0.03**	**0.09**	**0.44**	**0.46**	**0.45**	**0.37**	**0.49**	**0.30**	**0.35**

Table 7. 7. (Cont'd) Performance Comparison of Algorithms on Medium Instances with 15 Jobs for the EHFSP-V1

	IGD									**Distribution Spacing (DS)**								
Instance	**MILP**	**CP**	**E_{HFRN}**	**E_{HFR}**	**E_{HFN}**	**E**	**IG**	**IG_{ALL}**	**VBIH**	**MILP**	**CP**	**E_{HFRN}**	**E_{HFR}**	**E_{HFN}**	**E**	**IG**	**IG_{ALL}**	**VBIH**
j15c5a1	4.46	4.28	0.55	0.53	0.49	0.63	0.49	0.67	0.64	0.15	0.17	1.15	1.07	1.12	1.14	1.06	1.10	1.11
j15c5a2	4.68	3.68	0.23	0.21	0.21	0.33	0.14	0.37	0.28	0.32	0.20	1.17	1.29	1.27	1.25	1.20	1.17	1.28
j15c5a3	4.22	3.52	0.44	0.37	0.33	0.49	0.30	0.54	0.55	0.25	0.25	1.06	1.13	1.15	1.11	1.13	1.08	1.05
j15c5a4	4.53	3.92	0.29	0.46	0.44	0.56	0.32	0.73	0.53	0.32	0.20	0.98	1.02	1.03	0.92	1.02	0.95	0.97
j15c5a5	4.39	4.38	0.47	0.39	0.46	0.53	0.28	0.54	0.55	0.16	0.19	1.15	1.13	1.18	1.28	1.26	1.13	1.20
j15c5a6	4.47	4.10	0.23	0.23	0.30	0.34	0.18	0.52	0.35	0.16	0.14	1.08	1.14	1.14	1.07	1.24	1.05	1.05
j15c5b1	4.53	4.08	0.08	0.06	0.06	0.11	0.06	0.23	0.13	0.19	0.16	1.32	1.42	1.29	1.27	1.36	1.26	1.25
j15c5b2	5.39	3.46	0.05	0.04	0.07	0.07	0.04	0.19	0.14	0.31	0.17	1.47	1.48	1.50	1.42	1.50	1.46	1.40
j15c5b3	3.89	3.85	0.05	0.03	0.05	0.07	0.03	0.17	0.09	0.19	0.18	1.38	1.41	1.40	1.44	1.39	1.30	1.43
j15c5b4	5.09	4.12	0.04	0.05	0.06	0.18	0.06	0.13	0.22	0.32	0.20	1.21	1.24	1.24	1.24	1.15	1.16	1.22
j15c5b5	5.48	3.62	0.06	0.07	0.05	0.15	0.03	0.25	0.19	0.22	0.14	1.41	1.41	1.35	1.31	1.36	1.30	1.32
j15c5b6	4.09	4.03	0.05	0.05	0.05	0.15	0.05	0.24	0.18	0.23	0.26	1.43	1.43	1.46	1.42	1.45	1.48	1.44
j15c5c1	9.13	4.51	1.31	1.25	1.07	1.19	1.38	1.51	1.35	0.43	0.22	0.99	0.89	1.00	1.11	1.11	1.01	0.95
j15c5c2	7.36	3.81	1.77	1.77	1.62	1.70	1.66	1.75	1.45	0.27	0.24	1.17	1.05	1.07	1.07	1.11	0.98	1.07
j15c5c3	9.31	4.10	1.13	1.03	1.18	1.26	1.23	1.61	1.18	0.28	0.39	1.22	1.02	1.24	1.07	1.14	1.13	1.01
j15c5c4	5.65	3.58	1.32	1.29	1.63	1.56	1.46	1.63	1.47	0.30	0.15	0.62	0.93	0.89	1.14	0.83	0.75	0.77
j15c5c5	8.01	5.76	1.12	1.43	1.16	1.23	1.27	1.62	1.35	0.44	0.44	1.03	1.00	0.96	0.91	0.93	1.21	1.03
j15c5c6	6.05	3.95	1.09	1.21	1.25	1.32	0.95	1.24	1.20	0.27	0.32	1.12	1.17	1.19	1.13	1.16	1.09	1.24
j15c5d1	4.44	3.85	0.25	0.17	0.21	0.25	0.12	0.34	0.35	0.12	0.10	0.97	1.03	1.01	0.94	0.95	0.94	0.94
j15c5d2	9.69	5.74	1.33	1.15	1.11	1.24	1.45	1.19	1.33	0.66	0.32	0.94	0.99	0.91	1.09	0.95	1.11	0.99
j15c5d3	9.39	5.89	1.38	1.51	1.23	1.97	1.44	1.55	1.74	0.31	0.32	1.09	1.03	0.93	1.11	0.97	0.97	0.85
j15c5d4	10.15	5.56	1.63	1.54	1.23	1.65	1.53	1.46	1.51	0.33	0.25	0.80	1.10	0.85	1.13	0.96	1.00	1.08
j15c5d5	10.35	4.10	1.50	1.36	1.11	1.42	1.27	1.42	1.54	0.88	0.17	0.84	0.90	0.99	0.86	0.82	1.09	0.88
j15c5d6	9.97	5.24	1.19	1.28	1.42	1.42	1.02	1.47	1.13	0.52	0.34	0.84	1.00	1.00	0.88	0.97	1.05	1.10
j15c10a1	9.04	7.53	0.11	0.22	0.19	0.28	0.16	0.51	0.35	0.25	0.19	1.66	1.70	1.62	1.64	1.71	1.64	1.81
j15c10a2	10.96	7.47	0.86	0.68	0.69	0.87	0.54	1.18	1.03	0.24	0.24	1.67	1.58	1.54	1.42	1.71	1.55	1.48
j15c10a3	12.50	7.53	0.70	0.73	0.79	0.88	0.51	1.11	1.17	0.34	0.15	1.35	1.27	1.25	1.33	1.35	1.24	1.23
j15c10a4	7.94	7.08	0.45	0.38	0.39	0.40	0.28	0.57	0.65	0.24	0.18	1.78	1.69	1.63	1.46	1.79	1.48	1.49
j15c10a5	17.59	7.50	1.17	0.75	0.75	1.21	0.94	1.26	1.39	0.55	0.24	1.45	1.42	1.35	1.29	1.58	1.42	1.26
j15c10a6	12.02	7.12	0.47	0.43	0.50	0.59	0.41	0.78	0.77	0.42	0.23	1.68	1.65	1.73	1.76	1.62	1.59	1.59
j15c10b1	9.10	7.15	0.59	0.64	0.72	0.78	0.60	0.99	0.80	0.31	0.17	1.64	1.64	1.58	1.52	1.68	1.82	1.58
j15c10b2	10.58	6.82	0.31	0.44	0.41	0.53	0.46	0.76	0.69	0.33	0.24	1.71	1.93	1.90	1.77	1.88	1.80	1.91
j15c10b3	10.79	7.88	0.53	0.59	0.65	0.75	0.43	0.79	0.69	0.19	0.20	1.60	1.65	1.79	1.60	1.75	1.69	1.54
j15c10b4	10.38	7.61	0.47	0.27	0.42	0.62	0.42	0.70	0.57	0.22	0.23	1.49	1.83	1.60	1.57	1.67	1.31	1.61
j15c10b5	10.38	9.53	0.97	0.95	0.96	1.29	0.92	1.31	1.17	0.34	0.19	1.58	1.60	1.42	1.48	1.51	1.40	1.49
j15c10b6	8.69	6.99	0.64	0.59	0.77	0.90	0.55	0.96	0.77	0.19	0.18	1.40	1.39	1.31	1.41	1.46	1.37	1.38
Average	**7.91**	**5.37**	**0.69**	**0.67**	**0.67**	**0.80**	**0.64**	**0.90**	**0.82**	**0.31**	**0.22**	**1.26**	**1.30**	**1.27**	**1.27**	**1.30**	**1.25**	**1.25**

Table 7. 8. *p*-values of Wilcoxon Signed-Rank Tests for Medium Instances with 15 Jobs for the EHFSP-V1

Pairs of Algorithms	Cardinality	C_p	IGD	DS	Pairs of Algorithms	Cardinality	C_p	IGD	DS
MILP vs CP	0.00	0.00	0.00	0.00	E_{HFRN} vs IG	0.01	0.00	0.02	0.05
MILP vs E_{HFRN}	0.00	0.00	0.00	0.00	E_{HFRN} vs IG_{ALL}	0.00	0.00	0.00	0.41
MILP vs E_{HFR}	0.00	0.00	0.00	0.00	E_{HFRN} vs VBIH	0.00	0.00	0.00	0.47
MILP vs E_{HFN}	0.00	0.00	0.00	0.00	E_{HFR} vs E_{HFN}	0.68	0.37	0.63	0.16
MILP vs E	0.00	0.00	0.00	0.00	E_{HFR} vs E	0.01	0.00	0.00	0.09
MILP vs IG	0.00	0.00	0.00	0.00	E_{HFR} vs IG	0.01	0.02	0.06	0.96
MILP vs IG_{ALL}	0.00	0.00	0.00	0.00	E_{HFR} vs IG_{ALL}	0.00	0.00	0.00	0.06
MILP vs VBIH	0.00	0.00	0.00	0.00	E_{HFR} vs VBIH	0.00	0.00	0.00	0.01
CP vs E_{HFRN}	0.00	0.00	0.00	0.00	E_{HFN} vs E	0.03	0.00	0.00	0.35
CP vs E_{HFR}	0.00	0.00	0.00	0.00	E_{HFN} vs IG	0.02	0.01	0.24	0.15
CP vs E_{HFN}	0.00	0.00	0.00	0.00	E_{HFN} vs IG_{ALL}	0.00	0.00	0.00	0.09
CP vs E	0.00	0.00	0.00	0.00	E_{HFN} vs VBIH	0.01	0.00	0.00	0.09
CP vs IG	0.00	0.00	0.00	0.00	E vs IG	0.00	0.00	0.00	0.11
CP vs IG_{ALL}	0.00	0.00	0.00	0.00	E vs IG_{ALL}	0.00	0.00	0.00	0.40
CP vs VBIH	0.00	0.00	0.00	0.00	E vs VBIH	0.43	0.13	0.38	0.48
E_{HFRN} vs E_{HFR}	0.57	0.24	0.28	0.20	IG vs IG_{ALL}	0.00	0.00	0.00	0.01
E_{HFRN} vs E_{HFN}	0.90	0.57	0.76	0.64	IG vs VBIH	0.00	0.00	0.00	0.04
E_{HFRN} vs E	0.02	0.00	0.00	0.86	IG_{ALL} vs VBIH	0.04	0.00	0.00	0.97

Table 7.9 reports the results for $\mathrm{E\_EM_{HFRN}}\ (E_{HFRN})$, $\mathrm{E\_EM_{HFR}}\ (E_{HFR})$, $\mathrm{E\_EM_{HFN}}\ (E_{HFN})$, $\mathrm{E\_EM}\ (E)$, $\mathrm{E\_IG}\ (IG)$, $\mathrm{E\_IG_{ALL}} (IG_{ALL})$ and $\mathrm{E\_VBIH}\ (VBIH)$ algorithms on large instances. Table 7.10 also reports the *p*-value results of the Wilcoxon signed-rank tests for these instances. As shown in Table 7.9, ensembles of metaheuristic algorithms generate more non-dominated solutions than E_IG, $\mathrm{E\_IG_{ALL}}$ and E_VBIH algorithms, which is also confirmed by the Wilcoxon signed-rank test results reported in Table 7.10. Furthermore, $\mathrm{E\_EM_{HFRN}}$ finds 18%; $\mathrm{E\_EM_{HFN}}$, $\mathrm{E\_EM_{HFR}}$ and E_IG find 17%; E_VBIH finds 15%; E_EM finds 13%; and $\mathrm{E\_IG_{ALL}}$ finds 10% of the reference solutions on the overall average. According to the *p*-value results reported in Table 7.10, there is no statistically significant difference between the $\mathrm{E\_EM_{HFRN}}$, $\mathrm{E\_EM_{HFR}}$, $\mathrm{E\_EM_{HFN}}$ and E_IG algorithms in terms of C_p metric and they perform statistically better than the E_EM and $\mathrm{E\_IG_{ALL}}$ algorithms.

