



YAŞAR UNIVERSITY

GRADUATE SCHOOL

PHD THESIS

**NEW SOLUTION TECHNIQUES
FOR NO-WAIT PERMUTATION FLOWSHOP
SCHEDULING PROBLEMS**

DAMLA YÜKSEL

THESIS ADVISOR: PROF. (PHD) LEVENT KANDİLLER

PHD IN INDUSTRIAL ENGINEERING

BORNOVA / İZMİR
AUGUST 2024

JURY APPROVAL PAGE

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. (PhD) Levent Kandiller (Advisor)
Yaşar University

.....

Prof. (PhD) Mustafa Arslan Örnek
Yaşar University

.....

Prof. (PhD) Özgür Özpeynirci
Izmir University of Economics

.....

Prof. (PhD) Mehmet Cemali Dinçer
Yaşar University

.....

Assoc. Prof. (PhD) Engin Topan
University of Twente

.....

Prof. (PhD) Yücel Öztürkoğlu
Director of the Graduate School



ABSTRACT

NEW SOLUTION TECHNIQUES FOR NO-WAIT PERMUTATION FLOWSHOP SCHEDULING PROBLEMS

Yüksel, Damla

PhD, Industrial Engineering

Advisor: Prof. (PhD) Levent Kandiller

August 2024

No-Wait Permutation Flowshop Scheduling Problem (NWPFS) is a scheduling problem variant where jobs must proceed through machines in a fixed order without waiting times between operations. This thesis explores innovative solution techniques for the NWPFSs. The primary contributions of this thesis are twofold: *single-objective optimization* and *bi-criteria optimization*. For *single-objective optimization*, this thesis examines five mathematical model formulations — three using Mixed-Integer Linear Programming (MILP) and two using Constraint Programming (CP) — focused on separately minimizing makespan, total flow time, and total tardiness. One MILP model is enhanced with valid inequalities to address these objectives. A new Lower Bound (LB) mechanism based on the Shortest Path (SP) algorithm is developed to optimize makespan, total flow time, total tardiness, and the number of tardy jobs separately. Following that, two mathematical models, one belonging to the MILP class and the other to the CP class, have been studied for the number of tardy job minimization in NWPFSs. A novel upper bound, the Sacrifice and Rearrange Heuristic (SRH), is introduced to minimize the number of tardy jobs. Optimizing the number of tardy jobs in NWPFSs requires high-quality due dates, as they are crucial for improving performance metrics related to lateness. A new mechanism for generating high-quality due dates has been developed to address this. Incorporating the Sacrifice and Rearrange Heuristic (SRH), this mechanism ensures practical and effective due dates. For *bi-criteria optimization*, the NWPFS is approached as a combinatorial optimization problem with two objectives, aiming to minimize total flow time and makespan simultaneously: Bi-Criteria No-Wait Permutation Flowshop Scheduling Problems (BI-CRI NWPFSs). Initially, an MILP model formulation is



explored to address BI-CRI NWPFPSPs. Following this, Q-learning-guided algorithms are developed for Bi-CRI NWPFPSPs. Q-learning, a well-known reinforcement learning technique, is employed to direct action selection, thereby reducing the need for random exploration during the iterative metaheuristic process. The developed Q-learning guided metaheuristics are Bi-Criteria Iterated Greedy Algorithm with Q-Learning (BC-IG_{QL}) and Bi-Criteria Block Insertion Heuristic Algorithm with Q-Learning (BC-BIH_{QL}). The performance of these algorithms is compared with other state-of-the-art approaches. Thus, this thesis advances the literature on the NWPFPSPs by developing new solution techniques for both *single-objective* and *bi-criteria* scenarios.

Keywords: no-wait permutation flowshop scheduling problem, bi-criteria optimization, mixed-integer linear programming, constraint programming, heuristics, Q-learning, due dates, makespan, total flow time, total tardiness, number of tardy jobs



ÖZ

BEKLEMESİZ PERMÜTASYON AKIŞ TİPİ ÇİZELGELEME PROBLEMLERİ İÇİN YENİ ÇÖZÜM TEKNİKLERİ

Yüksel, Damla

Doktora Tezi, Endüstri Mühendisliği

Danışman: Prof. Dr. Levent Kandiller

August 2024

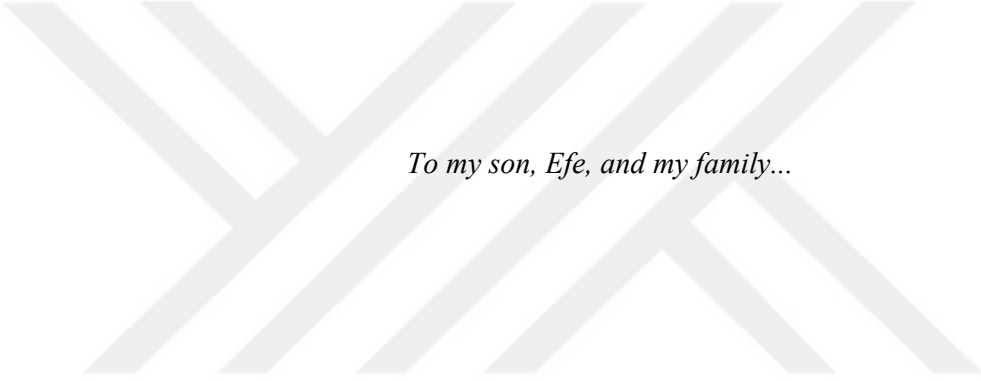
Beklemesiz Permütasyon Akış Tipi Çizelgeleme Problemi (NWPFSF) çizelgeleme problemlerinin bir varyantı olup bu problem tipinde işler, makineler arasında bekleme süreleri olmadan makinelerden sabit bir sırayla geçmelidir. Bu tez, Beklemesiz Permütasyon Akış Tipi Çizelgeleme Problemleri için yenilikçi çözüm tekniklerini araştırmaktadır. Bu tezin ana katkıları iki yönlüdür: **tek-hedefli** optimizasyon ve **iki-kriterli** optimizasyon. **Tek-hedefli** optimizasyon kapsamında, tamamlanma süresi, toplam akış süresi ve toplam gecikmeyi ayrı ayrı minimize etmeye yönelik beş matematiksel model formülasyonunu incelemektedir — üçü Karma-Tamsayılı Doğrusal Programlama (MILP) ve ikisi Kısıtlama Programlama (CP) kullanılarak geliştirilmiştir. Karma-Tamsayılı Doğrusal Programlama modellerden bir tanesi, bu hedeflere ulaşmak için geçerli eşitsizliklerle güçlendirilmiştir. Ayrıca, tamamlanma süresi, toplam akış süresi, toplam gecikme ve geciken işlerin sayısını ayrı ayrı optimize etmek için En Kısa Yol (SP) algoritmasına dayanan yeni bir Alt Sınır (LB) mekanizması geliştirilmiştir. Daha sonra, Beklemesiz Permütasyon Akış Tipi Çizelgeleme Problemlerinde geciken işlerin sayısını minimize etmek amacıyla bir Karma-Tamsayılı Doğrusal Programlama (MILP) sınıfına ve bir Kısıtlama Programlama (CP) sınıfına ait iki matematiksel model incelenmiştir. Geciken işlerin sayısını enazlamak için yeni bir üst sınır olan Feda Et ve Yeniden Düzenle Sezgisel Algoritması (SRH) önerilmiştir. Beklemesiz Permütasyon Akış Tipi Çizelgeleme Problemlerinde geciken işlerin sayısını optimize etmek, geç kalma ile ilgili performans ölçütlerini iyileştirmek için de önemli olan yüksek kaliteli teslim tarihleri gerektirir. Bu sorunu çözmek amacıyla, yüksek kaliteli teslim tarihleri üretmek için yeni bir



mekanizma geliştirilmiştir. Bu mekanizma, Feda Et ve Yeniden Düzenle Sezgisel Algoritmasını (SRH) içeren, hem pratik hem de etkili teslim tarihleri üretir. **İki-kriterli** optimizasyon kapsamında ise, Beklemesiz Permütasyon Akış Tipi Çizelgeleme Problemleri, iki-kriterli bir kombinatoriyal optimizasyon problemi olarak ele alınır ve toplam akış süresi ile tamamlanma süresini aynı anda minimize etmeyi amaçlar: İki-Kriterli Beklemesiz Permütasyon Akış Tipi Çizelgeleme Problemleri (BI-CRI NWPFSs). İlk olarak, İki-Kriterli Beklemesiz Permütasyon Akış Tipi Çizelgeleme Problemlerini (BI-CRI NWPFSs) ele almak amacıyla bir Karma-Tamsayılı Doğrusal Programlama (MILP) model formülasyonu incelenmiştir. Bunun ardından, İki-Kriterli Beklemesiz Permütasyon Akış Tipi Çizelgeleme Problemleri (BI-CRI NWPFSs) için Q-öğrenme destekli algoritmalar geliştirilmiştir. Q-öğrenme, eylem seçimlerini yönlendirmek için kullanılan tanınmış bir pekiştirmeli öğrenme tekniğidir ve bu sayede yinelemeli meta-sezgisel süreçte rastgele keşif ihtiyacını azaltır. Geliştirilen Q-öğrenme destekli meta-sezgiseller, Q-öğrenme ile İki-Kriterli Tekrarlı Açgözlü Algoritma (BC-IG_{QL}) ve Q-öğrenme ile İki-Kriterli Blok Ekleme İyileştirici Algoritma (BC-BIH_{QL}) olarak adlandırılmıştır. Bu algoritmaların performansı, diğer modern sezgisel yaklaşımların performanslarıyla karşılaştırılmıştır. Böylece, bu tez, hem **tek-hedefli** hem de **iki-kriterli** senaryolar için yeni çözüm teknikleri geliştirerek Beklemesiz Permütasyon Akış Tipi Çizelgeleme Problemleri (NWPFS) literatürünü zenginleştirmektedir.

Anahtar Kelimeler: beklemesiz permütasyon akış tipi çizelgeleme problemleri, iki-kriterli optimizasyon, karma-tamsayılı doğrusal programlama, kısıt programlama, sezgisel yöntemler, Q-öğrenme, teslim tarihi, tamamlanma süresi, toplam akış süresi, toplam gecikme, geciken işlerin sayısı





To my son, Efe, and my family...



ACKNOWLEDGEMENTS

First and foremost, I would like to extend my deepest thanks to my thesis advisor, Prof. Dr. Levent KANDİLLER, for his vast knowledge, unwavering guidance, contributions to my development during my PhD, and patience with me over such a long period. His invaluable support played a crucial role in each step of this thesis. I am grateful for everything he taught me and feel incredibly fortunate to have had the opportunity to work with him. Secondly, I would like to extend my special thanks to the late Prof. Dr. Mehmet Fatih TAŞGETİREN for his significant guidance. He consistently shared his insights with me to enhance my research skills. Furthermore, I would like to express my gratitude to the members of my doctoral thesis monitoring committee, Prof. Dr. Mustafa Arslan ÖRNEK and Prof. Dr. Özgür ÖZPEYNİRCİ, as well as the members of my thesis defense jury, Prof. Dr. Mehmet Cemali DİNÇER and Assoc. Prof. Dr. Engin TOPAN for their insightful comments, suggestions, and contributions, which significantly enriched my work.

Next, I want to extend a special thanks to my beloved husband, Yiğit YÜKSEL, for his everlasting support, love, and encouragement. He continually offered valuable insights, supported my skill development, and broadened my perspective at every step of this journey. My son, Efe YÜKSEL, was the most significant source of strength and inspiration throughout my doctoral journey. He has made me stronger and more patient than I ever thought possible. I feel incredibly fortunate to be his mother and am deeply grateful for him in my life. I express my deepest gratitude to my beloved mother, Figen KESİKBURUN, my father, Ergun KESİKBURUN, and my sister, Deniz KESİKBURUN, for their unwavering support and courage in every aspect of my life. I couldn't have succeeded without my mother's endless support. Furthermore, I am grateful to the faculty, colleagues, and especially my dear office mates, whose support has been constant at the Department of Industrial Engineering at Yaşar University.

Finally, my Ph.D. thesis was founded by the TÜBİTAK Science Fellowships and Grant Programmes Directorate (BİDEB) and I want to thank them for their financial support through the 2211-A National Ph.D. Scholarship Program.

Damla YÜKSEL
İzmir, 2024



TEXT OF OATH

I declare and honestly confirm that my study, titled “NEW SOLUTION TECHNIQUES FOR NO-WAIT PERMUTATION FLOWSHOP SCHEDULING PROBLEMS” and presented as a PhD Thesis, has been written without applying 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.

Damla YÜKSEL

28 August 2024



TABLE OF CONTENTS

JURY APPROVAL PAGE.....	iii
ABSTRACT.....	v
ÖZ.....	ix
ACKNOWLEDGEMENTS.....	xv
TEXT OF OATH.....	xvii
TABLE OF CONTENTS.....	xix
LIST OF FIGURES.....	xxv
LIST OF TABLES.....	xxvii
SYMBOLS AND ABBREVIATIONS.....	xxxvii
1. CHAPTER: INTRODUCTION.....	1
2. CHAPTER: PROBLEM DEFINITION AND RESEARCH QUESTIONS.....	5
3. CHAPTER: SINGLE-OBJECTIVE NO-WAIT PERMUTATION FLOWSHOP SCHEDULING PROBLEMS.....	11
3.1. Literature Review of Mathematical Model Formulations in NWPFSs.....	11
3.2. Mathematical Model Formulations.....	13
3.3. Computational Analysis of Mathematical Model Formulations.....	21
3.4. Empowering the Mathematical Model Formulations: Valid Inequalities.....	39
3.5. Computational Analysis of Valid Inequalities.....	41
3.6. Lower Bounds.....	46
3.7. Computational Analysis of Lower Bounds.....	55
3.8. Number of Tardy Jobs Minimization.....	65
3.8.1. Literature Review of Lateness-Related Performance Measurements in Flowshops.....	67
3.8.2. Mathematical Model Formulations.....	76
3.8.3. A Novel Quick Upper Bound: Sacrifice & Rearrange Heuristic.....	78

3.8.4.	A New Due Date Generation Mechanism and Benchmarking	85
3.8.5.	Computational Analysis on Number of Tardy Jobs Minimization in NWPFSs	87
4.	CHAPTER: BI-CRITERIA NO-WAIT PERMUTATION FLOWSHOP SCHEDULING PROBLEM.....	105
4.1.	Bi-Criteria Problem Setting.....	106
4.2.	Literature Review of Bi-Criteria NWPFSs and Q-learning-based Algorithms for Scheduling Problems	107
4.2.1.	Bi-Criteria No-Wait Permutation Flowshop Scheduling Problems .	107
4.2.1.	Q-Learning-Based Algorithms for Scheduling Problems.....	115
4.3.	Mathematical Model	117
4.4.	Metaheuristic Algorithms	119
4.4.1.	Iterated Greedy Algorithm	122
4.4.2.	Block Insertion Heuristic Algorithm.....	122
4.4.3.	Q-Learning	123
4.4.4.	Q-Learning Guided Metaheuristic Algorithms	124
4.4.5.	Bi-Criteria Genetic Local Search Algorithms.....	134
4.5.	Computational Analysis of Mathematical Model and Algorithms	136
5.	CHAPTER: DISCUSSION, CONCLUSION AND FUTURE RESEARCH DIRECTIONS.....	157
6.	REFERENCES	163
	APPENDICES	179
	APPENDIX A. Generated Due Dates for VRF Instances for Lateness-Related Performance Measures of NWPFSs.....	179
	APPENDIX A1. Generated Due Dates for VRF Instances with 10 Jobs.....	179
	APPENDIX A2. Generated Due Dates for VRF Instances with 20 Jobs.....	180

APPENDIX A3. Generated Due Dates for VRF Instances with 30 Jobs.....	181
APPENDIX A3. (Cont'd.) Generated Due Dates for VRF Instances with 30 Jobs	182
APPENDIX A4. Generated Due Dates for VRF Instances with 40 Jobs.....	183
APPENDIX A4. (Cont'd.) Generated Due Dates for VRF Instances with 40 Jobs	184
APPENDIX A5. Generated Due Dates for VRF Instances with 50 Jobs.....	185
APPENDIX A5. (Cont'd.) Generated Due Dates for VRF Instances with 50 Jobs	186
APPENDIX A5. (Cont'd.) Generated Due Dates for VRF Instances with 50 Jobs	187
APPENDIX A5. (Cont'd.) Generated Due Dates for VRF Instances with 50 Jobs	188
APPENDIX A6. Generated Due Dates for VRF Instances with 60 Jobs.....	189
APPENDIX A6. (Cont'd.) Generated Due Dates for VRF Instances with 60 Jobs	190
APPENDIX A6. (Cont'd.) Generated Due Dates for VRF Instances with 60 Jobs	191
APPENDIX A6. (Cont'd.) Generated Due Dates for VRF Instances with 60 Jobs	192



LIST OF FIGURES

Figure 2.1. An example for NWPFSF with 4 jobs on 3 machines	6
Figure 3.1. ΔF_{ik} and ΔL_{ik} values of the example with 4 jobs on 3 machines	15
Figure 3.2. Comparison of Model 1, Model 2, Model 3, Model 4, and Model 5 for Makespan on Taillard Instances	23
Figure 3.3. Comparison of Model 1, Model 2, Model 3, Model 4, and Model 5 for Total Flow Time on Taillard Instances.....	25
Figure 3.4. Comparison of Model 1, Model 2, Model 3, Model 4, and Model 5 for Total Tardiness on Taillard Instances.....	28
Figure 3.5. Comparison of Model 1, Model 2, Model 3, Model 4, and Model 5 for Makespan on VRF Instances	31
Figure 3.6. Comparison of Model 1, Model 2, Model 3, Model 4, and Model 5 for Total Flow Time on VRF Instances.....	34
Figure 3.7. Comparison of Model 1, Model 2, Model 3, Model 4, and Model 5 for Total Tardiness on VRF Instances.....	37
Figure 3.8. Directed graph of an NWPFSF	47
Figure 3.9. The Gantt Charts of Iterations that are provided in Table 3.20	82
Figure 4.1. The Gantt Chart represents the example with 5 jobs on 5 machines in an NWPFSF	106
Figure 4.2. The dominated solution set and the Pareto frontier of the example with jobs five and five machines in an NWPFSF	107
Figure 4.3. The overall structure of the Bi-Criteria IG algorithm with Q-learning	126
Figure 4.4. The overall structure of the Bi-Criteria BIH algorithm with Q-learning	131
Figure 4.5. Non-Dominated Solution Set of Instance 50-05-01 on the Small VRF Instances	141
Figure 4.6. Non-Dominated Solution Set of Instance 400-60-02 on the Large VRF Instances	141



LIST OF TABLES

Table 3.1. Sets, Indices, and Parameters for Mathematical Model Formulations.....	13
Table 3.2. Decision Variables for Mathematical Model Formulations	14
Table 3.3. Computational Results of the Model 1, Model 2, Model 3, Model 4, and Model 5 for Makespan on Taillard Instances	22
Table 3.4. Computational Results of the Model 1, Model 2, Model 3, Model 4, and Model 5 for Total Flow Time on Taillard Instances.....	24
Table 3.5. Computational Results of the Model 1, Model 2, Model 3, Model 4, and Model 5 for Total Tardiness on Taillard Instances.....	27
Table 3.6. Computational Results of the Model 1, Model 2, Model 3, Model 4, and Model 5 for Makespan on VRF Instances	30
Table 3.7. Computational Results of the Model 1, Model 2, Model 3, Model 4, and Model 5 for Total Flow Time on VRF Instances	33
Table 3.8. Computational Results of the Model 1, Model 2, Model 3, Model 4, and Model 5 for Total Tardiness on VRF Instances.....	36
Table 3.9. Comparison of Results with Valid Inequality (3.46) for Makespan	42
Table 3.10. Comparison of Results with Valid Inequality (3.49) for Makespan	42
Table 3.11. Comparison of Results with Valid Inequality (3.46) for Total Flow Time	43
Table 3.12. Comparison of Results with Valid Inequality (3.49) for Total Flow Time	43
Table 3.13. Comparison of Results with Valid Inequality (3.46) for Total Tardiness	44
Table 3.14. Comparison of Results with Valid Inequality (3.49) for Total Tardiness	44
Table 3.15. The Distance matrix of the directed graph	47
Table 3.16. Shortest path calculation of the LB_I matrix for the makespan and the total flow time	48