In terms of convergence, $\mathrm{E\_EM_{HFRN}}$ has the lowest IGD value in overall average, whereas $\mathrm{E\_EM_{HFR}}$, $\mathrm{E\_EM_{HFN}}$, and E_EM also have small IGD values. It can be said that ensembles of metaheuristic algorithms with HFR/HFN approaches outperform the other metaheuristic algorithms in terms of IGD metric. This statement is also consistent with the Wilcoxon signed-rank test results reported in Table 7.10. Note that, $\mathrm{E\_EM}_{\mathrm{HFRN}}$, $\mathrm{E\_EM}_{\mathrm{HFR}}$ and $\mathrm{E\_EM}_{\mathrm{HFN}}$ algorithms are statistically equivalent in terms of IGD metric. For the comparison between E_IG and E_EM algorithms on C_p and IGD metrics, the results are quite interesting. The E_IG algorithm outperforms the E_EM algorithm in the C_p metric while the E_EM algorithm outperforms E_IG in the IGD metric. This result indicates that the solution set of E_EM is closer to the reference frontier while the E_IG algorithm has more reference solutions in its own frontier. In terms of distribution spacing metric, all metaheuristic algorithms have low DS values, which indicates even dispersions. Note that, in terms of DS metric, there is a statistically significant difference between only $\mathrm{E\_EM}_{\mathrm{HFRN}}$ vs. E_IG, $\mathrm{E\_EM}_{\mathrm{HFRN}}$ vs. $\mathrm{E\_IG}_{\mathrm{ALL}}$, $\mathrm{E\_EM}_{\mathrm{HFR}}$ vs. E_IG, $\mathrm{E\_EM}_{\mathrm{HFN}}$ vs. E_EM, $\mathrm{E\_EM}_{\mathrm{HFN}}$ vs. E_IG and $\mathrm{E\_EM}_{\mathrm{HFN}}$ vs. $\mathrm{E\_IG}_{\mathrm{ALL}}$ pairs.

Table 7. 9. Performance Comparison of Algorithms on Large Instances for the EHFSP-V1

	Cardinality							**Ratio of Reference Solutions Found (C_p)**						
Instance	**EHFRN**	**EHFR**	**EHFN**	**E**	**IG**	**IGALL**	**VBIH**	**EHFRN**	**EHFR**	**EHFN**	**E**	**IG**	**IGALL**	**VBIH**
j30c5e1	129	124	124	119	99	107	103	0.12	0.14	0.07	0.17	0.24	0.13	0.12
j30c5e2	123	119	124	114	94	91	123	0.10	0.17	0.16	0.07	0.20	0.05	0.29
j30c5e3	128	137	137	111	115	115	127	0.18	0.15	0.14	0.14	0.21	0.15	0.15
j30c5e4	143	131	131	150	134	96	123	0.13	0.14	0.18	0.24	0.10	0.11	0.12
j30c5e5	140	133	130	117	119	122	130	0.12	0.17	0.16	0.13	0.24	0.19	0.07
j30c5e6	156	162	152	158	149	150	149	0.16	0.19	0.13	0.24	0.11	0.09	0.15
j30c5e7	147	152	111	117	139	142	151	0.11	0.09	0.34	0.15	0.17	0.02	0.19
j30c5e8	138	141	150	147	151	147	148	0.24	0.16	0.19	0.15	0.13	0.05	0.15
j30c5e9	136	144	159	140	137	119	144	0.11	0.15	0.12	0.13	0.24	0.12	0.12
j30c5e10	106	92	109	108	119	93	95	0.17	0.24	0.15	0.15	0.16	0.09	0.14
j40c5e1	152	141	150	137	147	131	130	0.15	0.12	0.12	0.13	0.21	0.08	0.26
j40c5e2	168	164	168	168	146	119	156	0.33	0.12	0.16	0.11	0.10	0.10	0.18
j40c5e3	148	165	150	154	124	112	140	0.23	0.13	0.22	0.20	0.07	0.09	0.12
j40c5e4	163	152	134	134	145	128	136	0.12	0.10	0.19	0.12	0.22	0.08	0.20
j40c5e5	157	118	153	147	130	100	142	0.18	0.22	0.11	0.12	0.22	0.11	0.12
j40c5e6	169	157	134	110	135	125	138	0.11	0.17	0.15	0.10	0.25	0.04	0.24
j40c5e7	147	164	139	157	137	102	138	0.17	0.13	0.11	0.12	0.12	0.08	0.32
j40c5e8	180	177	168	179	156	136	153	0.25	0.06	0.19	0.08	0.31	0.05	0.15
j40c5e9	214	220	185	205	194	180	182	0.21	0.19	0.21	0.06	0.16	0.06	0.19
j40c5e10	170	185	178	164	153	146	158	0.19	0.17	0.13	0.11	0.10	0.15	0.17
j50c5e1	194	200	237	214	167	166	160	0.15	0.40	0.22	0.08	0.07	0.09	0.07
j50c5e2	227	231	229	206	156	158	170	0.15	0.15	0.29	0.05	0.11	0.27	0.05
j50c5e3	218	223	229	206	186	174	191	0.21	0.16	0.18	0.09	0.20	0.08	0.11
j50c5e4	220	214	212	205	155	157	148	0.26	0.19	0.16	0.02	0.13	0.14	0.16
j50c5e5	194	177	197	164	165	123	151	0.23	0.20	0.22	0.08	0.17	0.11	0.08
j50c5e6	203	204	213	209	196	141	160	0.14	0.19	0.26	0.05	0.25	0.14	0.06
j50c5e7	171	185	168	150	131	127	131	0.22	0.29	0.18	0.17	0.06	0.09	0.14
j50c5e8	155	174	173	165	155	108	172	0.21	0.21	0.20	0.10	0.23	0.13	0.05
j50c5e9	185	166	140	142	139	99	116	0.18	0.15	0.31	0.10	0.13	0.13	0.10
j50c5e10	154	136	145	127	135	125	117	0.30	0.30	0.17	0.19	0.03	0.10	0.08
j60c5e1	243	203	206	203	136	145	164	0.20	0.19	0.21	0.15	0.17	0.05	0.18
j60c5e2	213	206	194	226	150	150	159	0.16	0.20	0.11	0.22	0.14	0.07	0.25
j60c5e3	169	185	154	190	154	114	123	0.19	0.13	0.27	0.14	0.08	0.05	0.23
j60c5e4	172	169	184	153	128	127	144	0.10	0.06	0.19	0.12	0.21	0.07	0.26
j60c5e5	205	198	211	206	192	148	152	0.19	0.12	0.11	0.11	0.25	0.13	0.15
j60c5e6	170	174	156	181	124	140	126	0.29	0.20	0.10	0.10	0.19	0.01	0.20
j60c5e7	210	200	194	202	171	182	169	0.14	0.14	0.16	0.28	0.15	0.07	0.07
j60c5e8	159	172	165	168	122	122	125	0.18	0.20	0.12	0.19	0.19	0.04	0.22
j60c5e9	185	169	171	200	175	109	141	0.16	0.16	0.12	0.14	0.24	0.11	0.13
j60c5e10	210	200	199	208	176	137	164	0.07	0.24	0.14	0.17	0.19	0.07	0.15
Average	**171.78**	**169.10**	**166.58**	**164.03**	**145.90**	**130.33**	**143.73**	**0.18**	**0.17**	**0.17**	**0.13**	**0.17**	**0.10**	**0.15**

Table 7. 9. (Cont'd) Performance Comparison of Algorithms on Large Instances for the EHFSP-V1

	IGD							Distribution Spacing (DS)						
Instance	**EHFRN**	**EHFR**	**EHFN**	**E**	**IG**	**IGALL**	**VBIH**	**EHFRN**	**EHFR**	**EHFN**	**E**	**IG**	**IGALL**	**VBIH**
j30c5e1	11.07	12.94	12.23	12.85	17.15	22.49	27.78	0.95	0.94	0.84	1.03	0.95	1.04	1.57
j30c5e2	12.50	13.69	12.63	15.90	19.28	20.65	12.19	0.99	1.05	1.12	0.84	1.31	0.94	0.88
j30c5e3	12.20	11.08	13.82	15.11	13.85	23.81	16.23	1.04	0.99	1.09	0.80	1.09	1.23	1.41
j30c5e4	9.99	10.96	10.62	10.13	20.04	24.34	15.86	1.05	0.92	0.74	0.96	0.95	1.20	0.95
j30c5e5	11.64	12.15	11.65	14.14	11.93	14.12	17.88	0.86	1.02	0.87	0.96	0.98	0.99	1.04
j30c5e6	8.58	8.15	10.31	7.70	10.52	13.65	9.10	0.88	0.95	0.90	0.89	1.15	0.99	1.15
j30c5e7	10.98	10.30	9.18	11.85	11.86	22.61	10.04	0.94	0.90	0.85	0.91	0.95	1.23	1.07
j30c5e8	8.54	11.21	10.76	10.58	10.52	15.63	16.81	0.90	1.08	0.94	0.94	1.07	0.98	1.11
j30c5e9	11.12	11.26	11.16	12.02	8.97	22.00	13.07	0.93	0.85	1.09	1.10	1.09	0.98	0.92
j30c5e10	14.58	16.48	16.49	18.68	16.69	29.51	29.03	0.90	1.24	1.06	1.42	1.32	1.08	1.02
j40c5e1	12.32	13.07	13.04	17.90	27.43	55.86	46.43	1.05	0.89	0.84	1.36	1.43	1.15	1.85
j40c5e2	11.11	16.02	11.63	15.21	20.83	67.88	41.17	1.01	0.96	0.93	0.98	1.39	0.98	1.08
j40c5e3	17.68	18.21	17.71	14.26	50.68	74.46	36.52	1.09	1.31	1.13	1.09	1.30	1.02	1.13
j40c5e4	12.44	16.74	17.55	16.31	45.62	52.26	32.37	0.90	1.08	1.14	0.91	1.38	1.08	0.98
j40c5e5	16.05	17.63	14.10	16.02	25.85	51.32	42.65	1.18	0.88	0.95	1.02	0.97	1.30	1.04
j40c5e6	16.33	21.34	17.89	28.66	40.63	71.31	38.33	0.87	1.10	0.85	1.15	1.03	0.98	2.53
j40c5e7	20.83	14.24	17.15	18.34	42.21	71.72	28.55	0.91	1.11	1.01	1.12	1.08	0.92	0.93
j40c5e8	12.97	14.21	14.35	13.74	28.54	70.71	48.05	1.11	1.14	1.10	1.01	1.02	1.14	1.02
j40c5e9	8.74	10.40	9.93	12.25	22.93	59.20	35.53	0.94	1.05	0.90	0.97	1.12	1.07	1.16
j40c5e10	11.41	12.22	11.24	16.13	23.76	52.87	32.88	1.02	1.07	0.99	1.51	1.15	0.96	0.95
j50c5e1	13.99	9.58	15.59	20.33	95.16	85.08	82.35	1.02	1.07	1.16	1.20	0.99	1.10	1.02
j50c5e2	18.42	13.39	20.67	29.67	137.75	96.39	102.49	1.12	0.99	1.21	1.07	1.01	1.36	0.90
j50c5e3	19.93	14.76	14.80	31.44	65.90	114.85	68.21	1.09	0.98	1.05	1.19	1.02	1.21	0.94
j50c5e4	14.15	14.28	18.39	21.63	72.88	103.27	87.69	1.15	1.11	1.03	1.03	1.33	1.00	1.16
j50c5e5	19.41	13.01	20.27	23.93	61.70	110.27	85.25	1.02	0.89	1.00	1.53	1.25	1.31	1.11
j50c5e6	26.69	18.77	11.26	16.88	72.09	121.70	55.06	1.09	0.95	1.04	0.93	1.01	0.90	1.61
j50c5e7	19.70	15.01	25.48	16.34	90.44	114.15	107.11	1.01	1.02	0.96	1.01	1.02	1.12	0.98
j50c5e8	19.50	13.52	20.22	30.42	43.24	115.51	83.42	1.07	0.83	1.01	1.11	1.08	1.64	1.01
j50c5e9	13.67	17.91	13.29	28.28	100.02	115.39	121.56	1.19	1.05	1.07	1.70	1.13	1.35	1.03
j50c5e10	14.89	24.18	24.37	26.12	93.84	111.61	120.49	1.12	0.98	1.31	1.38	1.12	1.06	0.99
j60c5e1	18.57	26.54	19.95	38.96	152.46	146.75	123.07	0.88	0.95	0.94	0.95	1.27	1.18	1.10
j60c5e2	16.60	23.76	36.67	24.57	143.06	155.25	160.80	1.13	0.96	1.33	1.15	1.15	1.72	1.04
j60c5e3	38.24	27.19	39.02	28.43	139.23	250.27	225.73	0.90	1.40	0.87	1.09	1.26	0.76	1.00
j60c5e4	20.54	43.49	29.58	29.37	138.26	175.03	114.48	1.09	1.20	0.99	1.09	0.93	1.70	0.94
j60c5e5	16.56	22.46	22.85	26.17	107.62	172.91	157.29	1.13	1.03	0.98	1.22	0.99	1.12	1.03
j60c5e6	35.60	28.28	29.98	33.97	146.84	175.70	136.55	1.08	1.15	1.10	0.96	1.46	0.93	0.89
j60c5e7	25.25	20.14	26.18	24.24	139.38	188.07	168.53	1.03	0.92	1.02	1.00	1.41	1.15	1.09
j60c5e8	48.14	24.08	42.17	29.49	157.37	202.05	136.32	1.04	1.49	1.17	1.40	0.91	1.14	1.03
j60c5e9	28.89	54.61	26.65	27.60	87.17	211.01	176.90	1.78	1.12	1.07	1.02	1.21	0.95	1.38
j60c5e10	25.00	22.88	22.80	27.78	153.75	218.97	182.05	1.60	1.73	1.24	1.13	1.10	0.97	2.21
Average	**17.62**	**18.00**	**18.59**	**20.84**	**66.69**	**95.52**	**76.15**	**1.05**	**1.06**	**1.02**	**1.10**	**1.13**	**1.12**	**1.16**