Table 3.17. Shortest path calculation of the LB_1 matrix for the total tardiness	49
Table 3.18. Shortest path calculation of the LB_1 matrix for the number of tardy jobs	50
Table 3.19. Shortest path calculation of the LB_1 -Lifted matrix for the makespan and the total flow time when $k=1$	52
Table 3.20. Shortest path calculation of the LB_1 -Lifted matrix for the makespan and the total flow time when $k=2$	52
Table 3.21. Shortest path calculation of the LB_1 -Lifted matrix for the makespan and the total flow time when $k=3$	53
Table 3.22. Shortest path calculation of the LB_1 -Lifted matrix for the makespan and the total flow time when $k=4$	53
Table 3.23. Shortest path calculation of the LB_1 -Lifted matrix for the makespan and the total flow time when $k=5$	54
Table 3.24. The Summary of shortest path calculation of the LB_1 -Lifted matrix for the makespan and the total flow time	55
Table 3.25. Performance of the Lower Bounds for Makespan on the Taillard Instances	56
Table 3.26. Performance of the Lower Bounds for Total Flow Time on the Taillard Instances	57
Table 3.27. Performance of the lower bounds for Total Tardiness on the Taillard Instances	58
Table 3.28. Performance of the Lower Bounds for the Number of Tardy Jobs on the Taillard Instances	59
Table 3.29. Performance of the Lower Bounds for Makespan on the VRF Instances	61
Table 3.30. Performance of the Lower Bounds for Total Flow Time on the VRF Instances	62
Table 3.31. Performance of the Lower Bounds for Total Tardiness on the VRF Instances	63
Table 3.32. Performance of the Lower Bounds for the Number of Tardy Jobs on the	



VRF Instances	64
Table 3.33. Various flowshop settings that minimize the (weighted) number of tardy jobs with due date considerations	69
Table 3.34. Various flowshop settings that minimize the (weighted) number of tardy jobs with due date considerations	70
Table 3.35. No-wait permutation flowshop settings that minimize various lateness-related objective functions with due date considerations	72
Table 3.36. No-wait permutation flowshop settings that minimize various lateness-related objective functions with due date considerations	73
Table 3.37. Issues in Existing Due Dates of the Taillard Set.....	75
Table 3.38. Example of the Sacrifice and Rearrange Heuristic on $\pi EDDs0 = \{4, 1, 2, 3, 5\}$	81
Table 3.39. Generation of other variants on $\pi EDDs0 = 4, 1, 2, 3, 5$ and $\pi E(DD - PT)s0 = \{4, 1, 2, 5, 3\}$	84
Table 3.40. Sacrifice and Rearrange Heuristics on Other Sequences	85
Table 3.41. The Summary Table of Experimentation on α Value	87
Table 3.42. Computational Analysis for MILP and CP on the Cropped Instances of Taillard Instances.....	89
Table 3.43. Computational Analysis for MILP and CP on the first 4 sets of VRF Instances	89
Table 3.44. Computational Analysis for MILP and CP on the Taillard Instances...	90
Table 3.45. Computational Analysis for MILP and CP on the Small VRF Instances	91
Table 3.46. Computational Analysis for MILP, CP, and SRH on the Cropped Instances of Taillard Instances.....	93
Table 3.47. Computational Analysis for MILP, CP, and SRH on the first 4 sets of VRF Instances.....	94
Table 3.48. Computational Analysis for MILP, CP, and SRH on the first 4 sets of VRF Instances.....	95



Table 3.49. Performance of Sacrifice & Rearrange Heuristic with EDD on Taillard Benchmark Set.....	97
Table 3.50. Performance of Sacrifice & Rearrange Heuristic with E(DD-PT) on Taillard Benchmark Set	98
Table 3.51. Performance of Sacrifice & Rearrange Heuristic with EDD on VRF Benchmark Set.....	99
Table 3.52. Performance of Sacrifice & Rearrange Heuristic on the variants of Construction Heuristic with E(DD-PT) on VRF Benchmark Set.....	100
Table 3.53. Comparison of Sacrifice & Rearrange Heuristic with the Mathematical Models on Taillard Benchmark Set	102
Table 3.54. Comparison of Sacrifice & Rearrange Heuristic with the Mathematical Models on VRF Benchmark Set	103
Table 4.1. Bi-Criteria NWPFSs	110
Table 4.2. Bi-Criteria NWPFSs	111
Table 4.3. Bi-Criteria NWPFSs	112
Table 4.4. Bi-Criteria NWPFSs	113
Table 4.5. Bi-Criteria NWPFSs	114
Table 4.6. Notations for MILP Model.....	118
Table 4.7. Computational Analysis for Cardinality on the First Four Sets of VRF Instances	138
Table 4.8. Computational Analysis for Proportion of Pareto Optimal Solution Set Discovered, Inverted General Distance, and Distribution Spacing on the First Four Sets of VRF Instances	139
Table 4.9. Computational Analysis for Cardinality Metric on Small VRF Instances	143
Table 4.10. Computational Analysis for Cardinality Metric on Large VRF Instances	144
Table 4.11. Computational Analysis for Distribution Metric on Small VRF Instances	146



Table 4.12. Computational Analysis for Distribution Metric on Large VRF Instances	147
Table 4.13. Computational Analysis for Coverage Metric on Small VRF Instances	148
Table 4.14. Computational Analysis for Coverage Metric on Small VRF Instances	149
Table 4.15. Computational Analysis for Coverage Metric on Small VRF Instances	150
Table 4.16. Computational Analysis for Coverage Metric on Large VRF Instances	151
Table 4.17. Computational Analysis for Coverage Metric on Large VRF Instances	152
Table 4.18. Computational Analysis for Coverage Metric on Large VRF Instances	153



SYMBOLS AND ABBREVIATIONS

SYMBOLS:

C_{max}	Makespan
$\sum C_i / \sum C_{iM}$	Total Flow Time
$\sum T_i$	Total Tardiness
$\sum U_i$	Number of Tardy Jobs
π	Sequence of jobs
N	Job set
M	Machine set
i, k	Indices of jobs ($1 \leq i, k \leq N$)
r, l, t	Indices of machines ($1 \leq r, l, t \leq M$)
p	Indices of positions ($1 \leq p \leq N$)
P_{ir}	Processing time of job $i \in N$ on machine $r \in M$
Δ_{Fik}	The time delay on the first machine between two consecutive jobs i and k ($i, k \in N$)
Δ_{Lik}	The time delay on the last machine between two consecutive jobs i and k ($i, k \in N$)
DD_i	Job i 's due date ($i \in N$)
Q	A big number
C_{ir}	Completion time of job i on machine r
S_{ir}	Starting time of job i on machine r
B_i	Starting time of job i on the first machine
C_i / C_{iM}	Completion time of job i on the last machine
C_{max}	Makespan
X_{ip}	1 if job i is assigned at position p , 0 otherwise



D_{ik}	1 if job i is assigned any time formerly job k , 0 otherwise ($i < k$)
δ_{ik}	1 if job i is assigned immediately before job k , 0 otherwise
T_i	Tardiness of job $i \in N$
S_p	Start time of a job at position $p \in N$
CoP_p	Completion time of a job at position $p \in N$
TP_p	Tardiness of a job at position $p \in N$
X_p	Equals to $i \in \{0, \dots, N\}$ if job $i \in \{0, \dots, N\}$ is placed in location $p \in \{0, \dots, N\}$
Job_{ir}	Interval variable related with the job i 's processing time (P_{ir}) on machine r
Mac_r	Sequence variable regarding machines overall Job_{ir}
ε	A small number
U_i	1 if job i is tardy, 0 otherwise
V_p	1 if the job at position p is tardy, 0 otherwise
$(F_m nwt C_{max})$	No-Wait Permutation Flowshop Scheduling Problem to Minimize Makespan
$(F_m nwt \sum C_i)$	No-Wait Permutation Flowshop Scheduling Problem to Minimize Total Flow Time
$(F_m nwt \sum T_i)$	No-Wait Permutation Flowshop Scheduling Problem to Minimize Total Tardiness
$(F_m nwt \sum U_i)$	No-Wait Permutation Flowshop Scheduling Problem to Minimize Number of Tardy Jobs
$(F_m nwt C_{max}, \sum C_i)$	Bi-Criteria No-Wait Permutation Flowshop Scheduling Problem to Minimize Makespan and Total Flow Time



ABBREVIATIONS:

FSP	Flowshop Scheduling Problem
PFSP	Permutation Flowshop Scheduling Problem
NWPFSP	No-Wait Permutation Flowshop Scheduling Problem
BI-CRI NWPFSP	Bi-Criteria No-Wait Permutation Flowshop Scheduling Problem
NIPFSP	No-Idle Permutation Flowshop Scheduling Problem
BPFSP	Blocking Permutation Flowshop Scheduling Problem
FJSSP	Flexible Job Shop Scheduling Problem
HFSP	Hybrid Flowshop Scheduling Problem
MILP	Mixed-Integer Linear Programming
CP	Constraint Programming
QMIP	Quadratic Mixed-Integer Programming
ATSP	Asymmetric Traveling Salesperson Problem
C-G	Chvatal-Gomory
LB	Lower Bound
LB ₁	Lower Bound 1
LB ₁ -Lifted	Lower Bound 1-Lifted
SP	Shortest Path
EDD	Earliest Due Date
IG	Iterated Greedy Algorithm
BIH	Block Insertion Heuristic
ILS	Insertion Local Search Algorithm
DE	Differential Evolution
NSGA-II	Non-Dominated Sorting Genetic Algorithm



VNS	Variable Neighborhood Search
ABC	Artificial Bee Colony Algorithm
RL	Reinforcement Learning
DRL	Deep Reinforcement Learning
BC-IG	Bi-Criteria Iterated Greedy Algorithm
BC-IG _{ALL}	Bi-Criteria Iterated Greedy Algorithm with a Local Search
BC-VBIH	Bi-Criteria Variable Block Insertion Heuristic Algorithm
BC-IG _{QL}	Bi-Criteria Iterated Greedy Algorithm with Q-learning
BC-BIH _{QL}	Bi-Criteria Block Insertion Heuristic with Q-learning
BC-GLS	Bi-Criteria Genetic Local Search Algorithm
BC-GLS _{ONECUT}	Bi-Criteria Genetic Local Search Algorithm with One-cut Crossover
BC-GLS _{TWOCUT}	Bi-Criteria Genetic Local Search Algorithm with Two-cut Crossover
BC-GLS _{OX}	Bi-Criteria Genetic Local Search Algorithm with Order Crossover
BC-GLS _{PMX}	Bi-Criteria Genetic Local Search Algorithm with Partially Mapped Crossover
BC-GLS _{PTL}	Bi-Criteria Genetic Local Search Algorithm with Particle Crossover



1. CHAPTER: INTRODUCTION

The allocation of jobs, machines, or resources to their specified task over time based on an optimization objective criterion is the research area of scheduling (Pinedo, M. L. (2012)). Scheduling is a vital point of the manufacturing environment because minimizing the cost incurred due to the operations is the fundamental goal of production. It aims to achieve specific criteria for efficiency and cost-effectiveness. The most frequently encountered processing layout in practice is the flow shop scheduling problem (FSP), which is of extensive engineering relevance. The FSPs have been a very productive and vivid research field and have taken the attention of researchers and people from the industry since the leading study of Johnson (Johnson, S. M. (1954)). Over the last 60 years, the problem resulted in nearly hundreds of algorithms, i.e., exact or heuristic algorithms, to address the complexities of scheduling tasks in a flow shop layout. In this type of scheduling problem, a group of products must follow a group of machines in a predefined order in a typical manufacturing environment (Hejazi and Saghafian (2004)).

The permutation flowshop scheduling problems (PFSP) are the most studied in FSPs and have a variety of applications in both production and service industries (Ruiz and Stützle (2007)). However, in PFSPs, there can be several restrictions in the flow shop, such as machine waiting times (Pan & Wang (2008a)) or job waiting times (Pan et al. (2008)) not being allowed during job operations, sequence-based setup times being raised or the jobs having to be distributed between various factories (Naderi and Ruiz 2010), etc. Addressing these restrictions efficiently is essential for improving production processes and overall operational performance. However, these restrictions result in various PFSP settings with different complexity levels.

This thesis investigates new solution techniques for the No-Wait Permutation Flowshop Scheduling Problem (NWPFSP). This type of scheduling problem is one of the restricted versions of PFSPs with a stipulation on the job waiting times between successive machines; that is, any waiting time is not allowed for the process of a job

on consecutive machines. A set of n jobs must be processed on m serial machines in the specified order (π) with no interruption of jobs between operations. Thus, NWPFSs aim to determine a sequence (π) for all jobs on the machines, ensuring that each job follows the same order and that there is no waiting time between consecutive machines for any job.

The contribution of this thesis to NWPFSs is twofold: *single-objective optimization* and *bi-criteria optimization*. For *single-objective optimization*, this thesis aims to develop new solution techniques for minimizing the makespan (maximum completion time) (C_{max}), the total flow time ($\sum C_i$), the total tardiness ($\sum T_i$), and the number of tardy jobs ($\sum U_i$) objective functions separately in NWPFSs. For *bi-criteria optimization*, the Bi-Criteria No-Wait Flowshop Scheduling Problem (BI-CRI NWPFS) aims to minimize both the makespan (C_{max}) and the total flow time ($\sum C_i$) simultaneously using new solution techniques.

Regarding *single-objective optimization*, the five mathematical model formulations consisting of three Mixed-Integer Linear Programming (MILP) and two Constraint Programming (CP) are studied for makespan (C_{max}), total flow time ($\sum C_i$), and total tardiness ($\sum T_i$) objective functions. Then, one of the MILP models is empowered by valid inequalities and is studied for makespan (C_{max}), total flow time ($\sum C_i$), and total tardiness ($\sum T_i$) objective functions. Next, a novel Lower Bound (LB) mechanism that is guided by the Shortest Path (SP) algorithm is developed for makespan (C_{max}), total flow time ($\sum C_i$), total tardiness ($\sum T_i$), and the number of tardy jobs ($\sum U_i$) objective functions. Following that, two mathematical models, one belonging to the MILP class and the other to the CP class, have been studied for the number of tardy jobs ($\sum U_i$) minimization in NWPFSs. Then, a novel and quick upper bound, the Sacrifice and Rearrange Heuristic (SRH), is developed from the ground up to achieve fast, feasible solutions in minimizing the number of tardy jobs ($\sum U_i$) in NWPFSs. Since due dates are crucial in maximizing performance metrics connected to lateness, optimizing the number of tardy jobs in NWPFSs necessitates decent-quality due dates. Thus, a recently developed and advantageous mechanism for generating due dates has been created to produce high-quality due dates for the NWPFSs. This mechanism also utilizes the SRH, yielding due dates of decent quality and practicality.

Regarding *bi-criteria optimization*, the NWPFSF is treated as a bi-criteria combinatorial optimization problem where the total flow time ($\sum C_i$) and makespan (C_{max}) are optimized simultaneously. First, a MILP model formulation is studied to solve this problem. Then, Q-learning-guided algorithms are developed for the Bi-CRI NWPFSFs. Q-learning is a prominent reinforcement learning method whose utilization aims to guide the selection of actions, thus eliminating the necessity for a random exploration in the iterative process of metaheuristics. The developed Q-learning guided metaheuristics are Bi-Criteria Iterated Greedy Algorithm with Q-learning (BC-IG_{QL}) and Bi-Criteria Block Insertion Heuristic Algorithm with Q-learning (BC-BIH_{QL}). Moreover, the performance of the proposed Q-learning guided algorithms is compared over a collection of Bi-Criteria Genetic Local Search Algorithms (BC-GLS), Bi-Criteria Iterated Greedy Algorithm (BC-IG), Bi-Criteria Iterated Greedy Algorithm with a Local Search (BC-IG_{ALL}) and Bi-Criteria Variable Block Insertion Heuristic Algorithm (BC-VBIH).

The following chapters of this thesis are organized as follows: Chapter 2 defines the NWPFSF and addresses the research questions of this thesis. Chapter 3 focuses on single-objective optimization. First, we provide a literature review on mathematical model formulations in NWPFSFs in Section 3.1. Then, mathematical model formulations for single-objective NWPFSFs are proposed and the computational results are provided in Sections 3.2, and 3.3, respectively. Next, valid inequalities to empower one of the MILP model formulations are discussed, and the computational results are provided in Sections 3.4, and 3.5, respectively. Following that, the Shortest Path (SP) guided Lower Bound (LB) mechanism is proposed for makespan, total flow time, total tardiness, and the number of tardy jobs in single-objective NWPFSFs, and the computational results are provided in Sections 3.6, and 3.7, respectively. Next, the number of tardy jobs is analyzed using various solution approaches in Section 3.8. We provide a literature review of lateness-related performance measurements in flowshops in Section 3.8.1. Section 3.8.2 presents the two mathematical models, one belonging to the MILP class and the other to the CP class, adapted for the number of tardy jobs minimization in NWPFSFs. The Sacrifice and Rearrange Heuristic (SRH) is introduced as a novel upper bounding heuristic for minimizing the number of tardy jobs in NWPFSFs in Section 3.8.3, along with a new due date generation mechanism in Section 3.8.4. Finally, the computational results on the number of tardy jobs

minimization in NWPFSs are provided in Section 3.8.5. Chapter 4 focuses on bi-criteria optimization. First, the bi-criteria problem setting is presented in Section 4.1, followed by a literature review of BI-CRI NWPFSs and Q-learning-based algorithms for scheduling problems in Section 4.2. The bi-criteria mathematical model formulation is provided in Section 4.3. In Section 4.4, the proposed metaheuristic algorithms are introduced, with an explanation of the Q-learning concept. Finally, the computational results of the bi-criteria problem are provided in Section 4.5. Chapter 5 discusses the findings of this thesis and provides concluding remarks, addressing how the research questions are answered. It finalizes the thesis by suggesting directions for future research.

2. CHAPTER: PROBLEM DEFINITION AND RESEARCH QUESTIONS

This chapter defines the No-Wait Permutation Flow Shop Problem (NWPFS). The problem will be detailed through specific features and constraints. One of the most well-known scheduling problems is the Permutation Flowshop Scheduling Problem (PFSP). In PFSPs, a set of n jobs with no relation are aimed to be treated on a set of m machines. The organization of machines is serial; hence, every single job must go through all machines in identical order. Following this, the No-Wait Flowshop Scheduling Problems (NWPFS) is a particular form among all PFSPs. NWPFSs, as a modified version of PFSPs, have recently taken significant attention from both the practical and academic communities due to their extensive applications across various industries (Aldowaisan and Allahverdi (2003)). The modification in NWPFSs is that no intermediate queues are permitted between consecutive machines (Fink and Voß (2003)). The most significant practical applications of NWPFSs are found in industries such as steel, chemicals, plastics, food processing, hot metal rolling, pharmaceuticals, concrete production, and electronics (Sapkal & Laha (2013)). The systems observed in the abovementioned industries can be considered no-wait processes because they do not permit in-process inventory. In other words, the technological constraints in the shop floor environment transform these issues into no-wait problems. Many specific features of the problem are listed below.

- All jobs are ready to be processed initially.
- A job can be handled solely by a machine at a particular time; inherently, a machine can treat a job at a specific time.
- The order of jobs is identical on all machines.
- Each job either proceeds a job or succeeds in a job.
- Jobs may not be waited between successive machine operations. Once jobs begin to be processed, they must be completed until the last machine, which alludes to the job's no-wait limitation.

- Hence, intermediate storage of consecutive machines does not exist in the floor shop in this problem.
- Operations are not preemptible and indivisible. No setup time exists. All jobs are ready at the beginning.

Figure 2.1 depicts a visualization of NWPFSF with 4 jobs on 3 machines for the sequence $\pi = \{1,3,2,4\}$. The processing times for the job $i \in N$ on machine $r \in M$ (P_{ir}) and the due date of job $i \in N$ (DD_i) in this example are as follows.

$$P_{ir} = \begin{bmatrix} 2 & 3 & 3 \\ 4 & 3 & 5 \\ 2 & 5 & 2 \\ 7 & 2 & 2 \end{bmatrix} \quad DD_i = [7 \ 12 \ 13 \ 8]$$



Figure 2.1. An example for NWPFSF with 4 jobs on 3 machines

Source: Author

In scheduling problems, the standard 3-tuple notation of Graham et al. (1979) is followed: $\alpha|\beta|\gamma$. The first element, α , indicates the shop configuration. Since this thesis addresses general m -machine permutation flow shop scheduling problems, F_m is carried out for α . The second element, β , expresses both the constraints and the specific nature of the problem. Thus, nwt denotes the no-wait limitation for β . The third element, γ , illustrates the interested *service level*, which is the performance criterion of the objective function. Several objective functions can be minimized, such as maximum completion time, maximum flow time, total flow time, total weighted completion time, maximum earliness, etc. In short, $(F_m|nwt|ServiceLevel)$ represents the NWPFSFs.

A variety of solution approaches are available when addressing combinatorial optimization problems. Key approaches include developing lower bounding schemes, mathematical model formulations like MILP and CP, upper bounds, and metaheuristics. These techniques form the foundation of various perspectives in the field. This thesis addresses several critical research questions in the realm of

scheduling problems, specifically focusing on developing new solution techniques for the $(F_m|nwt|ServiceLevel)$ problems. The primary question is whether we can generate more effective mathematical models, such as MILP and CP model formulations, for these problems. Additionally, this thesis explores the possibility of enhancing one of the MILP model formulations by incorporating valid inequalities to improve their performance. Another significant area of investigation is the generation of new lower bounds for the $(F_m|nwt|ServiceLevel)$ problems. The other important field of study is to develop effective heuristics as quick upper bounds for the $(F_m|nwt|ServiceLevel)$ problems. This research also aims to develop a due date generation mechanism and establish robust due dates to address lateness-related performance measures effectively. Furthermore, this thesis examines the potential to extend these studies into a bi-objective framework, allowing for the simultaneous consideration of two service levels, $(F_m|nwt|ServiceLevel_1, SerciveLevel_2)$. Finally, this research investigates the creation of Q-learning guided metaheuristic algorithms for tackling the bi-objective $(F_m|nwt|ServiceLevel_1, SerciveLevel_2)$ problem. To summarize, seven research questions in the light of research ideas can be outlined as follows:

Research Questions:

1. Can we generate more effective *mathematical models (MILP and CP model formulations)* for the $(F_m|nwt|ServiceLevel)$ problems?
2. Can we enhance the mathematical models developed for the $(F_m|nwt|ServiceLevel)$ problems by incorporating *valid inequalities*?
3. Can we generate new *lower bounds* for the $(F_m|nwt|ServiceLevel)$ problems?
4. Can we develop effective heuristics as quick *upper bounds* for the $(F_m|nwt|ServiceLevel)$ problems?
5. Can we create a *due date generation mechanism* and establish *strong due dates* to address lateness-related performance measures in $(F_m|nwt|ServiceLevel)$ problems?
6. Can we extend these studies in a *bi-objective framework* in such a way that two service levels are carried out simultaneously; i.e., $(F_m|nwt|ServiceLevel_1, SerciveLevel_2)$?

7. Can we create *Q-learning guided metaheuristic algorithms* for the bi-objective $(F_m | nwt | ServiceLevel_1, SerciveLevel_2)$ problem?

Four service levels are addressed in this thesis: makespan (C_{max}), total flow time ($\sum C_i$), total tardiness ($\sum T_i$), and the number of tardy jobs ($\sum U_i$).

- **Makespan (C_{max})** : The maximum completion time of the jobs, focusing on optimizing resource utilization. Minimizing the makespan is a key objective in scenarios where a batch of jobs received simultaneously needs to be completed as soon as possible (Allahverdi & Aydilek (2013)). Minimizing the makespan also enhances the utilization of machines and resources.
- **Total flow time ($\sum C_i$)** : The sum of the flow times of each job, prioritizing quickly meeting the demands (Aydilek & Allahverdi (2012)). This criterion is especially crucial in practical scenarios where minimizing inventory or holding costs is a top priority (Allahverdi & Aydilek (2013)). It is important to note that when the release time of jobs is not considered, the flow time and the completion time of jobs are equal. Typically, the jobs are ready at time zero in the flow shops. Hence, in the literature, the total flow time is mainly equivalent to the total completion time unless otherwise specified (when the jobs are ready at time zero).
- **Total tardiness ($\sum T_i$)** : The sum of the time each job is completed after its due date. Total tardiness quantifies the lateness of jobs (Aldowaisan & Allahverdi, (2012a)). In numerous real-world scenarios, costs rise with the length of delays, often in the form of financial penalties for different delay periods. Thus, total tardiness plays a significant role in situations where financial penalties depend on the duration of the delay (Aldowaisan & Allahverdi (2015)).
- **Number of tardy jobs ($\sum U_i$)** : The count of completed jobs after their respective due dates. It counts the tardy jobs that are not completed on time. Since on-time shipments are essential for reducing costs and raising customer satisfaction for practically all production systems, reducing the number of tardy jobs is crucial. For each late task, manufacturers typically have to pay penalties. Regardless of how long a job is delayed, the penalty is always the same in some production environments (Aldowaisan & Allahverdi (2012b)). For all these reasons, optimizing the number of tardy jobs plays a considerable role in

situations where it directly impacts costs related to late delivery and the percentage of on-time shipments.

Total tardiness and number of tardy jobs are lateness-related performance measurements. High-quality due dates are essential to optimize performance metrics related to lateness in scheduling problems. Due dates play a crucial role in improving these performance measures.

For the given sequence $\pi = \{1,3,2,4\}$ in Figure 2.1:

- **Makespan (C_{max})** : 21.
- **Total flow time ($\sum C_i$)** : 59, computed as $(8 - 0) + (12 - 0) + (18 - 0) + (21 - 0)$ since the jobs are ready at time zero.
- **Total tardiness ($\sum T_i$)** : 20, computed as $\max\{(8 - 7), 0\} + \max\{(12 - 13), 0\} + \max\{(18 - 12), 0\} + \max\{(21 - 8), 0\}$.
- **Number of tardy jobs ($\sum U_i$)** : 3, computed as $\min\{1, \max\{(8 - 7), 0\}\} + \min\{1, \max\{(12 - 13), 0\}\} + \min\{1, \max\{(18 - 12), 0\}\} + \min\{1, \max\{(21 - 8), 0\}\}$.

This thesis aims to develop new solution techniques for minimizing makespan (C_{max}), total flow time ($\sum C_i$), total tardiness ($\sum T_i$), and the number of tardy jobs ($\sum U_i$) objective functions in the context of single-objective optimization in NWPFSs. Additionally, it seeks to simultaneously minimize makespan and total flow time through bi-criteria optimization in NWPFSs. In other words, this thesis aims to contribute to several aspects of NWPFSs through new solution techniques. We address single-objective optimization problems including $(F_m|nwt|C_{max})$, $(F_m|nwt|\sum C_i)$, $(F_m|nwt|\sum T_i)$, and $(F_m|nwt|\sum U_i)$ problems, as well as bi-criteria optimization in the $(F_m|nwt|C_{max}, \sum C_i)$ problem, guided by the research questions.



3. CHAPTER: SINGLE-OBJECTIVE NO-WAIT PERMUTATION FLOWSHOP SCHEDULING PROBLEMS

This chapter explores the NWPFS in the context of single-objective minimization. Initially, Section 3.1 offers a literature review on mathematical model formulations in NWPFSs. Sections 3.2 and 3.3 propose mathematical model formulations for single-objective NWPFSs and present the corresponding computational results for makespan, total flow time, and total tardiness objective functions. In Sections 3.4 and 3.5, valid inequalities to enhance one of the MILP model formulations are discussed. Their computational results are provided for makespan, total flow time and total tardiness objective functions. Following this, Sections 3.6 and 3.7 introduce the Shortest Path (SP) guided Lower Bound (LB) mechanism for makespan, total flow time, total tardiness, and the number of tardy jobs in single-objective NWPFSs, along with their computational results. Section 3.8 analyzes the objective function for the number of tardy jobs through various solution approaches. Section 3.8.1 reviews lateness-related performance measurements in flowshops. Section 3.8.2 presents two mathematical models, one from the MILP class and the other from the CP class, adapted for minimizing the number of tardy jobs in NWPFSs. Section 3.8.3 introduces the Sacrifice and Rearrange Heuristic (SRH) as a novel upper bounding heuristic for minimizing tardy jobs, along with a new due date generation mechanism detailed in Section 3.8.4. Finally, Section 3.8.5 provides the computational results on minimizing the number of tardy jobs in NWPFSs.

3.1. Literature Review of Mathematical Model Formulations in NWPFSs

Röck, H. (1984) demonstrated that as the problem size increases, the NWPFS requires a non-polynomial amount of time to solve. This means that mathematical models cannot find optimal solutions in polynomial time for larger instances. However, different mathematical models yield varying objective values and gap percentages within a set time limit. Developing mathematical models is a foundational step in creating effective heuristics. Consequently, researchers aim to find more

advanced models and faster computation times that can outperform others in terms of objective values or gap percentages. Therefore, strengthening formulations for scheduling problems is a promising area of research in combinatorial optimization.

Various studies have developed diverse mathematical models in the literature on mathematical model formulations in NWPFSs. Samarghandi (2015) studied a mixed-integer linear programming (MILP) model to minimize the makespan with hard due date restrictions. Later, Samarghandi and Behrooz (2017) explored two quadratic mixed-integer programming (QMIP) models and two constraint programming (CP) model formulations, all focusing on makespan minimization with strict due date constraints for each job.

A comprehensive literature review by Lin and Ying (2016) investigated the makespan criterion by transforming the problem into an asymmetric traveling salesperson problem (ATSP). This approach allowed for optimal solutions using two metaheuristics, including an MILP model for the ATSP. Their study provided optimal results for makespan minimization. Building on this, Ying and Lin (2018) studied the NWPFS with sequence-dependent and sequence-independent setup times, applying the same ATSP transformation. They reported the optimal results for these problems as well. These studies highlight significant progress in mathematical model formulations for NWPFSs, emphasizing makespan minimization and the innovative transformation of problems to optimize solutions. Additionally, a MILP model was previously studied in the M.Sc. thesis of Yüksel, D. (2019). This model is based on Manne's (1960) fundamental PFSP model, with the addition of a no-wait constraint. Yüksel, D. (2019) presented the outcomes of this MILP model for makespan, total flow time, and total tardiness. Subsequently, this model is adapted for green NWPFSs to minimize the energy released during production, as explored in the studies by Yüksel et al. (2020a), Tasgetiren et al. (2019), and Yüksel et al. (2020b). The energy-efficient NWPFS setting is bi-criteria, with objective functions that include makespan and total energy consumption, total flow time and total energy consumption, and total tardiness and total energy consumption, respectively.

Regarding the CP perspective, a CP model was initially studied for NWPFSs by Yüksel, D. (2019). Computational results for bi-criteria energy-efficient NWPFSs, focusing on total tardiness and total energy consumption objective functions, were provided by Yüksel et al. (2020b). This CP model utilizes the *endAtStart*,

noOverlap, *sameSequence*, and *endOf* functions of the CP library. As mentioned earlier in the study by Samarghandi and Behrooz (2017), two CP models were developed to minimize makespan with hard due date constraints. These CP models utilize the *allDifferent* function of the CP library.

3.2. Mathematical Model Formulations

This section proposes a comparative analysis of various mathematical model formulations on NWPFSs for different objective functions. Three Mixed-Integer Linear Programming (MILP) models and two Constraint Programming (CP) models are studied on the NWPFSs for the three objective functions: makespan (C_{max}), total flow time ($\sum C_i$), and total tardiness ($\sum T_i$). Among five mathematical models, two are newly proposed to the literature on NWPFSs, and three of them have been previously studied in the literature for the variants of the NWPFSs. The sets, indices, parameters, and decision variables used in developing the mathematical models are detailed in Tables 3.1 and 3.2.

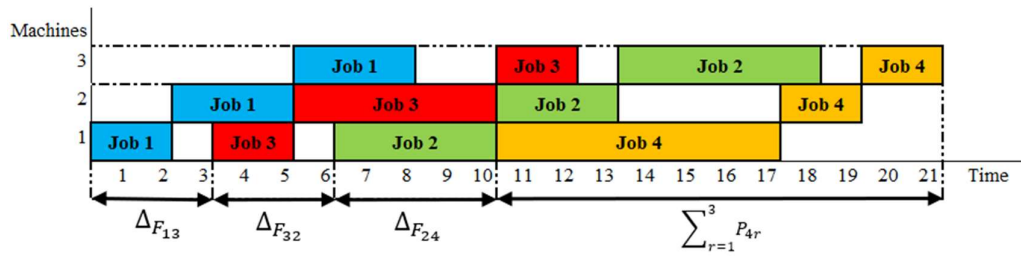
Table 3.1. Sets, Indices, and Parameters for Mathematical Model Formulations

Sets and Indices	
N	Job set
M	Machine set
i, k	Indices of jobs ($1 \leq i, k \leq N$)
r, l, t	Indices of machines ($1 \leq r, l, t \leq M$)
p	Indices of positions ($1 \leq p \leq N$)
Parameters	
P_{ir}	Processing time of job $i \in N$ on machine $r \in M$
Δ_{Fik}	The time delay on the first machine between two consecutive jobs i and k ($i, k \in N$)
Δ_{Lik}	The time delay on the last machine between two consecutive jobs i and k ($i, k \in N$)
DD_i	Job i 's due date ($i \in N$)
Q	A big number

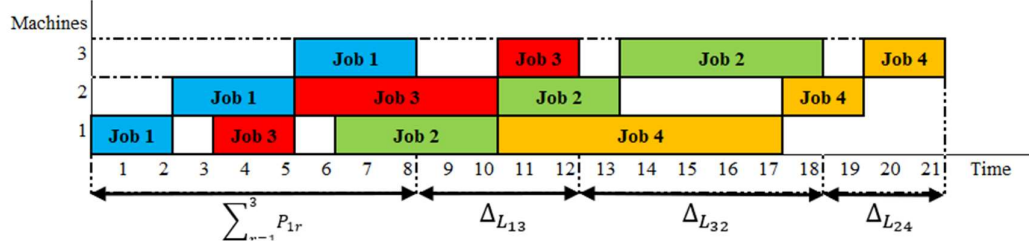
Table 3.2. Decision Variables for Mathematical Model Formulations

Decision Variables	
C_{ir}	Completion time of job i on machine r
S_{ir}	Starting time of job i on machine r
B_i	Starting time of job i on the first machine
C_i / C_{iM}	Completion time of job i on the last machine
C_{max}	Makespan
X_{ip}	1 if job i is assigned at position p , 0 otherwise
D_{ik}	1 if job i is assigned any time formerly job k , 0 otherwise ($i < k$)
δ_{ik}	1 if job i is assigned immediately before job k , 0 otherwise
T_i	Tardiness of job $i \in N$
S_p	Start time of a job at position $p \in N$
CoP_p	Completion time of a job at position $p \in N$
TP_p	Tardiness of a job at position $p \in N$
X_p	Equals to $i \in \{0, \dots, N\}$ if job $i \in \{0, \dots, N\}$ is placed in location $p \in \{0, \dots, N\}$
Job_{ir}	Interval variable related with the job i 's processing time (P_{ir}) on machine r
Mac_r	Sequence variable regarding machines overall Job_{ir}

How the values of $\Delta_{F_{ik}}$ and $\Delta_{L_{ik}}$ are calculated is illustrated in Figure 3.1.



(a) The time delays of consecutive jobs on the first machine ($\Delta_{F_{ik}}$)



(b) The time delays of consecutive jobs on the last machine ($\Delta_{L_{ik}}$)

Figure 3.1 $\Delta_{F_{ik}}$ and $\Delta_{L_{ik}}$ values of the example with 4 jobs on 3 machines

Source: Author

In Figure 3.1, it can be perceived that the time delay between the starting times of two successive jobs on the first machine ($\Delta_{F_{ik}}$) is subjected to only the processing times of both jobs. Similarly, the time delay between the completion times of two successive jobs on the last machine ($\Delta_{L_{ij}}$) is subjected to only the processing times of both jobs. Eventually, the time delays of two successive jobs on both the first machine ($\Delta_{F_{ik}}$) and the last machine ($\Delta_{L_{ij}}$) are not contingent on either the sequence of the other jobs or the positions of the jobs in the permutation (Wisner, D. A. (1972); Pan et al. (2008)) and hence can be calculated previously:

$\Delta_{F_{ik}} = \{(\text{the starting time of job } k \text{ on the first machine}) - (\text{the starting time of job } i \text{ on the first machine})\}$, if job k is processed immediately after job i .

$\Delta_{L_{ik}} = \{(\text{the completion time of job } k \text{ on the last machine}) - (\text{the completion time of job } i \text{ on the last machine})\}$, if job k is processed immediately after job i .

Five different mathematical models are studied on NWPFPSPs, which are brand-new to the literature, and three of which have already been examined for some NWPFPSP variations. Model 1, a MILP model, was previously studied by Yüksel, D. (2019) and adapted from Manne's (Manne, A. S. (1960)) essential PFSPs model additionally with a no-wait constraint. The outcomes of this model for makespan, total flow time, and total tardiness were previously provided in Yüksel, D. (2019). This model is then investigated for energy-efficient NWPFPSPs by the studies of Yüksel et al. (2020a), Tasgetiren et al. (2019), and Yüksel et al. (2020b). Model 2 is an entirely new MILP model that establishes a relationship between the starting times of jobs on the first machine. Model 3 is a new MILP model designed to optimize the starting times of jobs

at each position. Model 4 is a CP model initially studied for NWPFSs (Yüksel, D. (2019)) and other results for bi-objective energy-efficient NWPFSs are provided in Yüksel et al. (2020b). Thus, Model 4 is a plain version in which we optimize a single objective. Model 5 is another CP model adapted from one of the models proposed by Samarghandi & Behroozi (2017). Again, this model is a simplified version of the one proposed by Samarghandi & Behroozi (2017), with the hard due date constraints excluded. Five mathematical model formulations are presented below.

Model 1. The first MILP model		
Objective		
Minimize C_{max} or $\sum_{i \in N} C_{iM}$ or $\sum_{i \in N} T_i$		(3.1)
Constraints		
$C_{i1} \geq P_{i1}$	$\forall i \in N$	(3.2)
$C_{ir} - C_{i,r-1} \geq P_{ir}$	$\forall i \in N, \forall r \in M: r \geq 2$	(3.3)
$C_{ir} - C_{kr} + QD_{ik} \geq P_{ir}$	$\forall i \in N: k > i, \forall r \in M$	(3.4)
$C_{ir} - C_{kr} + QD_{ik} \leq Q - P_{kr}$	$\forall i \in N: k > i, \forall r \in M$	(3.5)
$C_{ir} - C_{i,r-1} \leq P_{ir}$	$\forall i \in N, \forall r \in M: r \geq 2$	(3.6)
$C_{max} \geq C_{iM}$	$\forall i \in N$	(3.7)
$T_i \geq C_{iM} - DD_i$	$\forall i \in N$	(3.8)
$T_i \geq 0$	$\forall i \in N$	(3.9)
$C_{ir} \geq 0$	$\forall i \in N, \forall r \in M$	(3.10)
$D_{ik} \in \{0,1\}$	$\forall i, k \in N: k > i$	(3.11)

Equation (3.1) minimizes either makespan, the total flow time or the total tardiness. Constraints (3.2)-(3.5) are the good-old PFSPs formulation of Manne, A. S. (1960). Constraint (3.6) ensures that each job's completion time on machine r cannot happen after its processing time on machine r plus its completion time on the former machine $r - 1$. Thus, the no-wait limitation is offered with both constraints (3.3) and (3.6). In other words, each job's processing time on machine r is determined by the time it took to complete on machine r minus machine $r - 1$. Makespan is computed by constraint (3.7). Each job's tardiness is calculated using constraint (3.8). According to restriction (3.8), there is no tardiness for a job if it is completed early. The sign and variable restrictions are provided by constraints (3.9), (3.10), and (3.11).

Using this formulation, constraints (3.2), (3.3), (3.4), (3.5), (3.6), (3.7), (3.10), and (3.11) define the model for makespan (C_{max}) minimization when it serves as the objective function. Constraints (3.2), (3.3), (3.4), (3.5), (3.6), (3.10), and (3.11) outline the model for total flow time ($\sum C_i$) minimization when it is the objective function. For total tardiness ($\sum T_i$), constraints (3.2), (3.3), (3.4), (3.5), (3.6), (3.8), (3.9), (3.10), and (3.11) constitute the model.

Model 2. MILP model		
Objective		
Minimize C_{max} or $\sum_{i \in N} C_i$ or $\sum_{i \in N} T_i$		(3.12)
Constraints		
$\sum_{i=1:i \neq k}^N \delta_{ik} \leq 1$	$\forall k \in N$	(3.13)
$\sum_{i=1:i \neq k}^N \delta_{ki} \leq 1$	$\forall k \in N$	(3.14)
$Q * (1 - \delta_{ik}) + B_k - B_i \geq \Delta_{F_{ik}}$	$\forall i, k \in N: i \neq k$	(3.15)
$\sum_{i=1:i \neq k}^N \sum_{k=1}^N \delta_{ik} = N - 1$		(3.16)
$C_{max} \geq C_i$	$\forall i \in N$	(3.17)
$C_i \geq B_i + \sum_{r \in M} P_{ir}$	$\forall i \in N$	(3.18)
$T_i - B_i \geq \sum_{r \in M} P_{ir} - DD_i$	$\forall i \in N$	(3.19)
$B_i \geq 0$	$\forall i \in N$	(3.20)
$\delta_{ik} \in \{0,1\}$	$\forall i, k \in N$	(3.21)

The objective function (3.12) minimizes either makespan, the total flow time or the total tardiness. Constraints (3.13) and (3.14) indicate the sequence relation of two consecutive jobs. Constraint (3.15) suggests that if two jobs are successive, then the difference between their starting times on the first machine will be at least the time delay of those jobs on the first machine ($\Delta_{F_{ik}}$). $\Delta_{F_{ik}}$ values for two consecutive jobs i and k are depicted in Figure 3.1(a). These values are predetermined values to the problem since they only rely on two successive jobs' processing times values (Wisner, D. A. (1972); Pan et al. (2008)). Constraint (3.16) restricts the binary variables to the number of jobs minus 1. The makespan of the sequence and the completion time of each job in the sequence are computed by constraints (3.17) and (3.18). The tardiness of a job is calculated by computing its completion time minus its due date in constraint (3.19). Constraints (3.20) and (3.21) are the sign restrictions and the binary variables,

respectively. This MILP model considers sequencing jobs on the first machine regarding the $\Delta_{F_{ik}}$ values.