Table 7. 10. *p*-values of Wilcoxon Signed-Rank Tests for Large Instances for the EHFSP-V1

Pairs of Algorithms	**Cardinality**	***C_p***	**IGD**	**DS**	**Pairs of Algorithms**	**Cardinality**	***C_p***	**IGD**	**DS**
E_{HFRN} vs E_{HFR}	0.34	0.56	0.79	0.86	E_{HFN} vs E	0.30	0.01	0.01	0.05
E_{HFRN} vs E_{HFN}	0.07	0.69	0.08	0.53	E_{HFN} vs IG	0.00	0.89	0.00	0.00
E_{HFRN} vs E	0.02	0.00	0.00	0.14	E_{HFN} vs IG_{ALL}	0.00	0.00	0.00	0.01
E_{HFRN} vs IG	0.00	0.86	0.00	0.02	E_{HFN} vs VBIH	0.00	0.28	0.00	0.10
E_{HFRN} vs IG_{ALL}	0.00	0.00	0.00	0.01	E vs IG	0.00	0.03	0.00	0.51
E_{HFRN} vs VBIH	0.00	0.11	0.00	0.25	E vs IG_{ALL}	0.00	0.00	0.00	0.81
E_{HFR} vs E_{HFN}	0.39	0.89	0.61	0.29	E vs VBIH	0.00	0.10	0.00	0.87
E_{HFR} vs E	0.07	0.01	0.00	0.44	IG vs IG_{ALL}	0.00	0.00	0.00	0.61
E_{HFR} vs IG	0.00	0.79	0.00	0.01	IG vs VBIH	0.64	0.40	0.04	0.31
E_{HFR} vs IG_{ALL}	0.00	0.00	0.00	0.17	IG_{ALL} vs VBIH	0.00	0.00	0.00	0.67
E_{HFR} vs VBIH	0.00	0.33	0.00	0.23					

7.5 Small Instances for the EHFSP-V2

Table 7.11 reports the results of C_p, IGD and DS performance metrics for each metaheuristic algorithm on the small-sized instances, where $E2$, $IG2$, $IG2_{ALL}$ and $VBIH2$ represent E_EM2, E_IG2, E_IG2_{ALL} and E_VBIH2 algorithms, respectively. As shown in the table, E_IG2_{ALL} finds 48%; E_VBIH2 finds 47%; E_EM2 finds 46% and E_IG2 finds 43% of the Pareto-optimal solutions on the overall average. In terms of convergence, E_IG2_{ALL} is the best performer with 0.64 IGD value in overall average whereas E_VBIH2 and E_EM2 also have very small (around 0.69) IGD values. However, it can be said that all algorithms provide very close approximations to the Pareto-optimal solution set P, as the maximum of their average IGD values is 0.78. In terms of distribution spacing, solutions obtained by the metaheuristic algorithms are evenly distributed due to their low DS values. Note that, E_VBIH2 has a slightly lower DS value than the other algorithms. Finally, as the E_IG2_{ALL} algorithm has higher C_p and lower IGD values, it can be said that E_IG2_{ALL} performs slightly better than the other algorithms for these small instances.

Table 7. 11. Performance Comparison of Algorithms on Small Instances for the EHFSP-V2

Instance	Cardinality				Ratio of Pareto-Optimal Solutions Found (C_p)				IGD				Distribution Spacing (DS)			
	E2	IG2	IG2$_{ALL}$	VBIH2	E2	IG2	IG2$_{ALL}$	VBIH2	E2	IG2	IG2$_{ALL}$	VBIH2	E2	IG2	IG2$_{ALL}$	VBIH2
j5c5a2	74	73	74	76	0.16	0.17	0.21	0.18	0.90	1.03	1.08	0.98	4.19	3.05	5.26	3.53
j5c5a3	88	90	96	105	0.58	0.49	0.62	0.67	0.49	0.46	0.40	0.49	1.62	1.88	3.22	1.17
j5c5a4	102	99	103	108	0.56	0.55	0.57	0.59	0.45	0.56	0.45	0.43	1.26	2.36	0.89	0.56
j5c5a5	70	71	64	53	0.00	0.01	0.01	0.01	3.02	2.96	2.64	2.80	1.10	1.64	1.06	1.22
j5c5a6	103	96	101	96	0.80	0.77	0.81	0.74	0.17	0.25	0.15	0.25	0.59	1.43	1.22	1.06
j5c5b1	128	125	139	114	0.55	0.54	0.58	0.54	0.45	0.59	0.40	0.55	0.72	0.82	1.14	1.07
j5c5b2	114	104	107	105	0.54	0.48	0.55	0.52	0.50	0.66	0.38	0.48	2.42	1.10	1.25	0.85
j5c5b3	103	113	104	101	0.35	0.31	0.36	0.36	0.69	0.82	0.54	0.71	3.34	3.54	2.42	3.00
j5c5b4	104	96	103	108	0.48	0.47	0.48	0.52	0.48	0.58	0.46	0.44	0.91	0.93	0.79	0.91
j5c5b5	123	113	123	121	0.76	0.69	0.78	0.73	0.30	0.43	0.23	0.31	0.87	0.83	0.97	1.06
j5c5b6	115	103	123	117	0.46	0.42	0.46	0.47	0.48	0.67	0.47	0.47	0.94	1.24	1.35	0.90
j5c5c1	86	84	85	85	0.32	0.28	0.31	0.28	0.44	0.40	0.44	0.37	1.55	0.76	1.36	1.88
Average	**100.83**	**97.25**	**101.83**	**99.08**	**0.46**	**0.43**	**0.48**	**0.47**	**0.70**	**0.78**	**0.64**	**0.69**	**1.63**	**1.63**	**1.74**	**1.43**

7.6 Medium & Large Instances for the EHFSP-V2

As mentioned in the beginning of Section 7, for medium and large instances, the non-dominated solution sets of time-limited MILP, time-limited CP and metaheuristic algorithms are compared with each other in terms of the aforementioned cardinality, C_p, IGD and DS metrics. As the Pareto-optimal solution sets (P) are not known for these instances, the reference sets (R) are used in C_p and IGD metrics. Note that the reference set includes only the high-quality non-dominated solutions, which are obtained by selecting all the non-dominated solutions found by the four metaheuristic algorithms, time-limited MILP and CP approaches.

In order to make the computational results statistically convincing, a series of Kruskal Wallis tests is also conducted at the significance level of $\alpha = 0.05$. The Kruskal Wallis test is a non-parametric test to determine if there are statistically significant differences between two or more groups, i.e., there is a statistically significant difference between at least one pair of groups. This test is employed to decide whether there is a statistically significant difference between at least two solution approaches in terms of a certain performance metric. Following a Kruskal Wallis test, the Dunn test is also carried out on each pair of algorithms as a post-hoc analysis procedure. Namely, the Dunn test is employed to make multiple pairwise comparisons, if there is a statistically significant Kruskal Wallis result. For each pair of the algorithms, the results of the Kruskal Wallis and Dunn tests are reported for all performance metrics.

Table 7.12 reports the results for the time-limited MILP (MILP), time-limited CP (CP), E_EM2($E2$), E_IG2 ($IG2$), E_IG2$_{\text{ALL}}$ ($IG2_{ALL}$) and E_VBIH2 ($VBIH2$) algorithms on medium instances with 10 jobs. Furthermore, Table 7.13 reports the results of statistical tests for these instances. As shown in Table 7.12, each metaheuristic algorithm finds approximately six times as many non-dominated solutions than the time-limited MILP and CP, in exceptionally fewer computation times. The results of statistical tests reported in Table 7.13 also confirm that all metaheuristics perform significantly better than the MILP and CP in terms of the cardinality metric. Note that there is no statistically significant difference between the metaheuristics in terms of cardinality.

As shown in Table 7.12, E_IG2$_{\text{ALL}}$ finds 41%; E_IG2 finds 35%; E_EM2 and E_VBIH2 find 33%; time-limited CP finds 22%, and the time limited MILP finds 16% of the reference solutions on the overall average. Note that, E_IG2$_{\text{ALL}}$ finds more than 80% of the reference solutions for 10 out of 41 instances, where E_VBIH2 and E_EM2 find at least 80% of the reference solutions for 8 instances; E_IG2 finds more than 80% of the reference solutions for 5 instances. According to the results of statistical tests reported in Table 7.13, there is a statistically significant difference between only pairs of MILP-E_EM2, MILP-E_IG2 and MILP-E_IG2$_{\text{ALL}}$ in terms of C_p metric.

In terms of convergence, E_IG2 (3.43) and E_EM2 (3.47) are the best performer ones on the overall average, whereas other metaheuristic algorithms also have small IGD values around 3.57. According to the results of the statistical tests reported in Table 7.13, there is no statistically significant difference between the solution methods in terms of IGD metric for these instances with 10 jobs. For the comparison of distribution spacing metric, time-limited MILP and CP approaches have smaller DS values than the metaheuristic algorithms, which implies that the solutions generated by MILP and CP approaches are spread more uniformly in their own discovered frontiers. This statement is also consistent with the results of the statistical tests reported in Table 7.13. This result is expected, as a constant ε level is used through the augmented ε-constraint method in the time-limited MILP and CP approaches. Nevertheless, the metaheuristic algorithms also have low DS values, indicating even dispersions.

Table 7. 12. Performance Comparison of Algorithms on Medium Instances with 10 Jobs for the EHFSP-V2

Instance	Cardinality						Ratio of Reference Solutions Found (C_p)					
	MILP	CP	E2	IG2	IG2$_{ALL}$	VBIH2	MILP	CP	E2	IG2	IG2$_{ALL}$	VBIH2
j10c5a2	18	19	139	120	130	117	0.09	0.17	0.80	0.77	0.80	0.81
j10c5a3	7	19	159	157	158	173	0.05	0.13	0.83	0.83	0.83	0.87
j10c5a4	21	20	158	164	166	168	0.14	0.13	0.82	0.79	0.88	0.86
j10c5a5	4	18	165	163	179	163	0.02	0.11	0.86	0.83	0.92	0.83
j10c5a6	21	19	141	140	144	146	0.16	0.15	0.84	0.82	0.86	0.85
j10c5b1	21	19	176	178	187	178	0.12	0.11	0.88	0.86	0.92	0.81
j10c5b2	5	21	143	144	151	148	0.04	0.17	0.79	0.78	0.82	0.86
j10c5b3	8	18	157	146	157	159	0.05	0.13	0.80	0.77	0.80	0.78
j10c5b4	19	21	147	149	155	150	0.13	0.15	0.76	0.68	0.75	0.68
j10c5b5	21	21	195	200	210	186	0.11	0.11	0.88	0.87	0.92	0.85
j10c5b6	20	20	155	160	159	160	0.13	0.13	0.79	0.79	0.83	0.76
j10c5c1	15	17	90	85	73	96	0.07	0.19	0.33	0.35	0.38	0.36
j10c5c2	14	15	85	88	87	86	0.06	0.15	0.23	0.21	0.32	0.37
j10c5c3	17	18	93	88	93	88	0.10	0.22	0.19	0.10	0.33	0.24
j10c5c4	16	16	74	80	65	73	0.06	0.17	0.30	0.30	0.47	0.36
j10c5c5	15	14	78	85	95	89	0.09	0.12	0.31	0.29	0.46	0.31
j10c5c6	15	16	83	90	78	69	0.08	0.17	0.16	0.19	0.41	0.30
j10c5d1	15	15	78	79	84	69	0.12	0.16	0.25	0.24	0.28	0.22
j10c5d2	16	16	68	82	67	69	0.13	0.18	0.15	0.17	0.27	0.27
j10c5d3	16	15	82	85	81	83	0.09	0.15	0.16	0.35	0.18	0.19
j10c5d4	16	20	92	97	83	87	0.08	0.23	0.30	0.44	0.21	0.17
j10c5d5	15	14	81	85	72	89	0.11	0.17	0.33	0.36	0.41	0.28
j10c5d6	16	15	76	74	73	75	0.06	0.15	0.21	0.32	0.37	0.26
j10c10a1	13	7	43	57	53	47	0.55	0.20	0.10	0.15	0.00	0.05
j10c10a2	14	11	70	66	63	63	0.41	0.59	0.00	0.00	0.00	0.00
j10c10a3	12	6	63	64	55	60	0.38	0.17	0.00	0.08	0.17	0.25
j10c10a4	11	8	40	49	50	59	0.33	0.17	0.04	0.21	0.25	0.04
j10c10a5	6	9	50	44	42	38	0.12	0.36	0.08	0.16	0.12	0.20
j10c10a6	16	5	54	53	42	60	0.65	0.13	0.17	0.09	0.00	0.00
j10c10b1	16	15	59	62	62	55	0.36	0.54	0.00	0.04	0.11	0.04
j10c10b2	9	14	74	83	83	65	0.06	0.40	0.00	0.11	0.31	0.17
j10c10b3	11	12	55	63	66	65	0.32	0.50	0.00	0.00	0.18	0.05
j10c10b4	9	11	72	83	56	71	0.08	0.42	0.17	0.25	0.08	0.00
j10c10b5	9	16	91	86	72	73	0.11	0.43	0.03	0.00	0.43	0.03
j10c10b6	15	10	61	79	77	65	0.39	0.39	0.17	0.04	0.00	0.04
j10c10c1	14	17	106	100	114	108	0.08	0.17	0.19	0.27	0.04	0.27
j10c10c2	15	15	94	88	71	92	0.02	0.26	0.04	0.15	0.45	0.08
j10c10c3	14	14	102	99	103	109	0.11	0.12	0.09	0.33	0.31	0.10
j10c10c4	15	16	116	106	92	101	0.08	0.13	0.27	0.20	0.27	0.05
j10c10c5	16	17	104	100	100	92	0.10	0.11	0.21	0.25	0.32	0.02
j10c10c6	18	19	87	105	100	93	0.15	0.30	0.15	0.09	0.26	0.04
Average	**14.24**	**15.32**	**98.93**	**100.63**	**98.73**	**98.46**	**0.16**	**0.22**	**0.33**	**0.35**	**0.41**	**0.33**