Using this formulation, constraints (3.13), (3.14), (3.15), (3.16), (3.17), (3.18), (3.20), and (3.21) define the model for makespan (C_{max}) minimization when it serves as the objective function. Constraints (3.13), (3.14), (3.15), (3.16), (3.18), (3.20), and (3.21) outline the model for total flow time ($\sum C_i$) minimization when it is the objective function. For total tardiness ($\sum T_i$), constraints (3.13), (3.14), (3.15), (3.16), (3.18), (3.19), (3.20), and (3.21) constitute the model.

Model 3. MILP model		
Objective		
Minimize C_{max} or $\sum_{p \in N} CoP_p$ or $\sum_{p \in N} T_p$		(3.22)
Constraints		
$\sum_{p=1}^N X_{ip} = 1$	$\forall i \in N$	(3.23)
$\sum_{i=1}^N X_{ip} = 1$	$\forall p \in N$	(3.24)
$S_p + \sum_{i=1}^N \sum_{l=1}^r P_{il} * X_{ip} \geq S_{p-1} + \sum_{i=1}^N \sum_{l=1}^r P_{il} * X_{i,p-1} + \sum_{i=1}^N P_{ir} * X_{ip}$	$\forall p \in \{2, \dots, N\}$ $\forall r \in M$	(3.25)
$C_{max} = S_p + \sum_{i=1}^N \sum_{r=1}^M P_{ir} * X_{ip}$		(3.26)
$CoP_p = S_p + \sum_{i=1}^N \sum_{r=1}^M P_{ir} * X_{ip}$	$\forall p \in N$	(3.27)
$TP_p = \max \{(S_p + \sum_{i=1}^N \sum_{r=1}^M P_{ir} * X_{ip} - \sum_{i=1}^N DD_i * X_{ip}), 0\}$	$\forall p \in N$	(3.28)
$S_p, CoP_p, TP_p \geq 0$	$\forall p \in N$	(3.29)
$X_{ip} \in \{0,1\}$	$\forall i, p \in N$	(3.30)

The objective function (3.22) minimizes either makespan, the total flow time of jobs at each position or the total tardiness of jobs at each position. Constraints (3.23) and (3.24) assign jobs to precisely one position, and one position is preserved for only one job. Constraint (3.25) controls the completion times of all jobs in the subsequent positions, ensuring that these completion times are larger than those of previous positions. The completion times are calculated by adding the starting times of the jobs at positions plus the processing times until the associated machine. Makespan, the completion time of any job at a position p , and the tardiness of any job at a position p are computed by constraints (3.26), (3.27), and (3.28). Constraints (3.29) and (3.30) are the sign restrictions and the binary variables, respectively. This model allocates

jobs to positions based on their process times, with the condition that no waiting is allowed.

Using this formulation, constraints (3.23), (3.24), (3.25), (3.26), and (3.30) define the model for makespan (C_{max}) minimization when it serves as the objective function. Constraints (3.23), (3.24), (3.25), (3.27), (3.29), and (3.30) outline the model for total flow time ($\sum C_i$) minimization when it is the objective function. For total tardiness ($\sum T_i$), constraints (3.23), (3.24), (3.25), (3.28), (3.29), and (3.30) constitute the model.

Model 4. CP Model		
Objective		
Minimize C_{max} or $\sum_{i \in N} C_i$ or $\sum_{i \in N} T_i$		(3.31)
Constraints		
$endAtStart(Job_{ir}, Job_{i,r+1})$	$\forall i \in N, \forall r \in M : r < M$	(3.32)
$noOverlap(Mac_r)$	$\forall r \in M$	(3.33)
$sameSequence(Mac_1, Mac_r)$	$\forall r \in M : 1 < r$	(3.34)
$C_{max} = \max_{i \in N}(ENDOF(Job_{iM}))$		(3.35)
$C_i = ENDOF(Job_{iM})$	$\forall i \in N$	(3.36)
$T_i = \max((ENDOF(Job_{iM}) - DD_i), 0)$	$\forall i \in N$	(3.37)

In Model 4, as the first CP formulation, the objective function (3.31) minimizes either makespan, the total flow time, or the total tardiness. The no-wait constraint (3.32) states that any given job i on machine r will have a job interval that ends at the beginning of a job interval for the same job i on the posterior machine $r + 1$. Each machine is only capable of handling one job at a time due to the guarantee provided by constraint (3.33) that there cannot be any overlap between the machines. Constraint (3.34) preserves the exact sequence of the jobs on every single machine. Constraint (3.35) defines the makespan calculation as the maximum of the ending times of job intervals on the final machine. The ending times of job intervals on the final machine represent the completion time of each job, as illustrated in constraint (3.36). Tardiness is defined as the maximum of either 0 or the difference between the ending of job intervals on the final machine and the corresponding due dates, as specified in constraint (3.37). This model is a sequencing-based CP formulation for NWPFSs.

Using this formulation, constraints (3.32), (3.33), (3.34), and (3.35) define the model for makespan (C_{max}) minimization when it serves as the objective function. Constraints (3.32), (3.33), (3.34), and (3.36) outline the model for total flow time ($\sum C_i$) minimization when it is the objective function. For total tardiness ($\sum T_i$), constraints (3.32), (3.33), (3.34), and (3.37) constitute the model.

Model 5. CP model		
Objective		
Minimize C_{max} or $\sum_{i \in N} C_i$ or $\sum_{i \in N} T_i$		(3.38)
Constraints		
$allDifferent(X_1, X_2, \dots, X_n)$		(3.39)
$S_{X_{(i+1)}r} \geq S_{X_{(i)}r} + P_{X_{(i)}r}$	$\forall r \in M, \forall i \in \{1..N-1\}$	(3.40)
$S_{i(r+1)} = S_{ir} + P_{ir}$	$\forall r \in \{1..M-1\}, \forall i \in N$	(3.41)
$C_{max} = \max_{i \in N} \{S_{iM} + P_{iM}\}$		(3.42)
$C_i = S_{iM} + P_{iM}$	$\forall i \in N$	(3.43)
$T_i = \max \{(S_{iM} + P_{iM} - DD_i), 0\}$	$\forall i \in N$	(3.44)
$S_{ir}, C_i, T_i \geq 0$	$\forall i \in N, \forall r \in M$	(3.45)

The objective function (3.38) minimizes either the makespan, the total flow time or the total tardiness. Constraint (3.39) assigns only one job to a position, and each is reserved for only one job. With this constraint, a new feature of CP, that is the *allDifferent* function, is explored. The jobs should not be overlapped and concerned by the constraint (3.39). Constraints (3.39) and (3.40) are adapted from the study of Samarghandi & Behrooz (2017). Then, the no-wait restriction is ensured by constraint (3.41). Constraint (3.42) computes the maximum completion time of jobs whereas the completion time of each job is calculated by constraint (3.43). The tardiness of a job is calculated by constraint (3.44). Finally, the sign restrictions are ensured by constraint (3.45). Note that in this formulation X_p takes a value from the job set ($X_p = i \in \{0, \dots, N\}$) if job $i \in \{0, \dots, N\}$ is placed in a location $p \in \{0, \dots, N\}$ where N positions exist. Namely, this CP formulation assigns jobs to positions with process times and no wait restriction. It essentially represents the CP approach of Model 3-MILP model.

Using this formulation, constraints (3.39), (3.40), (3.41), and (3.42) define the model for makespan (C_{max}) minimization when it serves as the objective function. Constraints (3.39), (3.40), (3.41), (3.43), and (3.45) outline the model for total flow time ($\sum C_i$) minimization when it is the objective function. For total tardiness ($\sum T_i$), constraints (3.39), (3.40), (3.41), (3.44), and (3.45) constitute the model.

3.3. Computational Analysis of Mathematical Model Formulations

Mathematical models are studied for three essential objective functions: makespan, total flow time, and total tardiness. The instance sets those are presented by Taillard, E. (1993) and Vallada et al. (2015) are used as benchmark problems in the experimentation of mathematical model formulations.

- ***The first benchmark problem*** - that is Taillard, E. (1993) - comprises 11 collections of problems in which each collection has 10 instances and varying sizes of 20, 50, 100, and 200 jobs with 5, 10, and 20 machines. In addition, three more sets of small instances (30 instances in total) are carried out. These small instances, with sizes of 5-5, 5-10, and 5-20, are cropped from the Taillard set of 20-5, 20-10, and 20-20 instances by a sound truncation mechanism proposed by Yüksel, D. (2019); Yüksel et al. (2020b), which means that we carry out 140 problem instances of Taillard, E. (1993) through this section.
- ***The second benchmark problem*** - Vallada et al. (2015) - comprises 24 small problems in which each collection has 10 instances and varying sizes of 10, 20, 30, 40, 50, and 60 jobs with 5, 10, 15, and 20 machines. Hence, 240 problem instances by the Vallada et al. (2015) are followed up through this section. Hereafter, Vallada et al. (2015) benchmark set will be called the VRF instances. The due dates used for minimizing total tardiness in the VRF instances are based on those generated in Section 3.8.4 of this thesis.

Each instance is executed for 3600 seconds by five mathematical models for three objective functions in Optimization Studio of IBM ILOG CPLEX.

Regarding the Taillard benchmark set:

All the details about the instance-based results in Taillard dataset can be found in the following link: <https://data.mendeley.com/datasets/4mvrpsws7h/1>. Set-based averages of objective functions and gap percentages are reported in Tables 3.3, 3.4,

and 3.5 for makespan, total flow time, and total tardiness. The last row of tables represents the overall averages to attain an ultimate overview of the mathematical models.

Table 3.3. Computational Results of the Model 1, Model 2, Model 3, Model 4, and Model 5 for Makespan on Taillard Instances

	Model 1		Model 2		Model 3		Model 4		Model 5	
	Objective Value	Gap %	Objective Value	Gap %	Objective Value	Gap %	Objective Value	Gap %	Objective Value	Gap %
5-5 Avg	542.10	0.00%	542.10	0.00%	542.10	0.00%	542.10	0.00%	542.10	0.00%
5-10 Avg	829.10	0.00%	829.10	0.00%	829.10	0.00%	829.10	0.00%	829.10	0.00%
5-20 Avg	1424.80	0.00%	1424.80	0.00%	1424.80	0.00%	1424.80	0.00%	1424.80	0.00%
20-5 Avg	1508.30	43.47%	1503.60	75.56%	1488.80	9.07%	1481.30	17.62%	1615.40	77.36%
20-10 Avg	2019.40	38.07%	2038.70	68.00%	2014.40	20.26%	1994.90	27.40%	2248.30	71.04%
20-20 Avg	2996.00	33.55%	3019.90	59.06%	3016.40	24.43%	2995.80	34.88%	3393.70	63.55%
50-5 Avg	3814.30	82.46%	3953.00	90.11%	3607.80	24.02%	3310.80	17.25%	4363.40	91.05%
50-10 Avg	5037.20	79.96%	5315.90	86.56%	4936.30	39.85%	4328.80	31.57%	6083.70	88.25%
50-20 Avg	6980.10	75.32%	7485.50	82.94%	6912.90	49.46%	6013.50	41.94%	8836.20	85.55%
100-5 Avg	8024.90	92.36%	8576.90	95.43%	7137.10	26.62%	6366.40	17.76%	9197.20	95.74%
100-10 Avg	no solution	-	no solution	-	10282.30	45.35%	8266.50	32.02%	12596.00	94.27%
100-20 Avg	no solution	-	no solution	-	7286343.60	99.80%	11093.40	44.58%	17429.70	92.38%
200-10 Avg	no solution	-	no solution	-	14266659.50	99.61%	16120.30	34.03%	25647.70	97.13%
200-20 Avg	no solution	-	no solution	-	20933005.80	99.94%	21398.60	47.84%	34627.10	96.14%
Overall Avg.	4340.03	63.60%	4556.21	79.66%	3865945.90	48.95%	7579.12	31.54%	11458.04	86.59%

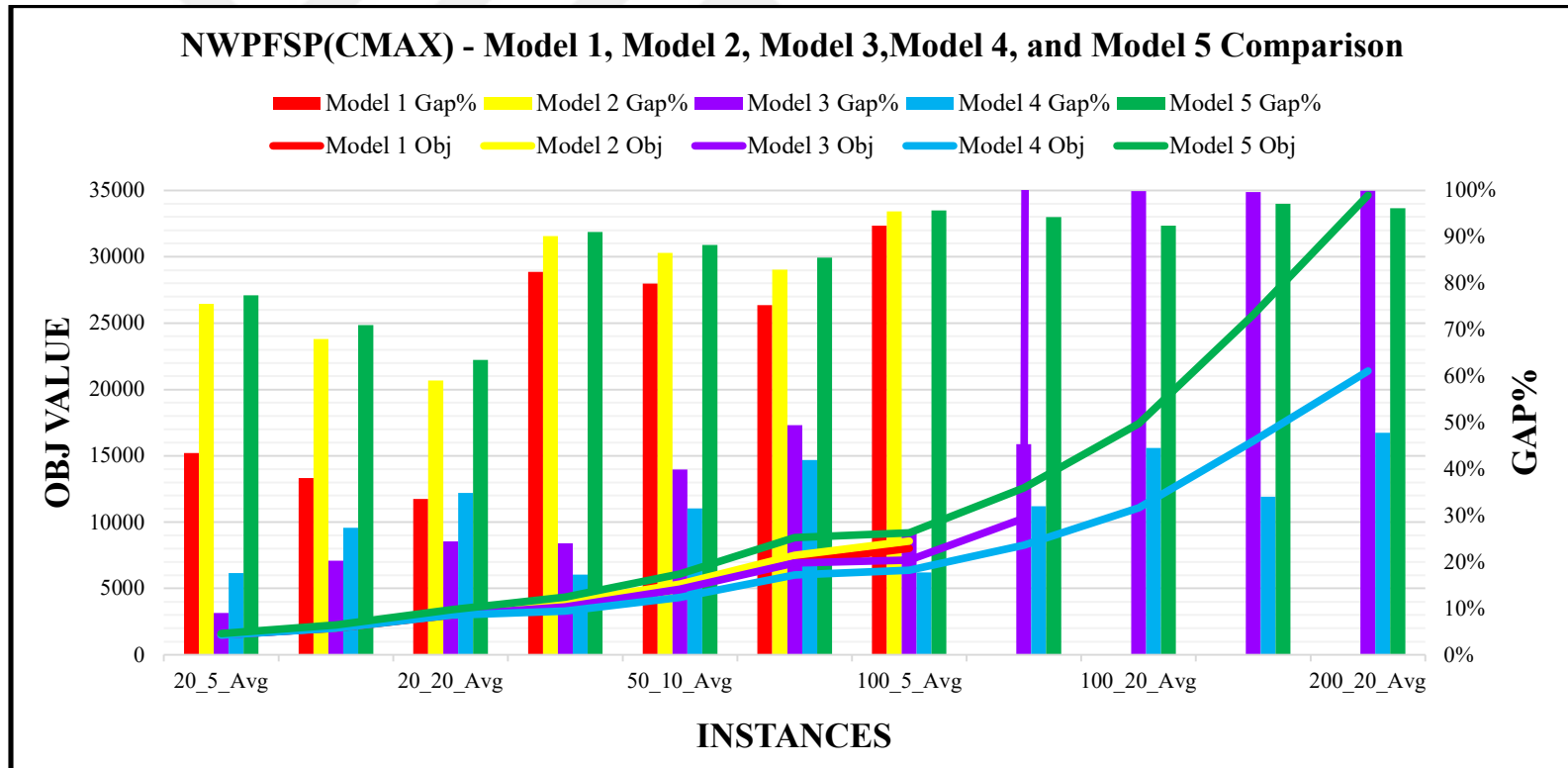


Figure 3.2. Comparison of Model 1, Model 2, Model 3, Model 4, and Model 5 for Makespan on Taillard Instances

Source: Yuksel et al. (2022a)

Referring to Table 3.3, all mathematical models find the optimal solutions in the first three sets of instances. All mathematical models can find a solution for up to 100x5 instances in makespan minimization. However, Model 1 and Model 2 cannot find any solution within the 1-hour time limit on 100x10, 100x20, 200x10, and 200x20 instances. However, we overcame this situation with Model 3, Model 4, and Model 5. They can find a feasible solution for all instances within a 1-hour time limit. The final inference from this analysis is that Model 4 performs the best among all models regarding the magnitude of the objective function with the value of 7579.12 and the gap percentage reached in 3600 seconds with a value of 31.54% in the overall analysis. Furthermore, to visualize the set-based differences among mathematical models reported in Table 3.3, a figure indicating the objective values as line charts and the gap values as bar charts are provided below, as in Figure 3.2. A conclusion can also be seen from Figure 3.2, that the objective function line of Model 4 is always the best, and the gap percentage bars of Model 4 are primarily the lowest except for the first three sets of instances.

Table 3.4. Computational Results of the Model 1, Model 2, Model 3, Model 4, and Model 5 for Total Flow Time on Taillard Instances

	Model 1		Model 2		Model 3		Model 4		Model 5	
	Objective Value	Gap %	Objective Value	Gap %	Objective Value	Gap %	Objective Value	Gap %	Objective Value	Gap %
5-5 Avg	1952.10	0.00%	1952.10	0.00%	1952.10	0.00%	1952.10	0.00%	1952.10	0.00%
5-10 Avg	3171.70	0.00%	3171.70	0.00%	3171.70	0.00%	3171.70	0.00%	3171.70	0.00%
5-20 Avg	5754.50	0.00%	5754.50	0.00%	5754.50	0.00%	5754.50	0.00%	5754.50	0.00%
20-5 Avg	16169.60	48.26%	16235.40	66.67%	16127.40	3.83%	16103.80	25.35%	16602.80	69.16%
20-10 Avg	23873.80	41.40%	23804.10	56.94%	23765.20	11.93%	23636.90	30.80%	24591.70	59.81%
20-20 Avg	38717.50	34.38%	38800.80	47.21%	38937.40	15.11%	38493.50	30.20%	41613.70	52.12%
50-5 Avg	91507.50	81.86%	97731.10	87.17%	85280.80	22.08%	81093.70	31.16%	100830.30	87.59%
50-10 Avg	126194.90	75.33%	139010.20	82.04%	126887.90	35.99%	114551.90	40.34%	163909.00	84.80%
50-20 Avg	183150.10	68.61%	200201.90	75.16%	189986.30	43.38%	168357.70	44.65%	243064.50	79.54%
100-5 Avg	369081.00	90.38%	422940.60	94.08%	337592.90	30.09%	302258.20	35.37%	475555.20	94.74%
100-10 Avg	no solution	-	no solution	-	494295.80	45.01%	416944.10	45.51%	655319.60	92.34%
100-20 Avg	no solution	-	no solution	-	734679.80	55.14%	593886.50	52.07%	902103.10	88.93%
200-10 Avg	no solution	-	no solution	-	1803486105.5	99.86%	1622338.3	50.73%	2598193.5	96.15%
200-20 Avg	no solution	-	no solution	-	no solution	-	2227640.2	59.25%	3529206.2	94.34%
Overall Avg.	121242.06	62.89%	134103.4	72.75%	180553365.9	36.24%	509573.16	40.49%	795544.51	81.77%