Table 7. 12. (Cont'd) Performance Comparison of Algorithms on Medium Instances with 10 Jobs for the EHFSP-V2

Instance	IGD						Distribution Spacing (DS)					
	MILP	**CP**	**E2**	**IG2**	**$IG2_{ALL}$**	**VBIH2**	**MILP**	**CP**	**E2**	**IG2**	**$IG2_{ALL}$**	**VBIH2**
j10c5a2	1.18	1.28	0.49	0.50	0.47	0.41	0.44	0.68	3.98	2.28	1.03	0.91
j10c5a3	3.84	1.19	0.35	0.39	0.32	0.35	0.71	0.49	8.16	2.80	1.56	1.24
j10c5a4	0.98	1.22	0.34	0.43	0.35	0.32	0.35	0.51	2.86	8.00	1.86	3.18
j10c5a5	6.25	1.62	0.23	0.30	0.19	0.27	0.31	0.75	2.89	4.39	2.00	1.66
j10c5a6	0.96	1.55	0.55	0.72	0.55	0.50	0.38	0.68	1.09	1.50	1.54	1.75
j10c5b1	0.96	1.60	0.27	0.26	0.24	0.33	0.30	0.70	1.40	6.40	8.52	3.48
j10c5b2	6.01	1.05	0.58	0.62	0.48	0.59	0.53	0.42	2.61	3.02	3.13	2.74
j10c5b3	5.55	1.96	0.68	0.68	0.65	0.68	0.60	0.51	2.37	3.13	2.09	3.85
j10c5b4	1.90	1.84	1.16	1.34	1.17	1.14	0.47	0.46	2.15	2.15	2.33	1.96
j10c5b5	1.13	1.13	0.30	0.33	0.28	0.30	0.29	0.29	8.26	2.06	1.07	7.49
j10c5b6	1.08	1.32	0.74	0.85	0.71	0.94	0.40	0.67	3.06	2.71	2.16	2.57
j10c5c1	3.41	2.71	1.31	1.50	2.52	1.13	1.32	0.45	1.80	1.42	1.20	1.59
j10c5c2	4.55	2.68	1.52	1.37	1.60	1.40	1.35	0.37	0.85	1.26	1.54	0.89
j10c5c3	4.31	2.59	2.25	1.96	2.77	2.65	1.01	0.47	1.14	1.01	1.11	1.24
j10c5c4	5.04	4.17	1.41	1.84	2.86	1.96	0.95	0.38	1.26	0.97	0.84	1.02
j10c5c5	4.08	2.91	0.97	1.01	1.06	0.98	0.40	0.43	1.09	1.46	1.81	1.13
j10c5c6	3.39	3.56	1.50	1.80	1.72	2.00	0.46	0.51	1.49	1.01	1.35	1.30
j10c5d1	3.28	3.55	2.50	2.17	2.75	2.42	0.10	0.42	0.82	0.98	1.48	1.07
j10c5d2	4.39	3.38	3.61	1.81	2.58	2.50	1.13	0.74	1.12	1.83	1.01	0.94
j10c5d3	4.17	2.56	1.72	1.61	1.81	1.91	1.09	0.54	1.25	0.81	1.86	1.13
j10c5d4	5.85	2.33	2.24	1.71	2.30	1.78	1.02	0.47	0.81	0.89	1.73	1.76
j10c5d5	4.64	4.59	1.54	1.49	2.69	2.11	0.80	0.11	1.14	0.99	1.12	1.08
j10c5d6	4.14	3.45	1.42	1.24	1.60	1.68	1.07	0.60	1.31	1.82	1.44	1.21
j10c10a1	1.69	6.28	6.02	6.11	7.63	6.40	0.32	0.77	2.60	2.61	1.55	1.53
j10c10a2	2.21	3.08	11.21	11.63	11.92	11.46	0.87	0.87	1.10	0.82	1.50	2.03
j10c10a3	4.40	8.75	7.39	6.32	6.28	6.04	0.81	1.38	2.35	1.45	1.35	0.98
j10c10a4	6.56	7.91	7.12	6.16	6.38	6.99	1.31	1.16	1.06	1.75	1.19	1.18
j10c10a5	6.54	5.84	5.85	6.29	5.92	5.41	0.66	1.09	1.40	2.07	1.27	1.10
j10c10a6	0.70	8.61	9.26	10.07	9.90	9.88	0.79	1.28	1.47	1.17	1.97	2.19
j10c10b1	1.91	2.90	10.97	10.87	10.02	11.17	0.68	1.53	1.71	2.19	1.80	1.57
j10c10b2	9.73	5.50	4.92	5.15	5.01	5.21	0.75	0.95	1.88	2.17	2.04	1.52
j10c10b3	3.59	3.01	7.86	8.17	7.98	7.80	0.34	0.71	1.62	2.68	1.66	2.21
j10c10b4	6.25	4.96	4.93	4.38	5.01	5.27	0.28	0.50	1.84	1.75	1.57	1.86
j10c10b5	5.89	5.14	5.88	5.56	4.81	5.47	0.25	0.73	2.13	2.06	1.94	1.60
j10c10b6	2.69	3.79	7.60	7.82	8.36	8.05	0.68	0.60	2.28	1.77	1.93	1.90
j10c10c1	13.25	6.11	2.91	3.03	4.29	3.42	0.92	0.74	1.37	1.32	1.27	1.24
j10c10c2	10.39	7.50	5.18	4.16	3.60	4.75	1.19	0.41	1.38	2.08	1.47	1.60
j10c10c3	11.17	10.15	3.79	3.61	5.43	5.45	0.98	0.86	1.41	1.58	1.01	1.00
j10c10c4	14.33	7.49	3.16	2.86	3.46	3.97	0.87	0.86	1.50	1.39	1.39	1.22
j10c10c5	11.36	7.70	4.40	3.27	2.86	3.59	1.29	0.70	1.12	2.00	1.58	2.33
j10c10c6	8.38	3.12	6.17	9.43	7.49	6.31	1.01	0.38	2.30	1.13	1.62	1.29
Average	**4.93**	**3.95**	**3.47**	**3.43**	**3.61**	**3.54**	**0.72**	**0.66**	**2.03**	**2.07**	**1.75**	**1.79**

Table 7. 13. Results of Statistical Tests for Medium Instances with 10 Jobs for the EHFSP-V2

	Cardinality	C_p	**IGD**	**DS**
***p*-value of Kruskal Wallis Test**	0.00	0.00	0.09	0.00
Pairwise Comparisons				
MILP vs CP	N	N	-	N
MILP vs E2	Y	Y	-	Y
MILP vs IG2	Y	Y	-	Y
MILP vs $IG2_{ALL}$	Y	Y	-	Y
MILP vs VBIH2	Y	N	-	Y
CP vs E2	Y	N	-	Y
CP vs IG2	Y	N	-	Y
CP vs $IG2_{ALL}$	Y	N	-	Y
CP vs VBIH2	Y	N	-	Y
E2 vs IG2	N	N	-	N
E2 vs $IG2_{ALL}$	N	N	-	N
E2 vs VBIH2	N	N	-	N
IG2 vs $IG2_{ALL}$	N	N	-	N
IG2 vs VBIH2	N	N	-	N
$IG2_{ALL}$ vs VBIH2	N	N	-	N
Y: significant (p-value $\leq$ 0.05)	**N:** not significant (p-value $>$ 0.05)			

Table 7.14 reports the results for each solution approach on the medium instances with 15 jobs. Furthermore, Table 7.15 reports the results of statistical tests for these instances. As shown in Table 7.14, each metaheuristic algorithm finds approximately eight times as many non-dominated solutions than the time-limited MILP and CP, in very short computation times. The statistical results reported in Table 7.15 also confirm that all metaheuristics perform significantly better than the MILP and CP in terms of the cardinality metric.

As shown in Table 7.14, E_$IG2_{ALL}$ finds 48%; E_IG2 finds 47%; E_EM2 finds 44%; E_VBIH2 finds 34%; time-limited CP finds 17%, and the time limited MILP finds 6% of the reference solutions on the overall average. Note that, E_$IG2_{ALL}$, E_IG2 and E_EM2 find more than 80% of the reference solutions for 12 out of 36 instances, where E_VBIH2 finds at least 80% of the reference solutions for 9 instances. According to the results of statistical tests reported in Table 7.15, all metaheuristics perform significantly and statistically better than the MILP in terms of C_p metric. Note that, E_$IG2_{ALL}$ and E_IG2 algorithms are statistically equivalent in terms of C_p metric and they perform statistically better than the MILP, CP and E_VBIH2.

In terms of proximity to the reference frontier, E_IG2 (3.01), E_EM2 (3.15) and $\text{E\_IG2}_{\text{ALL}}$ (3.26) are the closest ones in overall average, whereas E_VBIH2 also has small IGD value (3.73). According to the results of statistical tests reported in Table 7.15, all metaheuristic algorithms outperform the time limited MILP and CP approaches statistically in terms of IGD metric. Note that all metaheuristic algorithms are statistically equivalent in terms of IGD metric for these instances with 15 jobs. In terms of the spread of the solutions, even though metaheuristic algorithms have low DS values, time-limited MILP and CP approaches have smaller DS values than these algorithms due to the usage of a constant ε level. This statement is also consistent with the results of statistical tests reported in Table 7.15.

Consequently, it is clear from Tables 7.12 and 7.14, $\text{E\_IG2}_{\text{ALL}}$ and E_IG2 perform better than the other metaheuristic algorithms for these medium instances, due to their higher C_p and lower IGD values. Nevertheless, all metaheuristic algorithms perform much better than the time-limited MILP and CP approaches, in terms of both cardinality and proximity to the reference set, as expected.

Table 7. 14. Performance Comparison of Algorithms on Medium Instances with 15 Jobs for the EHFSP-V2

Instance	**Cardinality**						**Ratio of Reference Solutions Found (C_p)**					
	MILP	**CP**	**E2**	**IG2**	**IG2$_{ALL}$**	**VBIH2**	**MILP**	**CP**	**E2**	**IG2**	**IG2$_{ALL}$**	**VBIH2**
j15c5a1	20	18	229	224	226	214	0.09	0.08	0.82	0.79	0.82	0.79
j15c5a2	20	18	214	219	222	219	0.08	0.09	0.88	0.88	0.90	0.87
j15c5a3	21	16	181	171	178	168	0.11	0.10	0.83	0.83	0.84	0.77
j15c5a4	15	17	201	201	207	200	0.05	0.09	0.86	0.83	0.83	0.77
j15c5a5	22	19	215	219	213	216	0.11	0.09	0.87	0.86	0.92	0.85
j15c5a6	4	20	227	215	233	222	0.01	0.09	0.86	0.88	0.92	0.83
j15c5b1	20	21	222	229	220	214	0.07	0.10	0.85	0.86	0.89	0.82
j15c5b2	20	20	192	205	205	200	0.08	0.11	0.85	0.84	0.86	0.84
j15c5b3	6	21	206	207	203	203	0.03	0.12	0.83	0.80	0.88	0.80
j15c5b4	11	8	179	189	186	188	0.05	0.05	0.81	0.84	0.84	0.82
j15c5b5	14	20	203	207	212	194	0.04	0.12	0.80	0.81	0.87	0.83
j15c5b6	20	21	215	226	219	219	0.09	0.10	0.85	0.85	0.88	0.84
j15c5c1	15	15	102	101	94	83	0.05	0.19	0.44	0.21	0.13	0.11
j15c5c2	15	13	88	100	95	94	0.04	0.19	0.00	0.26	0.54	0.00
j15c5c3	13	14	98	102	98	91	0.00	0.14	0.48	0.21	0.25	0.10
j15c5c4	14	14	100	90	109	81	0.08	0.09	0.14	0.32	0.34	0.14
j15c5c5	16	16	94	98	79	91	0.00	0.17	0.24	0.28	0.28	0.28
j15c5c6	16	18	106	125	111	109	0.09	0.23	0.33	0.23	0.31	0.18
j15c5d1	18	16	223	239	233	220	0.06	0.05	0.76	0.81	0.77	0.75
j15c5d2	14	15	94	104	96	87	0.03	0.17	0.23	0.37	0.23	0.11
j15c5d3	13	19	89	79	95	77	0.00	0.15	0.32	0.45	0.01	0.13
j15c5d4	16	17	101	107	83	92	0.03	0.15	0.23	0.15	0.39	0.13
j15c5d5	16	17	102	105	87	89	0.00	0.18	0.28	0.33	0.31	0.08
j15c5d6	16	18	84	85	88	86	0.03	0.15	0.14	0.35	0.30	0.07
j15c10a1	10	10	67	57	47	52	0.15	0.77	0.00	0.00	0.08	0.00
j15c10a2	7	9	72	85	74	79	0.05	0.41	0.00	0.32	0.23	0.00
j15c10a3	10	8	67	72	70	64	0.05	0.36	0.09	0.23	0.14	0.14
j15c10a4	6	4	53	69	67	73	0.00	0.19	0.14	0.19	0.48	0.00
j15c10a5	6	7	76	63	81	74	0.05	0.15	0.15	0.35	0.30	0.03
j15c10a6	9	5	74	78	73	76	0.00	0.24	0.05	0.24	0.52	0.00
j15c10b1	8	6	47	46	53	49	0.15	0.19	0.26	0.22	0.19	0.00
j15c10b2	7	5	59	52	43	30	0.07	0.18	0.29	0.29	0.14	0.07
j15c10b3	5	3	40	42	40	37	0.06	0.09	0.16	0.38	0.22	0.13
j15c10b4	6	3	42	44	53	39	0.11	0.07	0.15	0.26	0.41	0.04
j15c10b5	6	6	55	61	63	62	0.05	0.16	0.54	0.08	0.19	0.00
j15c10b6	6	5	52	53	60	53	0.09	0.15	0.24	0.45	0.06	0.00
Average	**12.81**	**13.39**	**124.14**	**126.92**	**125.44**	**120.69**	**0.06**	**0.17**	**0.44**	**0.47**	**0.48**	**0.34**

Table 7. 14. (Cont'd) Performance Comparison of Algorithms on Medium Instances with 15 Jobs for the EHFSP-V2

Instance	IGD						Distribution Spacing (DS)					
	MILP	**CP**	**E2**	**IG2**	**IG2$_{ALL}$**	**VBIH2**	**MILP**	**CP**	**E2**	**IG2**	**IG2$_{ALL}$**	**VBIH2**
j15c5a1	1.85	2.61	0.78	0.83	0.83	0.77	0.36	0.71	2.94	2.27	2.25	1.65
j15c5a2	1.55	2.34	0.35	0.39	0.33	0.33	0.28	0.75	1.87	1.33	2.33	1.13
j15c5a3	1.01	2.88	0.55	0.61	0.54	0.51	0.37	0.79	2.37	1.84	2.48	1.70
j15c5a4	2.46	3.52	0.56	0.61	0.57	0.58	0.71	1.07	3.29	2.08	2.22	3.68
j15c5a5	1.23	2.06	0.39	0.45	0.33	0.42	0.38	0.70	2.23	2.45	1.93	2.22
j15c5a6	9.85	1.66	0.35	0.33	0.34	0.40	0.34	0.46	2.11	2.47	2.15	2.35
j15c5b1	1.50	1.32	0.62	0.63	0.53	0.64	0.39	0.32	2.67	2.53	2.41	2.76
j15c5b2	1.73	1.73	0.83	0.85	0.75	0.85	0.50	0.55	3.09	2.09	2.39	2.92
j15c5b3	8.79	1.50	0.76	0.81	0.69	0.82	0.90	0.40	2.68	8.11	2.35	2.40
j15c5b4	3.18	5.74	0.54	0.55	0.54	0.54	1.06	0.69	1.94	3.44	2.56	2.73
j15c5b5	2.41	2.58	0.94	0.95	0.86	0.87	0.36	0.63	3.14	2.77	2.65	1.70
j15c5b6	1.56	1.41	0.81	0.87	0.72	0.78	0.35	0.35	4.18	2.37	3.00	2.35
j15c5c1	8.84	4.46	2.01	3.35	3.16	3.39	0.68	0.58	1.53	1.34	0.80	1.12
j15c5c2	8.75	5.89	3.43	3.18	2.72	3.31	0.94	0.74	1.31	0.94	1.18	1.22
j15c5c3	10.62	4.87	2.42	1.37	2.11	1.62	1.45	1.47	1.58	1.41	1.05	1.59
j15c5c4	9.92	4.48	2.70	2.57	2.84	2.76	0.68	1.27	1.23	1.01	1.34	1.53
j15c5c5	8.32	4.85	1.46	1.16	1.91	1.46	0.80	1.04	1.05	0.98	1.09	1.42
j15c5c6	5.79	4.07	3.51	4.34	3.16	4.90	0.93	1.19	1.00	1.12	1.63	1.24
j15c5d1	1.68	6.28	0.45	0.44	0.45	0.39	0.33	1.51	1.79	1.29	2.30	1.91
j15c5d2	9.41	5.12	2.24	1.66	2.96	2.72	0.79	0.97	1.14	2.00	1.13	0.65
j15c5d3	11.83	4.01	2.01	2.13	2.88	2.32	0.76	0.77	1.44	1.05	1.31	1.54
j15c5d4	10.43	4.34	1.96	1.80	1.89	2.39	1.29	0.75	0.86	1.10	1.39	0.88
j15c5d5	10.67	3.83	2.07	1.85	2.35	2.80	1.25	0.70	1.09	1.53	1.14	0.73
j15c5d6	10.26	4.20	2.79	2.81	2.31	2.20	1.27	0.87	1.26	1.21	0.95	0.99
j15c10a1	6.86	1.80	22.10	21.74	20.31	22.95	0.50	1.74	1.57	1.63	1.29	1.50
j15c10a2	17.46	7.79	9.34	8.41	7.76	10.66	1.62	0.48	1.36	1.51	1.74	2.03
j15c10a3	14.92	8.66	6.94	6.32	7.30	8.26	1.81	0.48	1.58	1.27	1.99	1.58
j15c10a4	13.46	13.46	5.69	4.60	4.12	7.55	1.77	0.31	1.46	1.64	1.63	1.55
j15c10a5	14.53	8.73	3.83	3.20	3.09	5.43	1.43	0.56	1.77	1.73	2.26	1.64
j15c10a6	20.45	13.35	4.71	4.28	3.76	5.67	1.03	0.30	1.61	1.21	1.59	1.78
j15c10b1	11.04	16.27	5.70	4.94	7.34	7.85	0.64	0.69	1.28	1.25	1.54	1.15
j15c10b2	18.92	17.06	3.71	3.84	5.61	6.10	0.73	0.56	1.13	2.62	2.29	0.97
j15c10b3	18.61	24.18	3.02	3.07	3.04	4.62	0.63	0.79	1.20	1.18	1.19	2.33
j15c10b4	14.61	21.28	3.64	3.27	2.65	4.42	1.38	0.64	0.98	1.70	1.24	1.04
j15c10b5	19.19	23.13	6.46	6.52	10.65	6.91	0.79	1.43	1.66	2.19	1.12	2.27
j15c10b6	20.29	14.97	3.83	3.68	5.99	6.04	1.49	0.44	1.56	2.24	1.44	0.80
Average	**9.28**	**7.12**	**3.15**	**3.01**	**3.26**	**3.73**	**0.86**	**0.77**	**1.80**	**1.91**	**1.76**	**1.70**

Table 7. 15. Results of Statistical Tests for Medium Instances with 15 Jobs for the EHFSP-V2

	Cardinality	***C_p***	**IGD**	**DS**
***p*-value of Kruskal Wallis Test**	0.00	0.00	0.00	0.00
Pairwise Comparisons				
MILP vs CP	N	Y	N	N
MILP vs E2	Y	Y	Y	Y
MILP vs IG2	Y	Y	Y	Y
MILP vs IG2$_{\text{ALL}}$	Y	Y	Y	Y
MILP vs VBIH2	Y	Y	Y	Y
CP vs E2	Y	Y	Y	Y
CP vs IG2	Y	Y	Y	Y
CP vs IG2$_{\text{ALL}}$	Y	Y	Y	Y
CP vs VBIH2	Y	N	Y	Y
E2 vs IG2	N	N	N	N
E2 vs IG2$_{\text{ALL}}$	N	N	N	N
E2 vs VBIH2	N	N	N	N
IG2 vs IG2$_{\text{ALL}}$	N	N	N	N
IG2 vs VBIH2	N	Y	N	N
IG2$_{\text{ALL}}$ vs VBIH2	N	Y	N	N

Y: significant (p-value $\leq$ 0.05) **N:** not significant (p-value $>$ 0.05)

Table 7.16 reports the results for E_EM2 ($E2$), E_IG2 ($IG2$), E_IG2$_{\text{ALL}}$($IG2_{ALL}$) and E_VBIH2 ($VBIH2$) algorithms on large instances. Furthermore, Table 7.17 reports the results of statistical tests for these instances. As shown in Table 7.16, E_IG2 and E_EM2 algorithms generate more non-dominated solutions than E_IG2$_{\text{ALL}}$ and E_VBIH2 algorithms. According to the results of statistical tests reported in Table 7.17, E_IG2 outperforms the E_IG2$_{\text{ALL}}$ and E_VBIH2 algorithms statistically in terms of cardinality. Furthermore, E_IG2 finds 48%; E_IG2$_{\text{ALL}}$ finds 20%; E_EM2 finds 17%, and E_VBIH2 finds 16% of the reference solutions on the overall average. According to the results of statistical tests reported in Table 7.17, E_IG2 outperforms the other metaheuristics statistically in terms of C_p metric.

In terms of convergence, E_IG2 has the lowest IGD value in overall average, whereas E_IG2$_{\text{ALL}}$ and E_EM2 also have small IGD values. As shown in Table 7.17, all the pairwise differences are statistically significant at the $\alpha = 0.05$ level in terms of IGD metric, except the E_IG2$_{\text{ALL}}$ - E_EM2 pair. It can be said that the E_IG2 algorithm outperforms the other metaheuristic algorithms in terms of both C_p and IGD metrics. In terms of distribution spacing metric, all metaheuristic algorithms have low DS values, which indicates even dispersions. This statement is also consistent with the results of the statistical tests reported in Table 7.17.