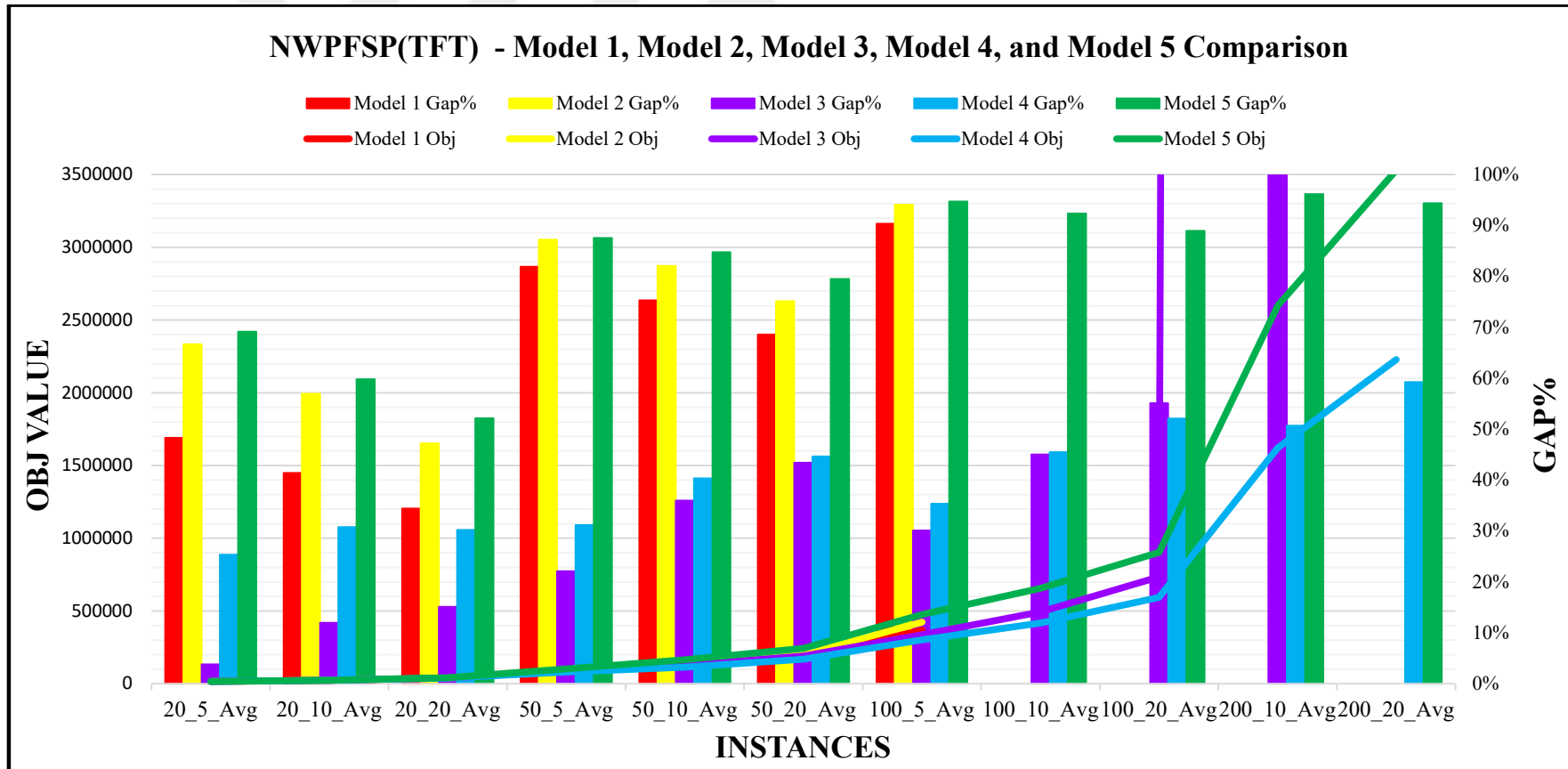


Figure 3.3. Comparison of Model 1, Model 2, Model 3, Model 4, and Model 5 for Total Flow Time on Taillard Instances

Source: Yuksel et al. (2022a)

Referring to Table 3.4, all mathematical models find the optimal solutions in the first three sets of instances. All mathematical models can reach a solution of up to 100x5 instances for total flow time minimization. However, Model 1 and Model 2 cannot find any solution within the 1-hour time limit on 100x10, 100x20, 200x10, and 200x20 instances. On the other hand, we overcome this situation with Model 3, Model 4, and Model 5. They can find a feasible solution for all instances within a 1-hour time limit. Hence, the last conclusion from this analysis is that Model 4 performs the best among all models according to the size of the objective function with the value of 509573.16; however, Model 3 performs the best among all in terms of the gap percentage found in 3600 seconds with the value 36.24% in the overall analysis.

Moreover, to better understand the model-wise differences demonstrated by Table 3.4 from a holistic point of view, a figure indicating the objective values as line charts and the gap values as bar charts are provided below, as in Figure 3.3. A conclusion can also be seen in Figure 3.3; that is, the objective function line of Model 4 is always the lowest, although the gap values of Model 3 are sometimes better than Model 4. Since Model 3 reaches a worse feasible solution in 200x10 instances and cannot find a feasible solution in 200x20 instances, Model 4 performs better in the overall average statistics.

Finally, it is also significant to mention that Model 3 can find optimal solutions for total flow time minimization for three instances in the set 20x5. Therefore, the average run time for this set is 2894.0 seconds.

Table 3.5. Computational Results of the Model 1, Model 2, Model 3, Model 4, and Model 5 for Total Tardiness on Taillard Instances

	Model 1		Model 2		Model 3		Model 4		Model 5	
	Objective Value	Gap %	Objective Value	Gap %	Objective Value	Gap %	Objective Value	Gap %	Objective Value	Gap %
5-5 Avg	956.10	0.00%	956.10	0.00%	956.10	0.00%	956.10	0.00%	956.10	0.00%
5-10 Avg	237.60	0.00%	237.60	0.00%	237.60	0.00%	237.60	0.00%	237.60	0.00%
5-20 Avg	263.20	0.00%	263.20	0.00%	263.20	0.00%	263.20	0.00%	263.20	0.00%
20-5 Avg	4512.50	80.49%	4696.20	99.87%	4463.80	13.62%	4414.20	91.61%	4803.00	100.00%
20-10 Avg	1805.10	48.92%	2299.60	100.00%	1903.40	93.37%	1795.20	98.23%	2488.60	100.00%
20-20 Avg	815.70	0.00%	960.60	59.85%	878.50	56.86%	824.50	9.56%	1802.50	90.00%
50-5 Avg	59010.90	99.11%	69242.40	100.00%	56501.60	36.25%	51276.00	51.92%	64374.50	100.00%
50-10 Avg	66065.20	99.24%	82970.20	100.00%	67914.30	68.71%	55571.00	88.46%	101566.00	100.00%
50-20 Avg	69870.40	99.17%	93325.70	100.00%	77582.30	100.00%	56625.00	99.62%	134755.70	100.00%
100-5 Avg	300853.70	99.67%	380345.60	100.00%	300672.40	41.97%	240911.80	44.93%	407844.00	100.00%
100-10 Avg	no solution	-	no solution	-	383234.30	61.39%	294361.70	65.99%	527868.20	100.00%
100-20 Avg	no solution	-	no solution	-	591687.50	84.70%	357854.70	90.80%	684221.60	100.00%
200-10 Avg	no solution	-	no solution	-	2343174.00	68.32%	1387844.10	60.52%	2395415.0	100.00%
200-20 Avg	no solution	-	no solution	-	3070510.50	79.60%	1780280.10	76.82%	no solution	-
Overall Avg.	71847.64	75.23%	90548.61	94.25%	627138.42	64.07%	384705.30	70.77%	432513.91	99.00%

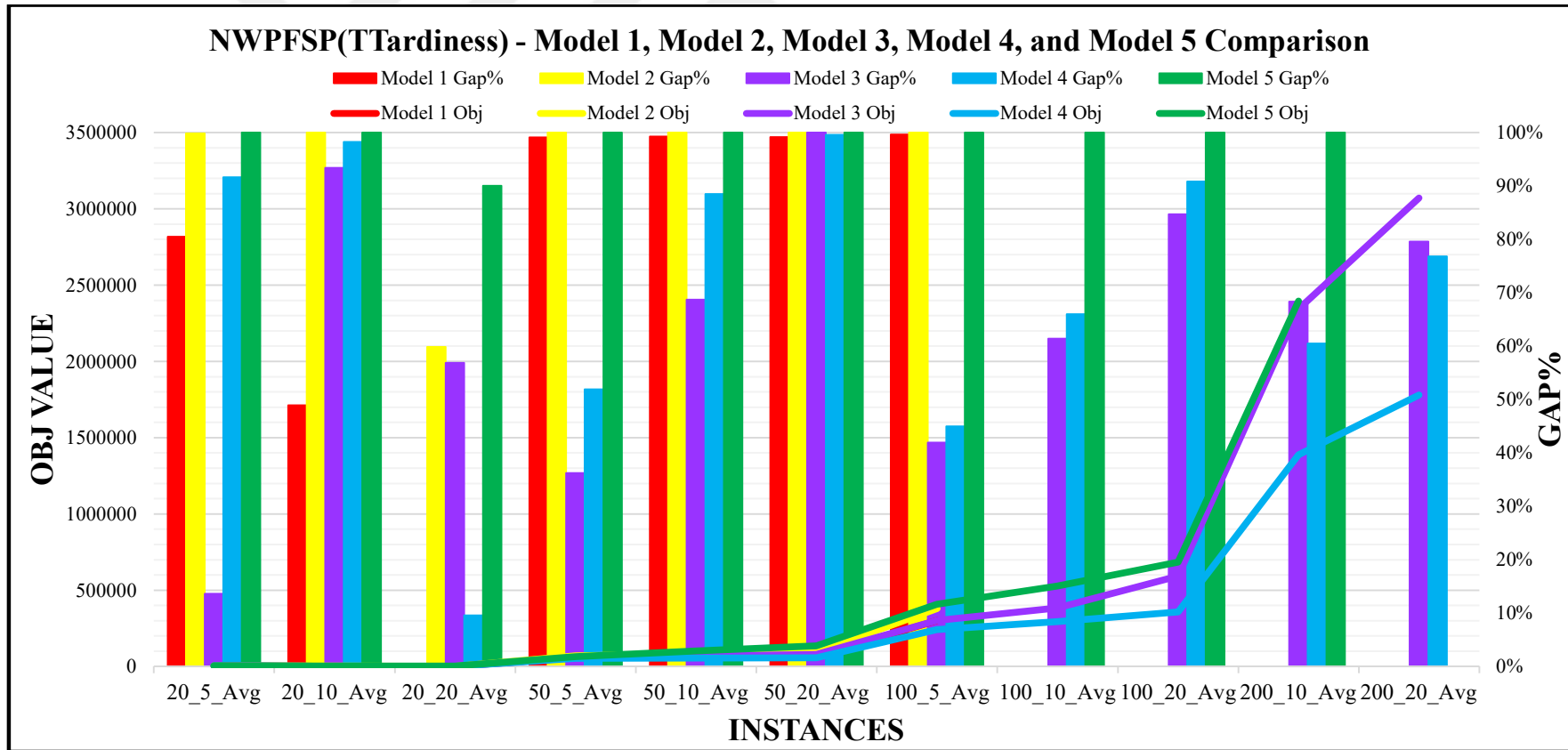


Figure 3.4. Comparison of Model 1, Model 2, Model 3, Model 4, and Model 5 for Total Tardiness on Taillard Instances

Source: Yuksel et al. (2022a)

Referring to Table 3.5, all mathematical models find the optimal solutions in the first three sets of instances. All mathematical models can solve up to 100x5 instances for total tardiness minimization. However, Model 1 and Model 2 cannot find any solution within the 1-hour time limit on 100x10, 100x20, 200x10, and 200x20 instances. On the other hand, we overcome this situation with Model 3, Model 4, and Model 5. They can find a feasible solution for all instances within a 1-hour time limit. Hence, the last conclusion from this analysis is that Model 4 performs the best among all models in terms of the objective function's magnitude with the value of 384705.30; however, Model 3 performs the best among all in terms of the gap percentage found in 3600 seconds with the value of 64.07% in the overall analysis.

In addition, Figure 3.4 is also generated to illustrate the data provided in Table 3.5. A conclusion can also be seen in Figure 3.4; that is, the objective function line of Model 4 is always the lowest, although the gap values of Model 3 are sometimes better than Model 4. Finally, it is also significant to point out that for total tardiness minimization,

- Model 1 can find optimal solutions for 2 and 10 instances in the sets 20x10 and 20x20, respectively. Therefore, the average run times for these sets are 3348.7 and 76.84 seconds, respectively.
- Model 2 can find optimal solutions for 4 instances in the set 20x20. So, the average run time for this set is 2718.2 seconds.
- Model 3 can find optimal solutions for 4 and 4 instances in the sets 20x5 and 20x20, respectively. Thus, the average run times for these sets are 2984.3 and 2199.4 seconds, respectively.
- Model 4 can find optimal solutions for 9 instances in the set 20x20; hence, the average run time for this set is 593.6 seconds.
- Model 5 can only find 1 optimal solution for total tardiness minimization; however, the time limit is consumed fully in this instance.

Regarding the VRF benchmark set:

All the details about the instance-based results in the VRF dataset can be found in the following link: <https://data.mendeley.com/datasets/xw33srdpxb/1>. Set-based averages of objective functions and gap percentages are reported in Tables 3.6, 3.7, and 3.8 for makespan, total flow time, and total tardiness. The last row of tables represents the overall averages to attain an ultimate overview of the mathematical models.

Table 3.6. Computational Results of the Model 1, Model 2, Model 3, Model 4, and Model 5 for Makespan on VRF Instances

	Model 1		Model 2		Model 3		Model 4		Model 5	
	Objective Value	Gap %	Objective Value	Gap %	Objective Value	Gap %	Objective Value	Gap %	Objective Value	Gap %
10-5 Avg	793.60	0.00%	793.60	0.00%	793.60	0.00%	793.60	0.00%	793.60	0.00%
10-10 Avg	1241.10	0.00%	1241.10	0.00%	1241.10	0.00%	1241.10	0.00%	1241.10	0.00%
10-15 Avg	1629.50	0.00%	1629.50	0.00%	1629.50	0.00%	1629.50	0.00%	1629.50	0.00%
10-20 Avg	1962.00	0.00%	1962.00	0.00%	1962.00	0.00%	1962.00	0.00%	1962.00	0.00%
20-5 Avg	1460.30	41.77%	1464.60	74.23%	1451.10	6.63%	1449.00	17.93%	1455.50	74.36%
20-10 Avg	2031.50	36.20%	2036.90	66.07%	2027.80	13.85%	2012.70	29.76%	2036.20	66.28%
20-15 Avg	2546.90	33.66%	2553.00	61.58%	2547.60	18.23%	2525.70	34.55%	2566.10	62.02%
20-20 Avg	3018.50	29.25%	3019.30	56.72%	3010.10	20.39%	2980.50	36.97%	3101.90	57.96%
30-5 Avg	2195.50	67.77%	2181.20	83.25%	2092.40	16.40%	2053.80	16.92%	2103.20	82.67%
30-10 Avg	2932.50	62.39%	2918.90	76.73%	2821.90	29.80%	2744.80	31.81%	2915.50	76.71%
30-15 Avg	3648.30	60.12%	3660.80	73.34%	3526.90	35.88%	3411.60	37.88%	3638.80	73.16%
30-20 Avg	4241.50	56.95%	4281.00	70.00%	4151.30	37.35%	4007.80	40.57%	4609.60	72.08%
40-5 Avg	3059.30	78.24%	3122.40	87.51%	2886.20	20.88%	2754.60	17.52%	2864.80	86.41%
40-10 Avg	4022.30	74.67%	4108.50	83.29%	3829.40	36.98%	3532.90	32.84%	3812.20	81.98%
40-15 Avg	4831.80	71.63%	4955.70	80.35%	4728.40	42.43%	4311.80	38.34%	4902.80	80.04%
40-20 Avg	5548.10	68.55%	5815.70	77.78%	5516.50	46.76%	4993.00	42.85%	6245.80	79.30%
50-5 Avg	3816.70	82.60%	3984.80	90.37%	3562.20	21.51%	3343.00	16.57%	3538.40	89.15%
50-10 Avg	4933.40	79.48%	5367.00	87.15%	4815.80	38.18%	4322.00	31.47%	4832.60	85.72%
50-15 Avg	6041.40	77.63%	6325.80	84.44%	5941.70	46.45%	5241.00	39.43%	6527.10	84.82%
50-20 Avg	6975.00	75.20%	7531.20	82.49%	6875.50	50.61%	6060.20	44.26%	8210.30	83.88%
60-5 Avg	4599.80	86.45%	4964.60	92.14%	4319.00	24.21%	3964.60	17.69%	4264.40	90.86%
60-10 Avg	6176.50	83.12%	6637.00	89.00%	5897.20	40.67%	5160.40	32.38%	6018.70	87.88%
60-15 Avg	7258.00	81.29%	7736.70	87.12%	6993.70	47.79%	6143.50	40.62%	8637.50	88.45%
60-20 Avg	8382.50	79.16%	9337.40	85.73%	8222.40	52.63%	7068.20	44.68%	9918.10	86.55%
Overall Avg.	3889.42	55.26%	4067.86	66.22%	3785.14	26.98%	3487.80	26.88%	4086.07	66.26%

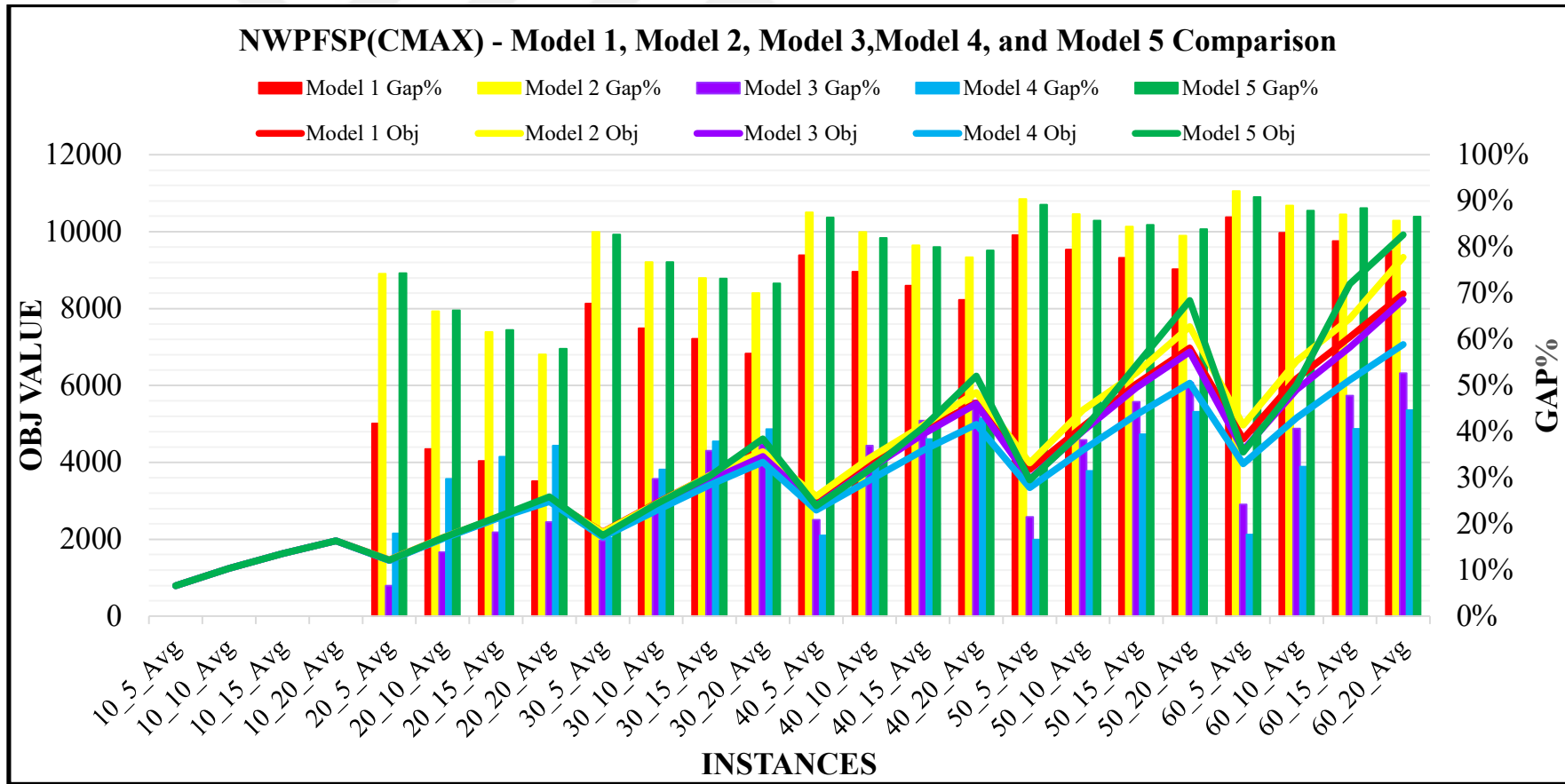


Figure 3.5. Comparison of Model 1, Model 2, Model 3, Model 4, and Model 5 for Makespan on VRF Instances

Source: Author

Referring to Table 3.6, all mathematical models find optimal solutions in the first four sets of instances, and each model can find a feasible solution for all instances within a one-hour time limit. The analysis concludes that Model 4 performs the best in terms of the objective function, achieving a value of 3487.80. However, Models 3 and 4 are competitive regarding the gap percentage reached in 3600 seconds, with 26.98% and 26.88%, respectively.