Table 7. 16. Performance Comparison of Algorithms on Large Instances for the EHFSP-V2

Instance	**Cardinality**				**Ratio of Reference Solutions Found (C_p)**			
	E2	**IG2**	**IG2$_{ALL}$**	**VBIH2**	**E2**	**IG2**	**IG2$_{ALL}$**	**VBIH2**
j30c5e1	123	124	124	121	0.18	0.72	0.07	0.04
j30c5e2	99	111	103	94	0.17	0.54	0.24	0.05
j30c5e3	111	130	116	111	0.08	0.36	0.48	0.08
j30c5e4	134	141	136	136	0.08	0.57	0.15	0.22
j30c5e5	113	145	106	93	0.11	0.65	0.15	0.13
j30c5e6	144	151	133	117	0.19	0.72	0.02	0.06
j30c5e7	127	140	142	132	0.19	0.44	0.20	0.19
j30c5e8	135	151	133	110	0.19	0.61	0.12	0.11
j30c5e9	147	161	149	135	0.12	0.62	0.14	0.12
j30c5e10	151	158	121	127	0.15	0.59	0.17	0.10
j40c5e1	123	139	120	100	0.12	0.55	0.26	0.08
j40c5e2	117	149	135	126	0.15	0.39	0.26	0.21
j40c5e3	89	120	119	108	0.19	0.51	0.20	0.12
j40c5e4	145	154	119	136	0.11	0.44	0.35	0.11
j40c5e5	132	147	101	106	0.07	0.64	0.21	0.07
j40c5e6	123	157	142	119	0.28	0.39	0.22	0.11
j40c5e7	145	147	152	145	0.05	0.79	0.06	0.10
j40c5e8	138	164	139	140	0.09	0.55	0.21	0.16
j40c5e9	157	176	146	140	0.10	0.68	0.14	0.12
j40c5e10	136	169	118	131	0.23	0.55	0.07	0.14
j50c5e1	199	197	182	174	0.12	0.58	0.11	0.20
j50c5e2	183	158	160	181	0.21	0.41	0.25	0.15
j50c5e3	196	202	211	172	0.12	0.45	0.35	0.09
j50c5e4	193	187	156	159	0.07	0.32	0.41	0.20
j50c5e5	128	166	133	137	0.23	0.42	0.17	0.18
j50c5e6	160	186	145	140	0.16	0.34	0.23	0.28
j50c5e7	148	154	142	122	0.18	0.31	0.32	0.20
j50c5e8	128	155	114	135	0.26	0.46	0.24	0.07
j50c5e9	144	140	133	142	0.25	0.46	0.27	0.05
j50c5e10	101	122	112	106	0.15	0.26	0.36	0.25
j60c5e1	135	124	116	136	0.36	0.42	0.08	0.14
j60c5e2	189	212	196	201	0.21	0.63	0.09	0.08
j60c5e3	116	135	106	112	0.28	0.20	0.27	0.28
j60c5e4	146	200	158	173	0.13	0.50	0.18	0.19
j60c5e5	195	159	153	147	0.08	0.55	0.24	0.15
j60c5e6	106	117	128	138	0.14	0.23	0.14	0.50
j60c5e7	127	140	143	100	0.41	0.30	0.15	0.15
j60c5e8	145	134	134	130	0.30	0.24	0.10	0.36
j60c5e9	134	139	130	123	0.11	0.55	0.13	0.21
j60c5e10	194	182	180	169	0.14	0.31	0.22	0.33
Average	**141.40**	**153.58**	**137.15**	**133.10**	**0.17**	**0.48**	**0.20**	**0.16**

Table 7. 16. (Cont'd) Performance Comparison of Algorithms on Large Instances for the EHFSP-V2

Instance	IGD				Distribution Spacing (DS)			
	E2	**IG2**	**IG2$_{ALL}$**	**VBIH2**	**E2**	**IG2**	**IG2$_{ALL}$**	**VBIH2**
j30c5e1	12.28	5.36	15.83	22.45	1.09	1.10	1.50	1.57
j30c5e2	16.77	9.89	10.94	22.18	1.24	1.90	1.18	0.92
j30c5e3	20.45	13.37	10.44	25.70	1.33	1.18	0.97	1.19
j30c5e4	12.77	11.39	19.05	10.40	1.43	1.10	1.58	1.25
j30c5e5	19.46	7.80	16.07	27.05	1.37	1.30	1.40	1.16
j30c5e6	11.25	5.31	16.68	18.52	1.40	1.20	1.69	1.50
j30c5e7	12.38	10.04	11.46	20.32	1.22	2.04	1.40	1.33
j30c5e8	14.32	5.10	15.27	23.11	1.01	1.58	1.01	1.30
j30c5e9	11.52	7.17	12.46	15.43	1.05	1.26	1.83	1.32
j30c5e10	13.68	7.74	17.44	18.11	1.61	1.92	1.97	1.25
j40c5e1	19.11	8.45	14.06	33.67	1.35	1.67	1.38	1.14
j40c5e2	15.11	11.27	11.85	16.14	1.01	1.15	0.97	1.49
j40c5e3	28.19	8.90	22.13	28.32	1.63	1.47	1.80	1.33
j40c5e4	29.40	11.15	29.16	24.82	1.29	1.30	1.40	1.54
j40c5e5	20.57	5.45	25.36	32.98	1.59	1.37	1.19	1.20
j40c5e6	12.26	7.67	13.15	19.02	1.22	1.46	1.22	1.13
j40c5e7	22.72	4.86	19.73	26.28	1.08	1.52	1.22	1.22
j40c5e8	20.87	8.73	17.71	30.57	1.12	1.25	1.27	1.45
j40c5e9	20.61	6.62	34.76	30.17	1.27	1.25	1.49	1.21
j40c5e10	15.65	8.76	24.15	28.22	1.04	1.20	0.93	1.11
j50c5e1	19.72	3.91	21.81	21.50	1.07	1.13	1.23	1.08
j50c5e2	12.85	13.66	22.69	18.89	1.03	1.93	0.97	1.24
j50c5e3	18.56	16.71	11.39	28.85	1.33	0.96	1.18	1.07
j50c5e4	15.69	13.94	16.56	23.27	0.93	1.12	1.44	1.42
j50c5e5	31.49	16.07	24.35	39.41	1.05	1.48	1.42	0.93
j50c5e6	21.98	11.81	21.38	25.90	1.60	1.34	0.95	0.87
j50c5e7	23.55	16.51	19.12	49.40	1.22	1.58	0.99	1.03
j50c5e8	19.35	10.45	19.65	24.49	1.05	1.30	1.34	1.15
j50c5e9	16.37	11.52	19.23	29.32	1.33	1.15	1.30	1.34
j50c5e10	19.86	16.29	36.87	42.95	1.36	1.21	0.85	1.09
j60c5e1	24.98	12.91	32.24	36.28	0.97	1.36	1.82	1.13
j60c5e2	21.54	5.60	16.88	25.22	1.34	1.05	1.00	1.35
j60c5e3	17.86	22.54	19.99	24.57	1.28	0.93	1.17	0.95
j60c5e4	29.99	9.13	16.24	24.37	1.02	1.32	1.03	1.10
j60c5e5	21.25	9.14	15.28	25.92	1.03	1.24	1.28	1.03
j60c5e6	36.09	28.42	22.98	18.15	0.92	1.00	1.14	1.91
j60c5e7	13.14	15.86	18.98	49.27	1.25	1.23	1.27	1.15
j60c5e8	27.18	20.78	23.80	32.69	1.26	1.15	1.04	1.27
j60c5e9	27.15	9.02	24.59	57.56	1.21	1.11	1.12	1.09
j60c5e10	18.72	14.92	18.06	21.48	1.11	1.12	1.30	1.29
Average	**19.67**	**11.11**	**19.49**	**27.32**	**1.22**	**1.32**	**1.28**	**1.23**

Table 7. 17. Results of Statistical Tests for Large Instances for the EHFSP-V2

	Cardinality	C_p	**IGD**	**DS**
***p*-value of Kruskal Wallis Test**	0.00	0.00	0.00	0.38
Pairwise Comparisons				
E2 vs IG2	N	Y	Y	-
E2 vs $IG2_{ALL}$	N	N	N	-
E2 vs VBIH2	N	N	Y	-
IG2 vs $IG2_{ALL}$	Y	Y	Y	-
IG2 vs VBIH2	Y	Y	Y	-
$IG2_{ALL}$ vs VBIH2	N	N	Y	-
Y: significant (p-value $\leq$ 0.05)	**N:** not significant (p-value $>$ 0.05)			

CHAPTER 8
CONCLUSION

Hybrid flowshop is a well-known extension of the basic flowshop layout, where there exist several parallel machines in certain stages. Hybrid flowshop layout is widely used in real-life production environments since it increases the total manufacturing capacity by decreasing the impact of bottleneck stages. For hybrid flowshops, makespan is the most common and main performance criterion to increase the utilization of resources and obtain high throughput. However, minimizing energy consumption is also an important issue for manufacturing companies due to a series of environmental effects and increasing energy costs. Hence, in this thesis, the energy-efficient hybrid flowshop scheduling problem was addressed to minimize the makespan and the total energy consumption.

In this thesis, energy efficiency was studied from an operational planning perspective for the hybrid flowshops by employing a speed-scaling strategy. In many real-life production environments, machines can operate at multiple speed levels. Hence, the speed-scaling strategy operates the existing machinery regarding their energy consumption by simply adjusting the machine speeds for job operations. Since there is no need for installing additional expensive energy-efficient machinery, the speed scaling strategy is applicable to many real-life production environments, including small and medium-sized enterprises. Incorporating energy efficiency into the hybrid flowshop scheduling in such a way can directly help to decrease the total energy consumption during the manufacturing process and reduce the resulting environmental effects.

Consequently, the motivation of this thesis is to develop effective optimization methods to address the trade-off between the two important performance criteria, i.e., the makespan and the total energy consumption, in the hybrid flowshop environments by employing a practical speed-scaling strategy. Lack of fundamental mathematical models and related solution methods for the problem are remarkable gaps in the current literature that needs to be filled. This thesis aims to fill this research gap by presenting new exact and heuristic solution methods for the EHFSP with the makespan and TEC criteria that employs speed scaling strategy.

In this thesis, two practical variants of the speed scaling strategy were considered. In the first variant of the EHFSP (EHFSP-V1), speed-scaling is taken to be job-based due to its simplicity and tractability; that is, the same speed level is employed for a certain job through all stages. On the other hand, in the second variant of the EHFSP (EHFSP-V2), the job-based speed scaling strategy assumption is relaxed, and it is assumed that the speed of a job can vary from stages to stages. Since the speed levels create a contradiction between the makespan and energy consumption criteria, bi-objective exact and heuristic solution methods were proposed for these two variants of the problem in this thesis. Namely, a new bi-objective MILP model, a new bi-objective CP model and seven new bi-objective metaheuristics were proposed for the EHFSP-V1 that employs a job-based speed scaling strategy. Then, these solution methods were extended with several modifications for the EHFSP-V2 that employs a matrix-based speed scaling strategy. Subsequently, a new bi-objective MILP model, a new bi-objective CP model and four new bi-objective metaheuristics were proposed for the EHFSP-V2.

In this thesis, new benchmark instances were also developed for the EHFSP by modifying the well-known HFSP benchmark set from the literature (Carlier and Neron, 2000; Liao et al., 2012; Öztop et al., 2019). Then, the performance evaluation of the proposed solution methods was made using this extensive set of benchmarks in terms of cardinality, diversity and closeness of the generated solutions.

As mentioned above, this thesis introduced new bi-objective MILP and CP models for both EHFSP-V1 and EHFSP-V2 to provide fundamental mathematical models for the studied problem. Then, the augmented ε-constraint method was used to solve the proposed bi-objective MILP and CP models, as an exact solution methodology. Namely, for small instances, the proposed MILP and CP models were solved through the augmented ε-constraint method without a time limit to obtain the Pareto-optimal solutions. Since the problem is NP-hard and the solution time grows exponentially, the sets of non-dominated solutions were obtained with augmented ε-constraint method under a time limit for larger instances. The computational results in Chapter 7 demonstrated that the exact Pareto-optimal solution sets could be obtained for small instances in reasonable computational times, by employing the proposed bi-objective MILP and CP models, especially for the EHFSP-V1.

To the best of our knowledge, this thesis presented a constraint programming approach to the EHFSP for the first time in the literature. In the literature, CP has been successfully applied to various single-objective scheduling problems, and it has become an important competitor to the state-of-the-art MILP technique. This thesis further applied the CP technique to the HFSP in a multi-objective optimization framework. The results in Chapter 7 showed that the CP performs much better than the MILP for both EHFSP-V1 and EHFSP-V2 in terms of the solution quality. Hence, this thesis also revealed the effectiveness of CP for solving such a complex bi-objective hybrid flowshop scheduling problem.

As mentioned above, due to the NP-hard nature of the problem, seven bi-objective metaheuristics were proposed for the EHFSP-V1, which are two variants of the IG algorithm (E_IG, $\mathrm{E\_IG_{ALL}}$), a VBIH algorithm (E_VBIH) and four variants of the ensemble of metaheuristic algorithms ($\mathrm{E\_EM}, \mathrm{E\_EM_{HFR}}, \mathrm{E\_EM_{HFN}}, \mathrm{E\_EM_{HFRN}}$). Similarly, four bi-objective metaheuristics were also proposed for the EHFSP-V2, which are two variants of the IG algorithm (E_IG2, $\mathrm{E\_IG2_{ALL}}$), a VBIH algorithm (E_VBIH2), and an ensemble of metaheuristic algorithms (E_EM2).

Furthermore, in this thesis, a new constructive heuristic NEH_M(x) was presented for the HFSP with the makespan criterion by modifying the well-known NEH heuristic. As mentioned in Chapter 6, all proposed bi-objective metaheuristics in this thesis employ NEH_M(x) to generate the initial solution. Consequently, the results in Section 7.2 showed that the proposed NEH_M(x) heuristic significantly outperforms the well-known NEH heuristic for the HFSP with the makespan criterion, since the average RPD of the NEH_M(x) is 2.42% from the best-known solutions reported in Öztop et al. (2019). Additionally, the results showed that employing NEH_M(x), as a constructive heuristic instead of NEH, significantly improves the performance of the single-objective (IG, $\mathrm{IG_{ALL}}$ and VBIH) algorithms with the makespan criterion.

In this thesis, two new heuristic fitness calculation approaches were also proposed to compensate for the inefficiency of the standard forward scheduling approach for fitness function calculation in HFSP. Then, these heuristic fitness calculation approaches, i.e., HFR and HFN, were employed in the proposed bi-objective metaheuristics ($E\_EM_{HFR}, E\_EM_{HFN}, E\_EM_{HFRN}$, E_IG2, E_IG2_{ALL}, E_VBIH2 and E_EM2). The computational results in Chapter 7 revealed that the HFR and HFN approaches substantially improve the solution quality of the proposed metaheuristic algorithms.

Extensive computational experiments were also conducted for both EHFSP-V1 and EHFSP-V2. Initially, the performance of the proposed bi-objective metaheuristics was assessed on small instances using the Pareto-optimal solution sets. Then, the performances of the proposed bi-objective metaheuristics were compared with each other as well as the time-limited MILP and CP solutions for larger instances. Comprehensive statistical analyses were also performed to verify the computational results statistically.

The computational results of the EHFSP-V1 showed that all proposed metaheuristic algorithms are able to find more than 67% of the Pareto-optimal solutions on the average over 47 instances. Especially, E_EM_{HFRN} and E_EM_{HFR} algorithms are able to find approximately 86% of the Pareto-optimal solutions, which indicates an outstanding performance for small instances. The results on medium instances also showed that all of the metaheuristic algorithms outperform the time-limited MILP and CP in terms of quality and cardinality of the Pareto frontier. It can also be said that ensembles of metaheuristic algorithms with the HFR/HFN approach perform superbly for all instances of the EHFSP-V1 in terms of high ratio of reference solutions found and closeness to the reference set, as compared the other solution approaches.

The computational results of the EHFSP-V2 showed that all proposed metaheuristic algorithms are able to find more than 43% of the Pareto-optimal solutions on the average over 12 instances. The results on medium instances also showed that all of the proposed metaheuristic algorithms outperform the time-limited MILP and CP in terms of quality and cardinality of the Pareto frontier. Furthermore, it can be said that the E_IG2 algorithm performs superbly for all instances of the EHFSP-V2 in terms of high ratio of reference solutions found and closeness to the reference set, as compared to the other solution methods.

Consequently, by using the proposed bi-objective metaheuristics in this thesis, good quality solutions can be obtained for both EHFSP-V1 and EHFSP-V2 in very reasonable computational times. The results demonstrated that the proposed bi-objective metaheuristics are compatible with the exact solution methods for small instances, and significantly outperforms them for larger instances.

In conclusion, this thesis contributes to the literature on energy-efficient scheduling and hybrid flowshop scheduling by:

(1) applying the job-based (EHFSP-V1) and matrix-based (EHFSP-V2) speed scaling strategy to the HFSP,

(2) applying constraint programming to the HFSP,

(3) presenting new bi-objective MILP and CP models for the EHFSP,

(4) applying the augmented ε-constraint method to solve proposed bi-objective MILP and CP models,

(5) presenting a new benchmark set for the EHFSP,

(6) presenting a new constructive heuristic for the HFSP,

(7) proposing two new heuristic fitness calculation approaches for the HFSP,

(8) developing seven original effective bi-objective metaheuristic algorithms for the EHFSP-V1,

(9) developing four original effective bi-objective metaheuristic algorithms for the EHFSP-V2.

From a practical viewpoint, the managers can make decisions considering both production and energy efficiency by using the developed solution methods in this thesis. The proposed methods do not require a significant financial investment from a managerial perspective. Since there is no need for installing costly energy-efficient machinery, they can also be employed by small and medium-sized enterprises. Consequently, proposed energy-efficient scheduling methods can provide economic savings from energy resource consumptions in addition to the environmental benefits, without making a significant financial investment.

Furthermore, the generated Pareto frontiers, by employing the proposed solution methods, can serve as visual tools for managers to make informed decisions considering both energy and production efficiency in hybrid flowshops. Utilizing an extensive set of compromises produced by the Pareto frontiers, managers can make comprehensive trade-off analyses and explore all the outcomes of a decision regarding both performance criteria.

This thesis can serve as a reference for the researchers and experts working on the energy-efficient HFSP with speed scaling strategy, which is scarcely studied in the current literature. In further studies, different bi-objective metaheuristics and/or matheuristics can be developed for the EHFSP. Especially, developing matheuristics, i.e., the mathematical model-based heuristics, is a very promising research direction, since it utilizes the advantages of both mathematical modeling and heuristic techniques. In future research, an effective matheuristic can be developed for the problem by combining the proposed bi-objective models and metaheuristics in this thesis.

Furthermore, valid inequalities can be developed for the problem in order to strengthen the proposed bi-objective MILP and CP models. Employing effective valid inequalities can decrease the solution time of the proposed models by narrowing the search space of the problem. Lower bounds can also be developed for the studied problem. Especially, an effective lower bound can accelerate the optimality proof process of the CP technique.

Studying different objectives such as total tardiness or total flow time with an energy-related objective in a hybrid flowshop environment is another research direction to follow. As pointed out by Öztop et al. (2019), the total flow time criterion is very important to reduce the total in-process inventory and time, since it affects the total capacity utilization. The tardiness criterion is also very critical to improve the customer service level. Hence, in future research, these two important production efficiency related objectives can be incorporated with the total energy consumption criterion for the hybrid flowshops. The proposed models and metaheuristics in this thesis can be easily extended for these two versions of the EHFSP.

Additionally, machine-based speed scaling can be studied for the EHFSP as another realistic extension. This variant can be very practical, especially for the hybrid flowshops, where there exists setup time for the machines during the speed adjustments. For such hybrid flowshops, a single speed level can be defined for each machine, instead of determining the speed levels of the machines for each job operation.

Furthermore, considering the existing literature, energy-efficient scheduling with sequence-dependent setup times has attracted relatively lower attention, most likely because of the complexity of the problems. However, recent studies on hybrid flowshop scheduling have focused on more realistic problems such as problems with sequence-dependent setup times, machine eligibilities, and job precedence constraints (Ribas et al., 2010). Therefore, based on the fundamental energy-efficient hybrid flowshop scheduling models and solution methods presented in this thesis, more complex models, such as EHFSP models with sequence-dependent setup times and/or machine eligibility constraints, can also be developed as well as the related solution methods.

REFERENCES

Behnamian, J., & Fatemi Ghomi, S. M. T. (2011). Hybrid flowshop scheduling with machine and resource-dependent processing times. *Applied Mathematical Modelling*, *35*(3),1107–1123.

Bruzzone, A. A. G., Anghinolfi, D., Paolucci, M., & Tonelli, F. (2012). Energy-aware scheduling for improving manufacturing process sustainability: a mathematical model for flexible flowshops. *CIRP Annals-Manufacturing Technology*, *61*(1), 459–62.

Carlier, J., & Neron, E. (2000). An exact method for solving the multiprocessor flowshop. *R.A.I.R.O. Operations Research, 34*, 1–25.

Che, A., Lv, K., Levner, E., & Kats, V. (2015). Energy consumption minimization for single machine scheduling with bounded maximum tardiness. *Proceedings of 2015 IEEE 12th International Conference on Networking, Sensing and Control,* 146-150.

Che, A., Wu, X., Peng, J., & Yan, P. (2017) Energy-efficient bi-objective single-machine scheduling with power-down mechanism. *Computers & Operations Research, 85*, 172-183.

Chen, T. L., Cheng, C. Y., & Chou, Y. H. (2018). Multi-objective genetic algorithm for energy-efficient hybrid flow shop scheduling with lot streaming. *Annals of Operations Research,* 1-24.

Coello, C. A. C., Lamont, G. B., & Van Veldhuizen, D. A. (2007). *Evolutionary algorithms for solving multi-objective problems* (Vol. 5). New York: Springer.

Dai, M., Tang, D., Giret, A., Salido, M. A., & Li, W. (2013). Energy-efficient scheduling for a flexible flowshop using an improved genetic-simulated annealing algorithm. *Robotics and Computer Integrated Manufacturing, 29,* 418–429.

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. A. M. T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, *6*(2), 182–197.

Ding, J-Y., Song, S., Zhang, R., Chiong, R., & Wu, C. (2016a). Parallel machine scheduling under time-of-use electricity prices: new models and optimization approaches. *IEEE Transactions on Automation Science and Engineering*, *13*, 1138-1154.

Ding, J. Y., Song, S., & Wu, C. (2016b). Carbon-efficient scheduling of flowshops by multi-objective optimization. *European Journal of Operational Research*, *248*(3), 758-771.

Deng, J., Wang, L., Wu, C., Wang, J., & Zheng, X. (2016). A competitive memetic algorithm for carbon-efficient scheduling of distributed flow-shop. In *International Conference on Intelligent Computing* (pp. 476-488). Springer, Cham.

Dubois-Lacoste, J., Pagnozzi, F., & Stützle, T. (2017). An iterated greedy algorithm with optimization of partial solutions for the makespan permutation flowshop problem. *Computers & Operations Research*, *81*, 160–166.

Fang, K-T., & Lin, B. M. T. (2013). Parallel-machine scheduling to minimize tardiness penalty and power cost. *Computers & Industrial Engineering*, *64*(1), 224–234.

Fang, K., Uhan, N., Zhao, F., & Sutherland, J. W. (2011). A new approach to scheduling in manufacturing for power consumption and carbon footprint reduction. *Journal of Manufacturing Systems*, *30*(4), 234–240.

Fang, K., Uhan, N. A., Zhao, F., & Sutherland, J. W. (2013). Flowshop scheduling with peak power consumption constraints. *Annals of Operations Research*, *206*, 115–145.

Gahm, C., Denz, F., Dirr, M., & Tuma, A. (2016). Energy-efficient scheduling in manufacturing companies: a review and research framework. *European Journal of Operational Research, 248*, 744-757.

Grabowski, J., & Pempera, J. (2000). Sequencing of jobs in some production system. *European Journal of Operational Research, 125*(3), 535–550.

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

Gupta, J. N. D. (1988). Two-stage, hybrid flowshop scheduling problem. *Journal of the Operational Research Society, 39*(4), 359–364.

IBM ILOG CPLEX, 2017. IBM ILOG CPLEX Optimization Studio V12.8, Language User's Manual.

Jiang, E. D., & Wang, L. (2019). An improved multi-objective evolutionary algorithm based on decomposition for energy-efficient permutation flow shop scheduling problem with sequence-dependent setup time. *International Journal of Production Research, 57*(6), 1756-1771.

Jiang, E., Wang, L., & Lu, J. (2017). Modified multi-objective evolutionary algorithm based on decomposition for low-carbon scheduling of distributed permutation flow-shop. In *2017 IEEE Symposium Series on Computational Intelligence (SSCI)* (pp. 1-7). IEEE.

Jin, Z. H., Ohno, K., Ito, T., & Elmaghraby, S. E. (2002). Scheduling hybrid flowshops in printed circuit board assembly lines. *Production and Operations Management*, *11*(2), 216–230.

Jouglet, A., Oğuz, C., & Sevaux, M. (2009). Hybrid flow-shop: a memetic algorithm using constraint-based scheduling for efficient search. *Journal of Mathematical Modelling and Algorithms*, *8*(3), 271-292.

Jungwattanakit, J., Reodecha, M., Chaovalitwongse, P., & Werner, F. (2008). Algorithms for flexible flowshop problems with unrelated parallel machines, setup times, and dual criteria. *International Journal of Advanced Manufacturing Technology*, *37*(3–4), 354–370.

Jungwattanakit, J., Reodecha, M., Chaovalitwongse, P., & Werner, F. (2009). A comparison of scheduling algorithms for flexible flowshop problems with unrelated parallel machines, setup times, and dual criteria. *Computers & Operations Research*, *36*(2), 358–378.

Lei, D., Gao, L., & Zheng, Y. (2018). A novel teaching-learning-based optimization algorithm for energy-efficient scheduling in hybrid flow shop. *IEEE Transactions on Engineering Management*, *65*(2), 330-340.

Li, M., Lei, D., & Cai, J. (2019). Two-level imperialist competitive algorithm for energy-efficient hybrid flow shop scheduling problem with relative importance of objectives. *Swarm and Evolutionary Computation*, *49*, 34-43.

Li, J., Sang, H., Han, Y., Wang, C., & Gao, K. (2018). Efficient multi-objective optimization algorithm for hybrid flow shop scheduling problems with setup energy consumptions. *Journal of Cleaner Production*, *181*, 584-598.