To visualize the set-based differences among the mathematical models reported in Table 3.6, Figure 3.5 presents the objective values as line charts and the gap values as bar charts. Figure 3.5 shows that Model 4 consistently achieves the best objective function values, and the gap percentage bars of Models 3 and 4 are generally the lowest. The figure shows a trend where the objective value increases within each set of job numbers when the number of machines increases. When moving to the next set of job numbers, the objective value drops and then continues to increase again, like a zigzag and increasing trend. This pattern is due to the NWPFS's nature and the jobs' processing time values in the VRF benchmark set.

Table 3.7. Computational Results of the Model 1, Model 2, Model 3, Model 4, and Model 5 for Total Flow Time on VRF Instances

	Model 1		Model 2		Model 3		Model 4		Model 5	
	Objective Value	Gap %	Objective Value	Gap %	Objective Value	Gap %	Objective Value	Gap %	Objective Value	Gap %
10-5 Avg	4506.30	0.00%	4506.30	0.00%	4506.30	0.00%	4506.30	0.00%	4506.30	0.00%
10-10 Avg	7790.40	0.00%	7790.40	0.00%	7790.40	0.00%	7790.40	0.00%	7790.40	0.00%
10-15 Avg	10971.00	0.00%	10971.00	0.00%	10971.00	0.00%	10971.00	0.00%	10971.00	0.00%
10-20 Avg	14119.40	0.00%	14119.40	0.00%	14119.40	0.00%	14119.40	0.00%	14119.40	0.00%
20-5 Avg	15655.50	46.64%	15658.10	65.10%	15590.30	0.08%	15591.40	24.62%	15674.40	67.67%
20-10 Avg	23824.30	40.90%	23706.30	55.61%	23751.40	7.09%	23607.80	28.90%	23782.40	58.01%
20-15 Avg	31502.70	36.43%	31426.90	50.53%	31458.70	10.79%	31344.60	31.34%	31773.60	53.11%
20-20 Avg	38479.90	33.39%	38119.80	46.09%	38307.50	12.30%	38149.70	30.78%	38698.50	48.82%
30-5 Avg	33030.80	68.26%	33470.10	77.26%	32043.50	12.36%	31806.90	29.46%	32870.40	77.35%
30-10 Avg	47574.00	60.31%	48159.50	68.94%	46637.80	22.38%	45807.50	35.42%	48117.00	69.48%
30-15 Avg	63332.90	57.32%	62410.40	64.06%	61078.70	26.73%	59789.90	37.77%	63852.00	65.40%
30-20 Avg	75164.20	53.46%	75289.10	60.16%	74145.10	27.77%	71902.40	37.10%	76193.70	61.18%
40-5 Avg	59449.00	76.65%	62439.60	83.49%	55805.80	16.81%	54543.90	28.05%	59140.80	82.64%
40-10 Avg	82667.20	70.08%	85585.20	76.96%	79438.90	29.38%	76299.70	38.62%	83309.00	76.40%
40-15 Avg	104194.40	65.84%	108277.50	72.14%	102104.10	32.65%	98131.00	41.45%	123456.40	75.63%
40-20 Avg	122219.80	62.62%	127772.20	68.95%	118578.90	34.22%	114562.50	41.24%	138967.20	71.53%
50-5 Avg	91474.90	81.11%	102420.90	87.68%	85597.20	20.30%	82396.10	30.19%	90389.20	86.07%
50-10 Avg	126653.80	73.25%	140898.40	82.32%	121609.40	32.11%	114908.00	39.10%	135957.30	81.69%
50-15 Avg	158790.70	72.43%	172503.70	78.11%	155117.40	38.38%	144519.50	43.80%	197070.90	80.89%
50-20 Avg	185714.70	68.95%	202540.50	75.34%	184247.80	41.77%	169314.50	45.56%	233882.70	78.67%
60-5 Avg	131877.70	84.54%	149507.50	89.90%	122354.90	23.47%	115123.80	31.47%	141690.20	89.33%
60-10 Avg	183879.30	80.24%	210048.90	85.65%	177940.00	37.29%	161155.00	42.12%	218029.60	86.15%
60-15 Avg	225314.90	76.63%	251470.30	82.12%	223586.80	43.22%	199225.10	46.02%	291128.30	84.55%
60-20 Avg	258129.50	73.48%	296582.10	79.76%	264239.30	46.32%	232986.10	47.33%	346753.20	82.72%
Overall Avg.	87346.55	53.44%	94819.75	60.42%	85459.19	21.48%	79939.69	30.43%	101171.83	61.55%

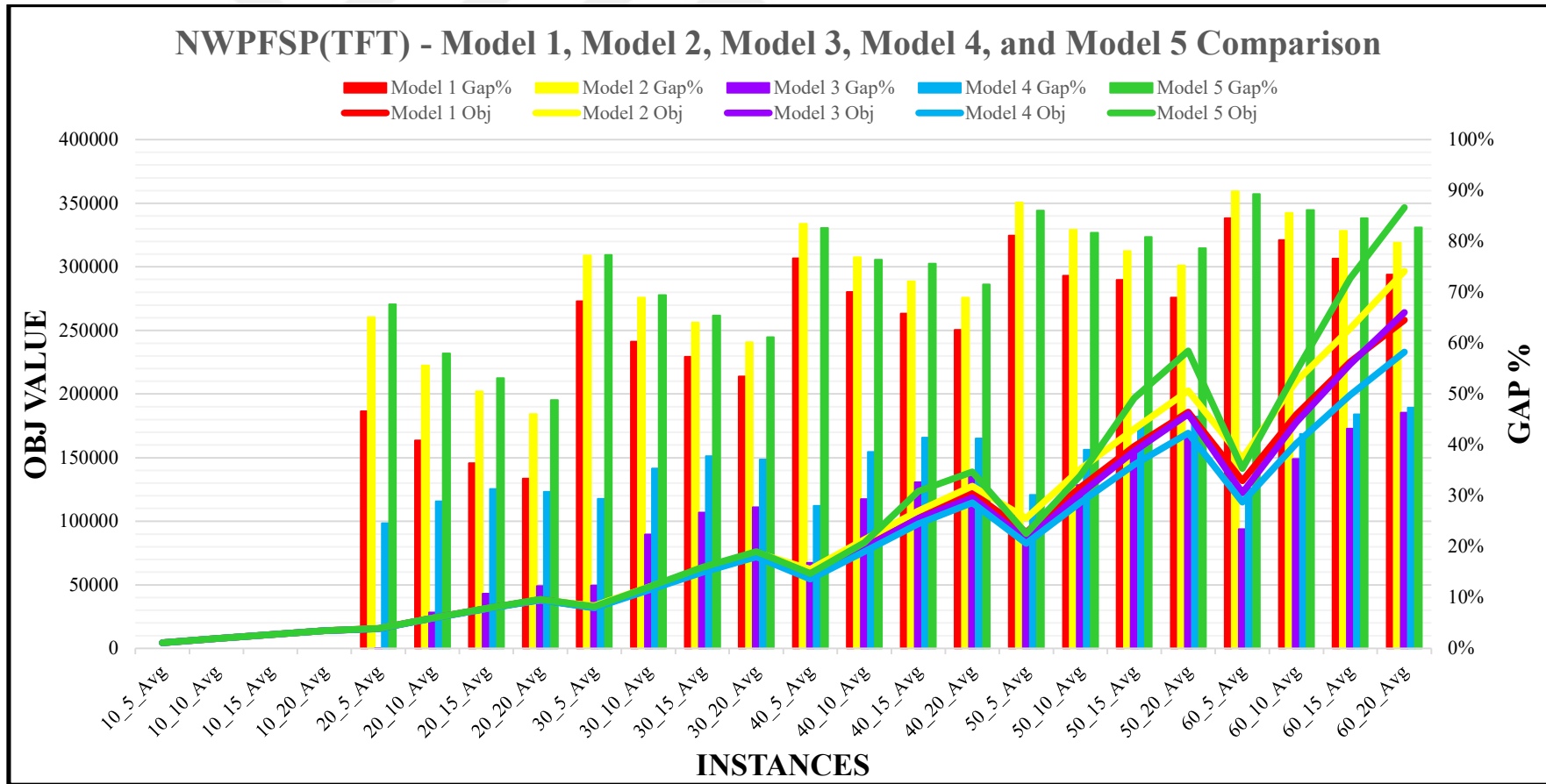


Figure 3.6. Comparison of Model 1, Model 2, Model 3, Model 4, and Model 5 for Total Flow Time on VRF Instances

Source: Author

According to Table 3.7, all mathematical models achieve optimal solutions in the initial four sets of instances. Each model can find a feasible solution for all instances within one 1-hour. The analysis reveals that Model 4 excels in terms of the objective function, reaching a value of 79939.69. However, Model 3 outperforms the others in terms of the gap percentage within 3600 seconds, with a value of 21.48%.

Furthermore, Figure 3.6 provides a comprehensive view of the model-wise differences in Table 3.7 by displaying the objective values as line charts and the gap values as bar charts. Figure 3.6 reveals that the objective function line for Model 4 consistently shows the lowest values, while the gap percentage bars for Model 3 are always the lowest.

Finally, it is also significant to mention that Model 3 can find optimal solutions for total flow time minimization for nine instances in the set 20x5. Therefore, the average run time for this set is 1005.3 seconds. The zigzag and increasing trend observed in the objective value for makespan is also evident in the total flow time.

Table 3.8. Computational Results of the Model 1, Model 2, Model 3, Model 4, and Model 5 for Total Tardiness on VRF Instances

	Model 1		Model 2		Model 3		Model 4		Model 5	
	Objective Value	Gap %	Objective Value	Gap %	Objective Value	Gap %	Objective Value	Gap %	Objective Value	Gap %
10-5 Avg	2016.10	0.00%	2016.10	0.00%	2016.10	0.00%	2016.10	0.00%	2016.10	0.00%
10-10 Avg	3254.00	0.00%	3254.00	0.00%	3254.00	0.00%	3254.00	0.00%	3254.00	0.00%
10-15 Avg	4885.90	0.00%	4885.90	0.00%	4885.90	0.00%	4885.90	0.00%	4885.90	0.00%
10-20 Avg	6376.80	0.00%	6376.80	0.00%	6376.80	0.00%	6376.80	0.00%	6376.80	0.00%
20-5 Avg	6225.50	65.42%	6333.70	9.88%	6100.20	0.49%	6113.10	81.31%	6153.80	95.51%
20-10 Avg	9618.00	60.60%	9681.20	90.43%	9561.60	20.79%	9458.30	83.46%	9585.70	91.37%
20-15 Avg	12949.20	61.53%	13155.70	89.09%	13044.50	32.01%	12756.70	84.29%	13128.10	90.23%
20-20 Avg	14799.10	56.41%	14952.20	86.28%	14923.20	34.58%	14693.90	82.98%	15316.00	87.84%
30-5 Avg	13230.30	90.93%	15057.10	98.21%	12756.50	40.12%	12163.10	89.59%	13232.90	98.00%
30-10 Avg	19330.40	87.95%	21087.50	96.62%	18481.40	63.80%	17303.00	93.31%	19470.10	96.30%
30-15 Avg	25165.70	85.91%	26646.00	94.69%	24311.90	76.03%	22585.30	91.73%	26075.80	94.63%
30-20 Avg	27313.70	83.95%	29738.90	94.13%	27300.60	78.47%	25299.30	91.28%	32470.90	94.63%
40-5 Avg	23588.00	95.71%	31867.20	99.01%	21954.30	54.93%	20346.50	94.30%	23068.00	98.63%
40-10 Avg	31620.90	95.12%	41638.70	98.40%	28705.60	86.52%	26202.70	96.11%	32740.50	97.96%
40-15 Avg	41598.70	93.47%	51559.30	97.47%	39800.90	94.34%	34648.10	94.91%	56590.40	97.67%
40-20 Avg	47437.50	91.97%	58320.90	96.62%	47508.30	97.57%	39738.70	93.88%	64567.00	96.91%
50-5 Avg	36807.20	97.81%	52979.20	99.42%	34312.90	64.40%	29096.20	96.66%	36732.40	99.18%
50-10 Avg	49647.60	96.68%	74119.70	98.95%	49658.20	92.41%	40115.90	97.08%	65614.30	98.82%
50-15 Avg	63012.10	96.22%	86386.00	98.62%	62414.70	99.29%	50259.50	96.73%	104569.90	98.87%
50-20 Avg	77987.10	95.11%	10084.60	97.92%	75999.30	99.74%	57996.70	95.61%	128588.40	98.37%
60-5 Avg	53601.20	98.77%	83240.80	99.66%	49532.30	70.73%	40486.10	96.92%	60047.50	99.53%
60-10 Avg	73214.60	98.09%	115591.00	99.34%	71014.00	99.27%	53840.90	97.81%	124229.20	99.39%
60-15 Avg	88619.20	97.50%	128794.30	99.05%	96364.10	99.90%	63499.20	97.45%	147868.60	99.18%
60-20 Avg	104069.60	96.99%	155567.00	98.68%	136155.20	99.90%	73972.80	96.64%	190811.70	98.93%
Overall Avg.	34848.68	72.76%	47253.99	80.31%	35684.69	58.55%	27796.20	77.17%	49474.75	80.50%

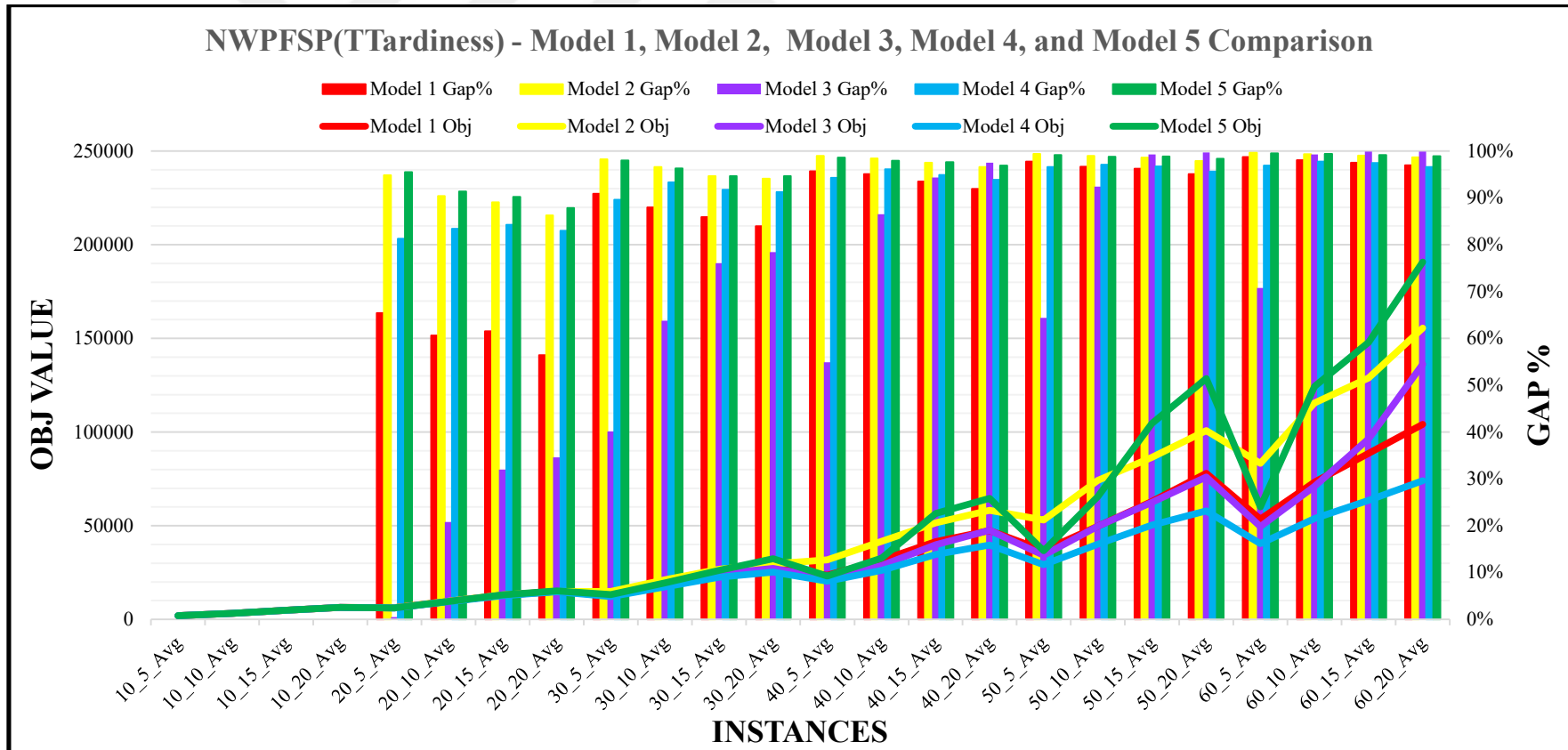


Figure 3.7. Comparison of Model 1, Model 2, Model 3, Model 4, and Model 5 for Total Tardiness on VRF Instances

Source: Author

According to Table 3.8, all mathematical models achieve optimal solutions in the initial four sets of instances, and each model can find a feasible solution for all instances within a 1-hour time frame. The analysis indicates that Model 4 excels in terms of the objective function, achieving a value of 27796.20. However, Model 3 surpasses the others in gap percentage within 3600 seconds, with a value of 58.55%.

Furthermore, Figure 3.7 offers a comprehensive view of the model-wise differences illustrated in Table 3.8 by presenting the objective values as line charts and the gap values as bar charts. The figure shows that Model 4's objective function consistently has the lowest values. Although Model 3's gap percentage bars are the lowest, the gap values of Model 4 are sometimes better than Model 3.

Finally, it is also significant to mention that Model 3 can find optimal solutions for total tardiness minimization for nine instances in the set 20x5. Therefore, the average run time for this set is 1632.50 seconds. The zigzag and increasing trend observed in the objective value for makespan and total flow time is also evident in the total flow tardiness.

Sections 3.2 and 3.3 address Research Question 1 of this thesis.

Research Question 1: Can we generate more effective *mathematical models* (MILP and CP model formulations) for the $(F_m|nwt|ServiceLevel)$ problems?

These sections discuss the $(F_m|nwt|C_{max})$, $(F_m|nwt|\sum C_i)$, and $(F_m|nwt|\sum T_i)$, problems, addressing Research Question 1 as follows. Five **mathematical models** have been introduced to the NWPFSPs literature in Sections 3.2 and 3.3. The computational results are represented for $(F_m|nwt|C_{max})$, $(F_m|nwt|\sum C_i)$, $(F_m|nwt|\sum T_i)$ problem in the **Taillard** and **VRF** datasets. Model 3 and Model 4 are the top performers overall, showing superior results in both the objective function and percentage gap for makespan, total flow time, and total tardiness. They can be effectively used in flowshops to obtain high-quality feasible solutions within a 1-hour time frame.

As the research output from this section, I had the opportunity to present the results of the **Taillard** benchmark set from these sections (Sections 3.2 and 3.3) at the International Symposium for Production Research 2022 (ISPR 2022), which occurred in a hybrid format from October 6th to 8st, 2022, in Antalya, Türkiye. ISPR 2022

proceedings are published as a book by Springer under "Towards Industry 5.0" and indexed in SCOPUS. The paper entitled "*Mathematical Models for No-Wait Permutation Flowshop Scheduling Problems*" is accessible online at the following link (Yüksel et al. (2022a)): https://link.springer.com/chapter/10.1007/978-3-031-24457-5_44.

3.4. Empowering the Mathematical Model Formulations: Valid Inequalities

Owing to the NP-hardness identity of the problem (Röck, H. (1984)), the mathematical models developed to solve this problem cannot reach optimal solutions for large instances in polynomial time. However, the quality of the objective functions and the gap values obtained by the mathematical models in a specific time window can be improved by valid inequalities. The aim of generating valid inequalities over the mathematical models is to enhance the outcomes of mathematical models, i.e., the objective function and the gap values.

In the literature, the valid inequalities are widely utilized in the mathematical models for various combinatorial optimization problems such as discrete lot-sizing and scheduling (Gicquel & Minoux (2015); Kaczmarczyk, W. (2020)), production scheduling (chemical) (Merchan & Maravelias (2016)), railway traffic management (Pellegrini et al. (2019)), project scheduling (Kis, T. (2005)). In the field of flowshop scheduling problems, the mathematical models aiming to optimize unrelated parallel machine scheduling (Hassan et al. (2016); Saberi-Aliabad et al. (2020)); two-machine permutation flowshop scheduling (Hamdi & Toumi (2019)), single machine scheduling (Dauzère-Pérès, S. (1997); Keha et al. (2009)) with sequence and time-dependent variants (Stecco et al. (2008)), continuous energy-constraint scheduling (Nattaf et al. (2019)) problems are enhanced by some valid inequalities. Among these studies, Gicquel et al. (2015), Kis, T. (2005), and Stecco et al. (2008) have implemented their valid inequalities into their branch and cut algorithms. Regarding NWPFSPs, a set of valid inequalities as lower and upper bounds are generated for makespan minimization by Yüksel, D. (2019). Additionally, a certain number of valid inequalities are produced to strengthen the developed mathematical models for two-machine NWPFSP (Labidi et al. (2018)) and two-stage hybrid NWPFSP (Azaiez et al. (2021)) problems.