Liao, C. J., Tjandradjaj, E., & Chung, T. P. (2012). An approach using particle swarm optimization and bottleneck heuristic to solve flowshop scheduling problem. *Applied Soft Computing*, *12*, 1755–1764.

Liu, C. Y., & Chang, S. C. (2000). Scheduling flexible flowshops with sequence dependent setup effects. *IEEE Transactions on Robotics and Automation*, *16*(4), 408–419.

Liu, C. H., & Huang, D. H. (2014). Reduction of power consumption and carbon footprints by applying multi-objective optimisation via genetic algorithms. *International Journal of Production Research, 52*, 337–352.

Liu, X., Zou, F., & Zhang, X. (2008). Mathematical model and genetic optimization for hybrid flow shop scheduling problem based on energy consumption. In *2008 Chinese Control and Decision Conference* (pp. 1002-1007). IEEE.

Lu, C., Gao, L., Li, X., Pan, Q. K., & Wang, Q. (2017). Energy-efficient permutation flow shop scheduling problem using a hybrid multi-objective backtracking search algorithm. *Journal of Cleaner Production*, *144*, 228-238.

Luo, H., Du, B., Huang, G. Q., Chen, H., & Li, X. (2013). Hybrid flowshop scheduling considering machine electricity consumption cost. *International Journal of Production Economics*, *146*, 423–439.

Mallipeddi, R., & Suganthan, P. N. (2010). Ensemble of Constraint Handling Techniques. *IEEE Transactions on Evolutionary Computation*, *14*(4), 561-579.

Mallipeddi, R., Suganthan, P. N., Pan, Q. K., & Tasgetiren, M. F. (2011). Differential evolution algorithm with ensemble of parameters and mutation strategies. *Applied Soft Computing*, *11*(2), 1679-1696.

Mansouri, S. A., & Aktas, E. (2016). Minimizing energy consumption and makespan in a two-machine flowshop scheduling problem. *Journal of the Operational Research Society, 67*, 1382-1394.

Mansouri, S. A., Aktas, E., & Besikci, U. (2016). Green scheduling of a two-machine flowshop: trade-off between makespan and energy consumption. *European Journal of Operational Research, 248*, 772-788.

Mavrotas, G. (2009). Effective implementation of the e-constraint method in multi-objective mathematical programming problems, *Applied Mathematics and Computation, 213*, 455–465.

Meng, L., Zhang, C., Shao, X., Ren, Y., & Ren, C. (2019). Mathematical modelling and optimisation of energy-conscious hybrid flow shop scheduling problem with unrelated parallel machines. *International Journal of Production Research*, *57*(4), 1119–1145.

Montgomery, D. C. (2008). *Design and Analysis of Experiments*. John Wiley & Sons.

Moon, J-Y., Shin, K., & Park, J. (2013). Optimization of production scheduling with time dependent and machine-dependent electricity cost for industrial energy efficiency. *International Journal of Advanced Manufacturing Technology*, *68*(1-4), 523-535.

Mouzon, G., Yildirim, M. B., & Twomey, J. (2007). Operational methods for the minimization of energy consumption of manufacturing equipment. *International Journal of Production Research*, *45*(18-19), 4247–4271.

Mouzon, G., & Yildirim, M. B. (2008). A framework to minimize total energy consumption and total tardiness on a single machine. *International Journal of Sustainable Engineering*, *1*(2), 105–116.

Nawaz, M., Enscore, Jr. E. E., & Ham, I. (1983). A heuristic algorithm for the *m*-machine *n*-job flowshop sequencing problem. *OMEGA*, *11*(1), 91–95.

Okabe, T., Jin, Y., & Sendhoff, B. (2003). A critical survey of performance indices for multi-objective optimisation. In *The 2003 Congress on Evolutionary Computation, 2003. CEC'03.* (Vol. 2, pp. 878-885). IEEE.

Osman, I., & Potts, C. (1989). Simulated annealing for permutation flow-shop scheduling. *OMEGA*, *17*(6), 551–557.

Öztop, H., Tasgetiren, M. F., Eliiyi, D. T., & Pan, Q. K. (2018). Green Permutation Flowshop Scheduling: A Trade-off-Between Energy Consumption and Total Flow Time. In *International Conference on Intelligent Computing* (pp. 753-759). Springer, Cham.

Öztop, H., Tasgetiren, M. F., Eliiyi, D. T., & Pan, Q. K. (2019). Metaheuristic algorithms for the hybrid flowshop scheduling problem. *Computers & Operations Research*, *111*, 177-196.

Öztop, H., Tasgetiren, M. F., Eliiyi, D. T., Pan, Q. K., & Kandiller, L. (2020). An energy-efficient permutation flowshop scheduling problem. *Expert Systems with Applications*, *150*, 113279.

Pan, Q., Wang, L., Li, J. Q., & Duan, J. H. (2014). A novel discrete artificial bee colony algorithm for the hybrid flowshop scheduling problem with makespan minimisation. *Omega*, *45*, 42–56.

Pan, Q., Wang, L., Mao, K., Zhao, J. H., & Zhang, M. (2013). An effective artificial bee colony algorithm for a real-world hybrid flowshop problem in steel making process. *IEEE Transactions on Automation Science and Engineering*, *10*(2), 307–322.

Pan, Q. K., Wang, L., & Qian, B. (2009). A novel differential evolution algorithm for bi-criteria no-wait flow shop scheduling problems. *Computers & Operations Research*, *36*(8), 2498-2511.

Pinedo, M. (2002). *Scheduling: Theory Algorithms and Systems* (Vol. 3). New York: Springer.

Ribas, I., Leisten, R., & Framinan, J. M. (2010). Review and classification of hybrid flowshop scheduling problems from a production system and a solutions procedure perspective. *Computers & Operations Research*, *37*, 1439–1454.

Ruiz, R., & Stützle, T. (2007). A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. *European Journal of Operational Research*, *177*(3), 2033–2049.

Ruiz, R., & Vazquez Rodriguez, J. A. (2010). The hybrid flowshop scheduling problem. *European Journal of Operational Research*, *205*, 1–18.

Salido, M. A., Escamilla, J., Giret, A., & Barber, F. (2016). A genetic algorithm for energy-efficiency in job-shop scheduling. *The International Journal of Advanced Manufacturing Technology*, *85*(5-8), 1303-1314.

Shen, J., Wang, L., & Wang, J. (2017). A discrete teaching-learning-based optimisation algorithm for hybrid flowshop scheduling problem with peak power consumption constraints. In *2017 IEEE Symposium Series on Computational Intelligence (SSCI)* (pp. 1-7). IEEE.

Sherali, H. D., Sarin, S. C., & Kodialam, M. S. (1990). Models and algorithms for a two-stage production process. *Production Planning and Control*, *1*(1), 27–39.

Shrouf, F., Ordieres-Mere, J., García-Sanchez, A., & Ortega-Mier, M. (2014). Optimizing the production scheduling of a single machine to minimize total energy consumption costs. *Journal of Cleaner Production*, *67*, 197-207.

Tan, K. C., Goh, C. K., Yang, Y., & Lee, T. H. (2006). Evolving better population distribution and exploration in evolutionary multi-objective optimization. *European Journal of Operational Research*, *171*, 463–495.

Tang, D., Dai, M., Salido, M. A., & Giret, A. (2016). Energy-efficient dynamic scheduling for a flexible flow shop using an improved particle swarm optimization. *Computers in Industry, 81*, 82-95.

Tasgetiren, M. F., Eliiyi, U., Öztop, H., Kizilay, D., & Pan, Q. K. (2018a). An energy-efficient single machine scheduling with release dates and sequence-dependent setup times. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion* (pp. 145-146).

Tasgetiren, M. F., Öztop, H., Eliiyi, U., Eliiyi, D. T., & Pan, Q. K. (2018b). Energy-Efficient Single Machine Total Weighted Tardiness Problem with Sequence-Dependent Setup Times. In *International Conference on Intelligent Computing* (pp. 746-758). Springer, Cham.

Tasgetiren, M. F., Öztop, H., Gao, L., Pan, Q. K., & Li, X. (2019). A variable iterated local search algorithm for energy-efficient no-idle flowshop scheduling problem. *Procedia Manufacturing, 39*, 1185-1193.

Tasgetiren, M. F., Pan, Q., Kizilay, D., & Gao, K. (2016). A variable block insertion heuristic for the blocking flowshop scheduling problem with total flowtime criterion. *Algorithms*, *9*(4), 71.

Tasgetiren, M. F., Pan, Q. K., Kizilay, D., & Vélez-Gallego, M. C. (2017). A variable block insertion heuristic for permutation flowshops with makespan criterion. In *2017 IEEE Congress on Evolutionary Computation (CEC)* (pp. 726-733). IEEE.

Tasgetiren, M. F., Suganthan, P. N., & Pan, Q.K. (2010). An ensemble of discrete differential evolution algorithms for solving the generalized traveling salesman problem. *Applied Mathematics and Computation*, *215*, 3356–3368.

Vignier, A., Billaut, J. C., & Proust, C. (1999). Hybrid flowshop scheduling problems: state of the art. *Rairo - Recherche Operationnelle – Operations Research, 33*(2), 117–183.

Wang, J. J., & Wang, L. (2018). A knowledge-based cooperative algorithm for energy-efficient scheduling of distributed flow-shop. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *99*, 1-15.

Wang, J., Wang, L., Wu, C., & Shen, J. (2017). A Cooperative Algorithm for Energy-efficient Scheduling of Distributed No-wait Flowshop. In *2017 IEEE Symposium Series on Computational Intelligence (SSCI)* (pp. 1-8). IEEE.

Wang, S., Wang, X., Yu, J., Ma, S., & Liu, M. (2018). Bi-objective identical parallel machine scheduling to minimize total energy consumption and makespan. *Journal of Cleaner Production*, *193*, 424-440.

Wittrock, J. R. (1988). An adaptable scheduling algorithm for flexible flow lines. *Operations Research*, *36*(3), 445–453.

Wu, X., & Che, A. (2019). A memetic differential evolution algorithm for energy-efficient parallel machine scheduling. *Omega*, *82*, 155-165.

Wu, X., Shen, X., & Cui, Q. (2018). Multi-objective flexible flow shop scheduling problem considering variable processing time due to renewable energy. *Sustainability*, *10*(3), 841.

Xu, H., Lu, Z., & Cheng, T. (2014). Iterated local search for single-machine scheduling with sequence-dependent setup times to minimize total weighted tardiness. *Journal of Scheduling*, *17*(3), 271-287.

Yan, J., Li, L., Zhao, F., Zhang, F., & Zhao, Q. (2016). A multi-level optimization approach for energy-efficient flexible flow shop scheduling. *Journal of Cleaner Production*, *137*, 1543-1552.

Yin, L., Li, X., Gao, L., Lu, C., & Zhang, Z. (2017). A novel mathematical model and multi-objective method for the low-carbon flexible job shop scheduling problem. *Sustainable Computing: Informatics and Systems,13*, 15-30.

Zeng, Z., Hong, M., Man, Y., Li, J., Zhang, Y., & Liu, H. (2018). Multi-objective optimization of flexible flow shop scheduling with batch process—Consideration total electricity consumption and material wastage. *Journal of Cleaner Production*, 183, 925-939.

Zhang, R., & Chiong, R. (2016). 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*, *112*, 3361-3375.

Zhang, H., Fang, Y., Pan, R., & Ge, C. (2018). A new greedy insertion heuristic algorithm with a multi-stage filtering mechanism for energy-efficient single machine scheduling problems. *Algorithms*, 11: 18.

Zhang, Q., & Li, H. (2007). MOEA/D: a multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, *11*(6), 712–731.

Zhang, B., Pan, Q. K., Gao, L., Meng, L. L., Li, X. Y., & Peng, K. K. (2019a). A Three-Stage Multiobjective Approach Based on Decomposition for an Energy-Efficient Hybrid Flow Shop Scheduling Problem. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*.

Zhang, B., Pan, Q. K., Gao, L., Li, X. Y., Meng, L. L., & Peng, K. K. (2019b). A multi-objective evolutionary algorithm based on decomposition for hybrid flowshop green scheduling problem. *Computers & Industrial Engineering*, *136*, 325-344.

Zhang, H., Zhao, F., Fang, K., & Sutherland, J. (2014). Energy-conscious flowshop scheduling under time-of-use electricity tariffs. *CIRP Annals - Manufacturing Technology*, *63*, 37-40.

Zheng, X. L., & Wang, L. (2018). A collaborative multi-objective fruit fly optimization algorithm for the resource-constrained unrelated parallel machine green scheduling problem. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *48*(5), 790-800.

Zhou, A., Qu, B. Y., Li, H., Zhao, S. Z., Suganthan, P. N., & Zhang, Q. (2011). Multiobjective evolutionary algorithms: a survey of the state of the art. *Swarm and Evolutionary Computation*, *1*, 32-49.