This section proposes new valid inequalities to improve a mathematical model's performance, optimizing the NWPFSF. The performance of valid inequalities is tested for three significant objective functions: makespan (C_{max}), total flow time ($\sum C_i$), total tardiness ($\sum T_i$). Then, the computational outcomes are reported in Section 3.5.

New intelligent valid inequalities are proposed for the Model 1 – MILP Model formulation explained in Section 3.2. This mathematical model is constituted for NWPFSF by Yüksel, D. (2019) as it is adapted from Manne's good-old PFSF formulation (Manne, A. S. (1960)). The computational results for makespan, total flow time, and tardiness minimization were reported separately in Yüksel, D. (2019). Since the computational results are good enough, this mathematical model is also studied for energy-efficient scheduling of NWPFSFs in Yüksel et al. (2020a); Yüksel et al. (2020b); Tasgetiren et al. (2019)). Chvatal-Gomory (C-G) valid inequalities are applied to Equation (3.3) of the Model 1 – MILP Model. A reminder for Equation (3.3):

$$C_{ir} - C_{i,r-1} \geq P_{ir} \quad \forall i \in N, \forall r \in M: r \geq 2 \quad (3.3)$$

The following C-G valid inequalities are acquired.

$$C_{iM} - C_{i1} \geq \lfloor (1/\alpha) * \sum_{r \in \{2, \dots, R\}} [\alpha * P_{ir}] \rfloor \quad \forall i \in N \quad (3.46)$$

$$C_{ir} - C_{i1} \geq \lfloor (1/\alpha) * \sum_{t \in \{2, \dots, r\}} [\alpha * P_{it}] \rfloor \quad \forall i \in N, \forall r \in \{3, \dots, M\} \quad (3.47)$$

$$C_{iM} - C_{il} \geq \lfloor (1/\alpha) * \sum_{t \in \{l+1, \dots, M\}} [\alpha * P_{it}] \rfloor \quad \forall i \in N, \forall l \in \{1, \dots, M-2\} \quad (3.48)$$

$$C_{ir} - C_{il} \geq \lfloor (1/\alpha) * \sum_{t \in \{l+1, \dots, r\}} [\alpha * P_{it}] \rfloor \quad \forall i \in N, \forall r \in \{3, \dots, R\} \forall l \in \{1, \dots, R-2\} \mid l < r-1 \quad (3.49)$$

Equation (3.46) constructs a relationship between a job's finishing time on the first and the last machine. Equation (3.47) is the one-step-extended version of Equation (3.46) in which it considers the relationship between a job's finishing time on the first machine and all machines from 3 to $|M|$. Similarly, Equation (3.48) is another point of view to Equation (3.46) in which it considers the relationship between a job's completion time on the last machine and all machines from 1 to $|M-2|$. Finally, Equation (3.49) is the complete version of all combinations. It considers the relationship between a job's completion time on two different machines r and l such that l is always less than $r-1$.

3.5. Computational Analysis of Valid Inequalities

The valid inequalities are added to the MILP model as a new set of constraints. To decide the value of α in the formulation, an experiment is conducted considering all combinations of α values, which can be obtained from the finite prime numbers set, are $\{2,3,5,7,11,13,17,19\}$. The experiment is conducted over the truncated small instances of Taillard proposed in the literature by Yüksel et al. (2020b). There are 30 small instances of 10 instances in each set of 5 jobs and 5, 10, and 20 machines. According to the experiment results, the best α values yielding to solve the small instances fastest are 13/19, 7/11, and 2/17 for makespan, total flow time, and total tardiness objective functions, respectively. Therefore, these α values are carried out to measure the performance of valid inequalities in large instances. The mathematical model is executed by valid inequalities and corresponding α values on the *Taillard* dataset (Taillard, E. (1993)). The detailed computation results are provided on the website <https://data.mendeley.com/datasets/ck35y8gfxk/1>. Each instance is run for 3600 seconds. The objective values and the gap values of the model without valid inequalities, the objective values, and the gap values of the model with valid inequalities, the improvement in objective value, and the gap percentage are reported in Tables 3.9-3.14. Tables 3.9 and 3.10 represent the results for makespan with valid inequalities (3.46) and (3.49), respectively. Tables 3.11 and 3.12 depict the results for total flow time with valid inequalities (3.46) and (3.49), respectively. Tables 3.13-3.14 list the results for total tardiness with valid inequalities (3.46) and (3.49), respectively.

Table 3.9. Comparison of Results with Valid Inequality (3.46) for Makespan

Instance	Objective Value	Gap %	Objective Value with Valid Inequality (3.46)	Gap % with Valid Inequality (3.46)	Improvement in Objective Value	Improvement in Gap %
20x5_Avg	1508.3	43.47%	1500.0	39.66%	0.55%	8.76%
20x10_Avg	2019.4	38.07%	2004.4	35.51%	0.74%	6.73%
20x20_Avg	2996.0	33.55%	2989.7	30.68%	0.21%	8.56%
50x5_Avg	3814.3	82.46%	3773.2	82.30%	1.08%	0.19%
50x10_Avg	5037.2	79.96%	4982.7	79.76%	1.08%	0.25%
50x20_Avg	6980.1	75.32%	6889.1	75.26%	1.30%	0.08%
100x5_Avg	8024.9	92.36%	7894.6	91.68%	1.62%	0.73%

Table 3.10. Comparison of Results with Valid Inequality (3.49) for Makespan

Instance	Objective Value	Gap %	Objective Value with Valid Inequality (3.49)	Gap % with Valid Inequality (3.49)	Improvement in Objective Value	Improvement in Gap %
20x5_Avg	1508.3	43.47%	1500.6	37.99%	0.51%	12.60%
20x10_Avg	2019.4	38.07%	2007.1	34.21%	0.61%	10.14%
20x20_Avg	2996.0	33.55%	3000.7	30.08%	-0.16%	10.34%
50x5_Avg	3814.3	82.46%	3763.3	81.60%	1.34%	1.04%
50x10_Avg	5037.2	79.96%	4933.0	78.90%	2.07%	1.33%
50x20_Avg	6980.1	75.32%	6839.9	75.29%	2.01%	0.04%
100x5_Avg	8024.9	92.36%	7879.2	91.57%	1.82%	0.86%

Table 3.11. Comparison of Results with Valid Inequality (3.46) for Total Flow Time

Instance	Objective Value	Gap %	Objective Value with Valid Inequality (3.46)	Gap % with Valid Inequality (3.46)	Improvement in Objective Value	Improvement in Gap %
20x5_Avg	16169.6	48.26%	16230.8	45.42%	-0.38%	5.88%
20x10_Avg	23873.8	41.40%	23688.4	37.67%	0.78%	8.99%
20x20_Avg	38717.5	34.38%	38752.7	31.58%	-0.09%	8.15%
50x5_Avg	91507.5	81.86%	87902.8	81.39%	3.94%	0.58%
50x10_Avg	126194.9	75.33%	122682.1	74.92%	2.78%	0.55%
50x20_Avg	183150.1	68.61%	180125.1	68.34%	1.65%	0.39%
100x5_Avg	369081.0	90.38%	363814.8	90.40%	1.43%	-0.03%

Table 3.12. Comparison of Results with Valid Inequality (3.49) for Total Flow Time

Instance	Objective Value	Gap %	Objective Value with Valid Inequality (3.49)	Gap % with Valid Inequality (3.49)	Improvement in Objective Value	Improvement in Gap %
20x5_Avg	16169.6	48.26%	16247.5	45.60%	-0.48%	5.51%
20x10_Avg	23873.8	41.40%	23712.6	38.32%	0.68%	7.43%
20x20_Avg	38717.5	34.38%	38652.4	31.37%	0.17%	8.75%
50x5_Avg	91507.5	81.86%	87608.6	81.26%	4.26%	0.73%
50x10_Avg	126194.9	75.33%	123469.1	75.33%	2.16%	0.01%
50x20_Avg	183150.1	68.61%	180784.6	68.54%	1.29%	0.10%
100x5_Avg	369081.0	90.38%	369219.8	91.53%	-0.04%	-1.28%

Table 3.13. Comparison of Results with Valid Inequality (3.46) for Total Tardiness

Instance	Objective Value	Gap %	Objective Value with Valid Inequality (3.46)	Gap % with Valid Inequality (3.46)	Improvement in Objective Value	Improvement in Gap %
20x5_Avg	4512.5	80.49%	4484.9	76.19%	0.61%	5.35%
20x10_Avg	1805.1	48.92%	1787.8	37.71%	0.96%	22.91%
20x20_Avg	815.70	0.00%	815.7	0.00%	0.00%	0.00%
50x5_Avg	59010.9	99.11%	58983.8	98.96%	0.05%	0.16%
50x10_Avg	66065.2	99.24%	64352.6	99.09%	2.59%	0.16%
50x20_Avg	69870.4	99.17%	68033.8	98.94%	2.63%	0.23%
100x5_Avg	300853.7	99.67%	295578.4	99.74%	1.75%	-0.07%

Table 3.14. Comparison of Results with Valid Inequality (3.49) for Total Tardiness

Instance	Objective Value	Gap %	Objective Value with Valid Inequality (3.49)	Gap % with Valid Inequality (3.49)	Improvement in Objective Value	Improvement in Gap %
20x5_Avg	4512.5	80.49%	4461.6	75.89%	1.13%	5.71%
20x10_Avg	1805.1	48.92%	1772.1	33.74%	1.83%	31.02%
20x20_Avg	815.70	0.00%	815.7	0.00%	0.00%	0.00%
50x5_Avg	59010.9	99.11%	56492.6	98.92%	4.27%	0.20%
50x10_Avg	66065.2	99.24%	62879.4	99.07%	4.82%	0.18%
50x20_Avg	69870.4	99.17%	68508.1	99.03%	1.95%	0.15%
100x5_Avg	300853.7	99.67%	295574.3	99.75%	1.75%	-0.08%

According to Tables 3.9 and 3.10, valid inequality (3.49) improves gap percentage by 8.76%, 6.73%, and 8.56% in the first three sets of instances, whereas valid inequality (3.49) improve gap percentage by 12.60%, 10.14%, and 10.34% in the same set of

instances. This result illustrates that valid inequality (3.49) immensely improves the performance of makespan minimization.

According to Tables 3.11 and 3.12, valid inequality (3.46) achieves 5.88%, 8.99%, and 8.15% improvement in gap percentage in the first three sets of instances. In contrast, valid inequality (3.49) achieves 5.51%, 7.43%, and 8.75% improvement in gap percentage in the same set of instances.

According to Tables 3.13 and 3.14, 5.35% and 22.91% improvement in gap percentage are achieved in the first two sets of instances by valid inequality (3.46), whereas 5.71% and 31.02% improvement in gap percentage are achieved in the same set of instances by valid inequality (3.49). This outcome illustrates that the valid inequality (3.46) performance improves significantly by valid inequality (3.49) in the 20x10 set of instances in total tardiness minimization. In the 20x20 set of instances, all the instances are solved optimally. Thus, it seems like there is no improvement. However, it is essential to mention that there is also a time improvement. In the 20x20 set of instances, the average run time is 76.8 seconds, whereas it reduces to 12.1 and 11.2 seconds, including valid inequalities (3.46) and (3.49), respectively. Similarly, In the 20x10 set of instances, the average run time is 3348.7 seconds, whereas it reduces to 3119.2 and 2820.6 seconds by adding the valid inequality (3.46) and (3.49), respectively. The number of optimal solutions found in this set of instances increases from 2 to 3 and 4, including valid inequality (3.46) and (3.49), respectively.

Sections 3.4 and 3.5 address Research Question 2 of this thesis.

Research Question 2: Can we enhance the mathematical models developed for the $(F_m|nwt|ServiceLevel)$ problems by incorporating *valid inequalities*?

These sections discuss the $(F_m|nwt|C_{max})$, $(F_m|nwt|\sum C_i)$, and $(F_m|nwt|\sum T_i)$, problems, addressing Research Question 2 as follows. New **Chvatal-Gomory valid inequalities** have been introduced to enhance the performance of a mathematical model for NWPFSs in Sections 3.4 and 3.5. These inequalities were evaluated for $(F_m|nwt|C_{max})$, $(F_m|nwt|\sum C_i)$, and $(F_m|nwt|\sum T_i)$ problem in the **Taillard** dataset. The computational results indicate that the valid inequalities significantly improved the model's performance, particularly in the first three sets of instances, where a substantial reduction in the gap percentage was observed.

As the research output from this section, I had the opportunity to present the results of these sections (Section 3.4 and 3.5) at the International Conference on Intelligent and Fuzzy Systems 2022 (INFUS 2022), which occurred in a hybrid format from July 19th to 21st, 2022, in Izmir, Türkiye. INFUS 2022 proceedings are published as a book by Springer under “Lecture Notes in Networks and Systems” and are indexed in Scopus and Web of Science (Yüksel et al. (2022b)). The paper entitled “*Intelligent Valid Inequalities for No-Wait Permutation Flowshop Scheduling Problems*” is accessible online at the following link: https://link.springer.com/chapter/10.1007/978-3-031-09173-5_105.

3.6. Lower Bounds

It is a fact that the NWPFSPs problem is NP-hard (Röck, 1984). Therefore, the Lower Bound (LB) generation is another challenge while reaching optimal solutions. A Lower Bound (LB) generation scheme has been developed by taking inspiration from the Shortest Path (SP) calculation of a given directed graph. Two sound LBs are developed: LB_I and LB_I -Lifted.

LB_I :

Note that the sets for the jobs and the machines are given as follows: $N = \{1, \dots, n\}$ and $M = \{1, \dots, m\}$. In addition, P_{ir} is set as the processing time of job $i \in N$ on machine $r \in M$ along with the definition of the time delay between any consecutive job i and k on the last machine, which is set to $\Delta_{L_{ik}}$ (See Figure 3.1 (b)). Considering this information, let's assume that we are given a directed graph $G = (V, A)$, where $V = \{0\} \cup \{1, \dots, n\}$ and $A = \{(0, i): i \in N\} \cup \{(i, k): i, k \in N\}$ together with the following distances:

$$d_{0,i} = \sum_{r=1}^M P_{ir}, \quad \forall i \in N \quad (3.50)$$

$$d_{i,i} = d_{i,0} = \infty, \quad \forall i \in N \cup \{0\} \quad (3.51)$$

$$d_{i,j} = \Delta_{L_{ij}}, \quad \forall i, j \in N: i \neq j \quad (3.52)$$

The distance from the 0^{th} vertex to all other vertices (3.50) is the total processing time of job i . However, the reverse distances are set equal to infinity (3.51) and loops. Finally, the distance between the i^{th} vertex and the k^{th} vertex is set to $\Delta_{L_{ik}}$ (3.52).

Note that $\Delta_{L_{ik}}$ and $\Delta_{L_{ki}}$ are not equal. The directed graph with six vertices given in Figure 3.8. is a generic example of five jobs.

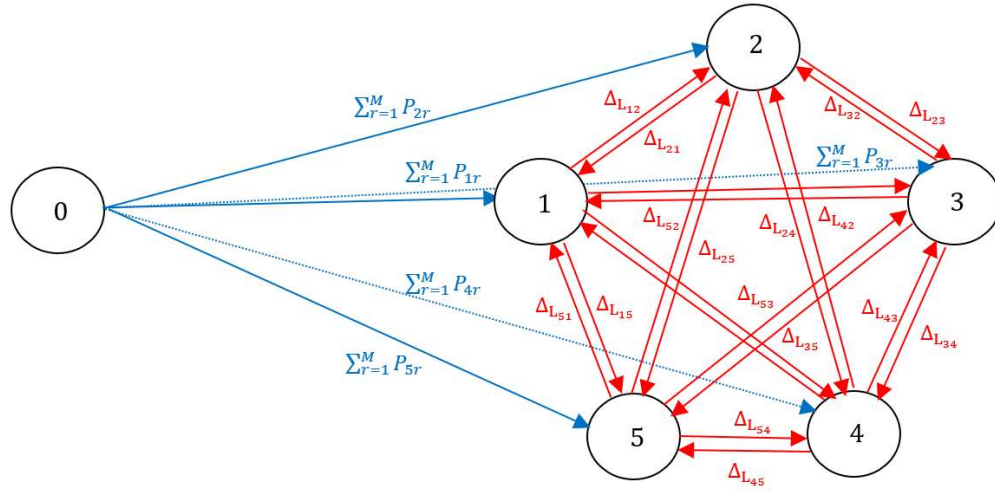


Figure 3.8. Directed graph of an NWPFSF

Regarding Figure 3.8., the distance matrix whose dimensions are $(n + 1)$ by $(n + 1)$ can be written as proposed in Table 3.15.

Table 3.15. The Distance matrix of the directed graph

Distances	0	1	2	3	4	5
0	∞	$d_{0,1} = \sum_{r=1}^M P_{1r}$	$d_{0,2} = \sum_{r=1}^M P_{2r}$	$d_{0,3} = \sum_{r=1}^M P_{3r}$	$d_{0,4} = \sum_{r=1}^M P_{4r}$	$d_{0,5} = \sum_{r=1}^M P_{5r}$
1	∞	∞	$\Delta_{L_{12}}$	$\Delta_{L_{13}}$	$\Delta_{L_{14}}$	$\Delta_{L_{15}}$
2	∞	$\Delta_{L_{21}}$	∞	$\Delta_{L_{23}}$	$\Delta_{L_{24}}$	$\Delta_{L_{25}}$
3	∞	$\Delta_{L_{31}}$	$\Delta_{L_{32}}$	∞	$\Delta_{L_{34}}$	$\Delta_{L_{35}}$
4	∞	$\Delta_{L_{41}}$	$\Delta_{L_{42}}$	$\Delta_{L_{43}}$	∞	$\Delta_{L_{45}}$
5	∞	$\Delta_{L_{51}}$	$\Delta_{L_{52}}$	$\Delta_{L_{53}}$	$\Delta_{L_{54}}$	∞

LB_1 has been originated by calculating the SP from the source s to sink t with exactly k edges, to be computed using dynamic programming. The SP with exactly k edges from source s to sink t , as given in the directed graph in Figure 3.8. is denoted as $SP_k(s; t)$. These values can be written in an n by n matrix, as illustrated in Table 3.16.

Table 3.16. Shortest path calculation of the LB_I matrix for the makespan and the total flow time

To From	Edges	1	2	3	4	5
0	k=1	$SP_1(0; 1)$	$SP_1(0; 2)$	$SP_1(0; 3)$	$SP_1(0; 4)$	$SP_1(0; 5)$
0	k=2	$SP_2(0; 1)$	$SP_2(0; 2)$	$SP_2(0; 3)$	$SP_2(0; 4)$	$SP_2(0; 5)$
0	k=3	$SP_3(0; 1)$	$SP_3(0; 2)$	$SP_3(0; 3)$	$SP_3(0; 4)$	$SP_3(0; 5)$
0	k=4	$SP_4(0; 1)$	$SP_4(0; 2)$	$SP_4(0; 3)$	$SP_4(0; 4)$	$SP_4(0; 5)$
0	k=5	$SP_5(0; 1)$	$SP_5(0; 2)$	$SP_5(0; 3)$	$SP_5(0; 4)$	$SP_5(0; 5)$

Table 3.16 constitutes the SP computation from source 0 to all other sinks with exactly k edges in each cell. Hence, we know the cost of positioning job i to any position (edge k), which is $SP_k(0; i)$. This is the flow time of jobs in the permutation. This matrix is a base matrix for all objective functions such as makespan (maximum flow time), total flow time, total tardiness, number of tardy jobs, etc. For example, it is required for makespan to calculate the completion time of the last job positioned in the sequence. Namely, the maximum completion time of all the jobs in the sequence. From Table 3.16, $SP_5(0; i)$ values positioned in the last row are the lower bounds for the completion time of each job if they are positioned as the last job in the sequence, i.e. $SP_5(0; 2)$ is the completion time of job 2, if it is positioned as the 5th job in the sequence. Hence, the minimum of $SP_5(0; i)$ values is a LB of makespan for this example. To generalize, the **minimum of $SP_{K=\max\{k\}}(0; i)$** values are the **lower bound of the makespan** of an instance. Therefore, this is a sound lower bound for makespan.

If the total flow time criterion is examined, the flow time of jobs for each step (k edges) is recorded in Table 3.16. Thus, once we solve the assignment problem for Table 3.16, a lower bound for the overall total flow time will be achieved. The main idea for this assignment problem is to assign the jobs to the positions with the minimum cost, which is the shortest path calculation. $SP_k(0; i)$ indicates the lower bound of the flow time of the job i , if it is scheduled in the k^{th} position in the sequence. After that, the only thing to be decided is which jobs will be assigned to the position (edge). The $SP_k(0; i)$ values represent the cost of assignments. Hence, the problem turns out to be an assignment problem to be solved in a reasonable time. To conclude, only one job will be scheduled at only one position so that the sum of the $SP_k(0; i)$ costs yielded by the assignment

matrix are minimized. In the end, the *objective function of the assignment problem led by LB_1 matrix* will be a *lower bound for the total flow time* of the given instance. Thus, this is a sound lower bound for the overall total flow time.

For the total tardiness, we record the tardiness of each job differences (maximum of assigned completion time minus due date or zero), that is, corresponding individual tardiness values - $\max\{SP_k(0; i) - DD_i, 0\}$ -. Once we solve the overall assignment problem over this matrix, we will have *lower bound for overall tardiness. Namely, the objective function of the assignment problem led by LB_1 matrix for the total tardiness* will be a *lower bound for the total tardiness* of the given instance. Since we choose the best possible positions for all jobs, with the best job subsequences possible, to get the minimum total tardiness value. Thus, this is a sound lower bound for total tardiness. This computation is illustrated in Table 3.17.

Table 3.17. Shortest path calculation of the LB_1 matrix for the total tardiness

To From	Edges	(0; 1)	(0; 2)	(0; 3)	(0; 4)	(0; 5)
0	k=1	$\max\{SP_1(0; 1) - DD_1, 0\}$	$\max\{SP_1(0; 2) - DD_2, 0\}$	$\max\{SP_1(0; 3) - DD_3, 0\}$	$\max\{SP_1(0; 4) - DD_4, 0\}$	$\max\{SP_1(0; 5) - DD_5, 0\}$
0	k=2	$\max\{SP_2(0; 1) - DD_1, 0\}$	$\max\{SP_2(0; 2) - DD_2, 0\}$	$\max\{SP_2(0; 3) - DD_3, 0\}$	$\max\{SP_2(0; 4) - DD_4, 0\}$	$\max\{SP_2(0; 5) - DD_5, 0\}$
0	k=3	$\max\{SP_3(0; 1) - DD_1, 0\}$	$\max\{SP_3(0; 2) - DD_2, 0\}$	$\max\{SP_3(0; 3) - DD_3, 0\}$	$\max\{SP_3(0; 4) - DD_4, 0\}$	$\max\{SP_3(0; 5) - DD_5, 0\}$
0	k=4	$\max\{SP_4(0; 1) - DD_1, 0\}$	$\max\{SP_4(0; 2) - DD_2, 0\}$	$\max\{SP_4(0; 3) - DD_3, 0\}$	$\max\{SP_4(0; 4) - DD_4, 0\}$	$\max\{SP_4(0; 5) - DD_5, 0\}$
0	k=5	$\max\{SP_5(0; 1) - DD_1, 0\}$	$\max\{SP_5(0; 2) - DD_2, 0\}$	$\max\{SP_5(0; 3) - DD_3, 0\}$	$\max\{SP_5(0; 4) - DD_4, 0\}$	$\max\{SP_5(0; 5) - DD_5, 0\}$

According to Table 3.17, $\max\{SP_k(0; i) - DD_i, 0\}$ indicates the lower bound of the tardiness of job i , if it is scheduled in the k^{th} position in the sequence. After that, the only thing to be decided is which jobs will be assigned to the position (edge). Now, the $\max\{SP_k(0; i) - DD_i, 0\}$ values represent the cost of assignments. In the end, the objective function of the assignment will be a lower bound for the total tardiness of the given instance.

A similar trend can be observed for the minimization of the number of tardy jobs as well. For number of tardy jobs, we record either 0 or 1 values as to control whether a job is tardy or not (minimum of 1 or -maximum of minus due date from the assigned completion time or 0-), that is, corresponding individual tardiness condition- $\min \{1, \max \{SP_k(0; i) - DD_i, 0\}\}$ -. Once we solve the overall assignment problem over this matrix, we will have *lower bound for overall number of tardy jobs*. Namely, the *objective function of the assignment problem led by LB1 matrix for the number of tardy jobs* will be a *lower bound for number of tardy jobs* of the given instance. Since we choose the best possible positions for all jobs, with the best job subsequences possible, to get the minimum number of tardy jobs value. Thus, this is a sound lower bound for number of tardy jobs. This computation is illustrated in Table 3.18.

Table 3.18. Shortest path calculation of the LB_1 matrix for the number of tardy jobs

To From	Edges	(0; 1)	(0; 2)	(0; 3)	(0; 4)	(0; 5)
0	k=1	$\min \{1, \max \{SP_1(0; 1) - DD_1, 0\}\}$	$\min \{1, \max \{SP_1(0; 2) - DD_2, 0\}\}$	$\min \{1, \max \{SP_1(0; 3) - DD_3, 0\}\}$	$\min \{1, \max \{SP_1(0; 4) - DD_4, 0\}\}$	$\min \{1, \max \{SP_1(0; 5) - DD_5, 0\}\}$
0	k=2	$\min \{1, \max \{SP_2(0; 1) - DD_1, 0\}\}$	$\min \{1, \max \{SP_2(0; 2) - DD_2, 0\}\}$	$\min \{1, \max \{SP_2(0; 3) - DD_3, 0\}\}$	$\min \{1, \max \{SP_2(0; 4) - DD_4, 0\}\}$	$\min \{1, \max \{SP_2(0; 5) - DD_5, 0\}\}$
0	k=3	$\min \{1, \max \{SP_3(0; 1) - DD_1, 0\}\}$	$\min \{1, \max \{SP_3(0; 2) - DD_2, 0\}\}$	$\min \{1, \max \{SP_3(0; 3) - DD_3, 0\}\}$	$\min \{1, \max \{SP_3(0; 4) - DD_4, 0\}\}$	$\min \{1, \max \{SP_3(0; 5) - DD_5, 0\}\}$
0	k=4	$\min \{1, \max \{SP_4(0; 1) - DD_1, 0\}\}$	$\min \{1, \max \{SP_4(0; 2) - DD_2, 0\}\}$	$\min \{1, \max \{SP_4(0; 3) - DD_3, 0\}\}$	$\min \{1, \max \{SP_4(0; 4) - DD_4, 0\}\}$	$\min \{1, \max \{SP_4(0; 5) - DD_5, 0\}\}$
0	k=5	$\min \{1, \max \{SP_5(0; 1) - DD_1, 0\}\}$	$\min \{1, \max \{SP_5(0; 2) - DD_2, 0\}\}$	$\min \{1, \max \{SP_5(0; 3) - DD_3, 0\}\}$	$\min \{1, \max \{SP_5(0; 4) - DD_4, 0\}\}$	$\min \{1, \max \{SP_5(0; 5) - DD_5, 0\}\}$

According to Table 3.18, $\min \{1, \max \{SP_k(0; i) - DD_i, 0\}\}$ indicates the lower bound of the tardiness condition of job i , if it is scheduled in the k^{th} position in the sequence. After that, the only thing to be decided is which jobs will be assigned to the position (edge). Now, the $\min \{1, \max \{SP_k(0; i) - DD_i, 0\}\}$ values represent the cost of

assignments. In the end, the objective function of the assignment will be a lower bound for the number of tardy jobs of the given instance.

LB_I-Lifted:

Unfortunately, each cell of this matrix might contain a tour (or several tours) because repetition is allowed. Therefore, we aim to find a new lower bound that is better than LB_I , yet still achievable in polynomial time. In other words, we intend to break some of the tours and slightly increase the values in the matrix. This, in turn, will raise LB_I .

Instead of directly calculating $SP_k(0; i)$ values, we aim to calculate $SP_k(0; i; j)$ values for each step size, the SP from 0 to i , where j is the first node visited after 0. We fix j which is the first node after 0 and i that is the destination node. There are two tags for nodes. In other words, the starting node of the SP will be determined by each step of iterations. The distance matrix given in Table 3.15 will be the same; however, we will iteratively calculate the SP values among nodes as follows.

$$SP_k(0; i; j) = \min_t \{ SP_{k-1}(0; t | t \neq i, j; j) + \Delta_{L_{ti}} \} \quad \forall t | t \neq i \quad (3.53)$$

$$SP_k(0; i; j) = \infty | j = i \quad (3.54)$$

Equation (3.53) computes the SP from 0 to i , where j is the first node visited after 0. The idea of this equation is not to visit either i or j during the path. As demonstrated in Equation (3.54), it takes value infinity if $i = j$, namely the diagonal of the matrix. Tables 3.19-3.23 illustrates the iterative shortest path calculations for each step size k .

Table 3.19. Shortest path calculation of the *LB₁-Lifted* matrix for the makespan and the total flow time when $k=1$

		k=1 edges				
$SP_1(0; i; j)$	To From- by 1 st Passing	1	2	3	4	5
$SP_1(0; i; 1)$	0-1	d_{01}	∞	∞	∞	∞
$SP_1(0; i; 2)$	0-2	∞	d_{02}	∞	∞	∞
$SP_1(0; i; 3)$	0-3	∞	∞	d_{03}	∞	∞
$SP_1(0; i; 4)$	0-4	∞	∞	∞	d_{04}	∞
$SP_1(0; i; 5)$	0-5	∞	∞	∞	∞	d_{05}

Table 3.20. Shortest path calculation of the *LB₁-Lifted* matrix for the makespan and the total flow time when $k=2$

		k=2 edges				
$SP_2(0; i; j)$	To From- by 1 st Passing	1	2	3	4	5
$SP_2(0; i; 1)$	0-1	∞	$\min \begin{cases} SP_1(0; 1; 1) + \Delta_{L_{12}} \\ \infty + \infty \\ \infty + \Delta_{L_{22}} \\ \infty + \Delta_{L_{42}} \\ \infty + \Delta_{L_{52}} \end{cases}$	$\min \begin{cases} SP_1(0; 1; 1) + \Delta_{L_{13}} \\ \infty + \Delta_{L_{23}} \\ \infty + \infty \\ \infty + \Delta_{L_{43}} \\ \infty + \Delta_{L_{53}} \end{cases}$	$\min \begin{cases} SP_1(0; 1; 1) + \Delta_{L_{14}} \\ \infty + \Delta_{L_{24}} \\ \infty + \Delta_{L_{34}} \\ \infty + \infty \\ \infty + \Delta_{L_{54}} \end{cases}$	$\min \begin{cases} SP_1(0; 1; 1) + \Delta_{L_{15}} \\ \infty + \Delta_{L_{25}} \\ \infty + \Delta_{L_{35}} \\ \infty + \Delta_{L_{45}} \\ \infty + \infty \end{cases}$
$SP_2(0; i; 2)$	0-2	$\min \begin{cases} \infty + \infty \\ SP_1(0; 2; 2) + \Delta_{L_{21}} \\ \infty + \Delta_{L_{31}} \\ \infty + \Delta_{L_{41}} \\ \infty + \Delta_{L_{51}} \end{cases}$	∞	$\min \begin{cases} \infty + \Delta_{L_{12}} \\ SP_1(0; 2; 2) + \Delta_{L_{23}} \\ \infty + \infty \\ \infty + \Delta_{L_{43}} \\ \infty + \Delta_{L_{53}} \end{cases}$	$\min \begin{cases} \infty + \Delta_{L_{14}} \\ SP_1(0; 2; 2) + \Delta_{L_{24}} \\ \infty + \Delta_{L_{34}} \\ \infty + \infty \\ \infty + \Delta_{L_{54}} \end{cases}$	$\min \begin{cases} \infty + \Delta_{L_{15}} \\ SP_1(0; 2; 2) + \Delta_{L_{25}} \\ \infty + \Delta_{L_{35}} \\ \infty + \Delta_{L_{45}} \\ \infty + \infty \end{cases}$
$SP_2(0; i; 3)$	0-3	$\min \begin{cases} \infty + \infty \\ \infty + \Delta_{L_{21}} \\ SP_1(0; 3; 3) + \Delta_{L_{31}} \\ \infty + \Delta_{L_{41}} \\ \infty + \Delta_{L_{51}} \end{cases}$	$\min \begin{cases} \infty + \Delta_{L_{12}} \\ \infty + \infty \\ SP_1(0; 3; 3) + \Delta_{L_{23}} \\ \infty + \Delta_{L_{42}} \\ \infty + \Delta_{L_{52}} \end{cases}$	∞	$\min \begin{cases} \infty + \Delta_{L_{14}} \\ \infty + \Delta_{L_{24}} \\ SP_1(0; 3; 3) + \Delta_{L_{34}} \\ \infty + \infty \\ \infty + \Delta_{L_{54}} \end{cases}$	$\min \begin{cases} \infty + \Delta_{L_{15}} \\ \infty + \Delta_{L_{25}} \\ SP_1(0; 3; 3) + \Delta_{L_{35}} \\ \infty + \Delta_{L_{45}} \\ \infty + \infty \end{cases}$
$SP_2(0; i; 4)$	0-4	$\min \begin{cases} \infty + \infty \\ \infty + \Delta_{L_{21}} \\ \infty + \Delta_{L_{31}} \\ SP_1(0; 4; 4) + \Delta_{L_{41}} \\ \infty + \Delta_{L_{51}} \end{cases}$	$\min \begin{cases} \infty + \Delta_{L_{12}} \\ \infty + \infty \\ \infty + \Delta_{L_{32}} \\ SP_1(0; 4; 4) + \Delta_{L_{42}} \\ \infty + \Delta_{L_{52}} \end{cases}$	$\min \begin{cases} \infty + \Delta_{L_{13}} \\ \infty + \Delta_{L_{23}} \\ \infty + \infty \\ SP_1(0; 4; 4) + \Delta_{L_{33}} \\ \infty + \Delta_{L_{53}} \end{cases}$	∞	$\min \begin{cases} \infty + \Delta_{L_{15}} \\ \infty + \Delta_{L_{25}} \\ \infty + \Delta_{L_{35}} \\ SP_1(0; 4; 4) + \Delta_{L_{45}} \\ \infty + \infty \end{cases}$
$SP_2(0; i; 5)$	0-5	$\min \begin{cases} \infty + \infty \\ \infty + \Delta_{L_{21}} \\ \infty + \Delta_{L_{31}} \\ \infty + \Delta_{L_{41}} \\ SP_1(0; 5; 5) + \Delta_{L_{51}} \end{cases}$	$\min \begin{cases} \infty + \Delta_{L_{12}} \\ \infty + \infty \\ \infty + \Delta_{L_{32}} \\ \infty + \Delta_{L_{42}} \\ SP_1(0; 5; 5) + \Delta_{L_{52}} \end{cases}$	$\min \begin{cases} \infty + \Delta_{L_{13}} \\ \infty + \Delta_{L_{23}} \\ \infty + \infty \\ \infty + \Delta_{L_{43}} \\ SP_1(0; 5; 5) + \Delta_{L_{53}} \end{cases}$	$\min \begin{cases} \infty + \Delta_{L_{14}} \\ \infty + \Delta_{L_{24}} \\ \infty + \Delta_{L_{34}} \\ \infty + \infty \\ SP_1(0; 5; 5) + \Delta_{L_{44}} \end{cases}$	∞

Table 3.23. Shortest path calculation of the LB_1 -Lifted matrix for the makespan and the total flow time when $k=5$

k=5 edges						
$SP_5(0; i; j)$	To From- by 1 st Passing	1	2	3	4	5
$SP_5(0; i; 1)$	0-1	∞	$\min \begin{Bmatrix} \infty + \Delta_{i,12} \\ SP_5(0; 2; 1) + \infty \\ SP_5(0; 3; 1) + \Delta_{i,32} \\ SP_5(0; 4; 1) + \Delta_{i,42} \\ SP_5(0; 5; 1) + \Delta_{i,52} \end{Bmatrix}$	$\min \begin{Bmatrix} \infty + \Delta_{i,13} \\ SP_5(0; 2; 1) + \Delta_{i,23} \\ SP_5(0; 3; 1) + \infty \\ SP_5(0; 4; 1) + \Delta_{i,43} \\ SP_5(0; 5; 1) + \Delta_{i,53} \end{Bmatrix}$	$\min \begin{Bmatrix} \infty + \Delta_{i,14} \\ SP_5(0; 2; 1) + \Delta_{i,24} \\ SP_5(0; 3; 1) + \Delta_{i,34} \\ SP_5(0; 4; 1) + \infty \\ SP_5(0; 5; 1) + \Delta_{i,54} \end{Bmatrix}$	$\min \begin{Bmatrix} \infty + \Delta_{i,15} \\ SP_5(0; 2; 1) + \Delta_{i,25} \\ SP_5(0; 3; 1) + \Delta_{i,35} \\ SP_5(0; 4; 1) + \Delta_{i,45} \\ SP_5(0; 5; 1) + \infty \end{Bmatrix}$
$SP_5(0; i; 2)$	0-2	$\min \begin{Bmatrix} SP_5(0; 1; 2) + \infty \\ \infty + \Delta_{i,21} \\ SP_5(0; 3; 2) + \Delta_{i,31} \\ SP_5(0; 4; 2) + \Delta_{i,41} \\ SP_5(0; 5; 2) + \Delta_{i,51} \end{Bmatrix}$	∞	$\min \begin{Bmatrix} SP_5(0; 1; 2) + \Delta_{i,13} \\ \infty + \Delta_{i,23} \\ SP_5(0; 3; 2) + \infty \\ SP_5(0; 4; 2) + \Delta_{i,43} \\ SP_5(0; 5; 2) + \Delta_{i,53} \end{Bmatrix}$	$\min \begin{Bmatrix} SP_5(0; 1; 2) + \Delta_{i,14} \\ \infty + \Delta_{i,24} \\ SP_5(0; 3; 2) + \Delta_{i,34} \\ SP_5(0; 4; 2) + \infty \\ SP_5(0; 5; 2) + \Delta_{i,54} \end{Bmatrix}$	$\min \begin{Bmatrix} SP_5(0; 1; 2) + \Delta_{i,15} \\ \infty + \Delta_{i,25} \\ SP_5(0; 3; 2) + \Delta_{i,35} \\ SP_5(0; 4; 2) + \Delta_{i,45} \\ SP_5(0; 5; 2) + \infty \end{Bmatrix}$
$SP_5(0; i; 3)$	0-3	$\min \begin{Bmatrix} SP_5(0; 1; 3) + \infty \\ SP_5(0; 2; 3) + \Delta_{i,21} \\ \infty + \Delta_{i,31} \\ SP_5(0; 4; 3) + \Delta_{i,41} \\ SP_5(0; 5; 3) + \Delta_{i,51} \end{Bmatrix}$	$\min \begin{Bmatrix} SP_5(0; 1; 3) + \Delta_{i,12} \\ SP_5(0; 2; 3) + \infty \\ \infty + \Delta_{i,32} \\ SP_5(0; 4; 3) + \Delta_{i,42} \\ SP_5(0; 5; 3) + \Delta_{i,52} \end{Bmatrix}$	∞	$\min \begin{Bmatrix} SP_5(0; 1; 3) + \Delta_{i,14} \\ SP_5(0; 2; 3) + \Delta_{i,24} \\ \infty + \Delta_{i,34} \\ SP_5(0; 4; 3) + \infty \\ SP_5(0; 5; 3) + \Delta_{i,54} \end{Bmatrix}$	$\min \begin{Bmatrix} SP_5(0; 1; 3) + \Delta_{i,15} \\ SP_5(0; 2; 3) + \Delta_{i,25} \\ \infty + \Delta_{i,35} \\ SP_5(0; 4; 3) + \Delta_{i,45} \\ SP_5(0; 5; 3) + \infty \end{Bmatrix}$
$SP_5(0; i; 4)$	0-4	$\min \begin{Bmatrix} SP_5(0; 1; 4) + \infty \\ SP_5(0; 2; 4) + \Delta_{i,21} \\ SP_5(0; 3; 4) + \Delta_{i,31} \\ \infty + \Delta_{i,41} \\ SP_5(0; 5; 4) + \Delta_{i,51} \end{Bmatrix}$	$\min \begin{Bmatrix} SP_5(0; 1; 4) + \Delta_{i,12} \\ SP_5(0; 2; 4) + \infty \\ SP_5(0; 3; 4) + \Delta_{i,32} \\ \infty + \Delta_{i,42} \\ SP_5(0; 5; 4) + \Delta_{i,52} \end{Bmatrix}$	$\min \begin{Bmatrix} SP_5(0; 1; 4) + \Delta_{i,13} \\ SP_5(0; 2; 4) + \Delta_{i,23} \\ SP_5(0; 3; 4) + \infty \\ \infty + \Delta_{i,43} \\ SP_5(0; 5; 4) + \Delta_{i,53} \end{Bmatrix}$	∞	$\min \begin{Bmatrix} SP_5(0; 1; 4) + \Delta_{i,15} \\ SP_5(0; 2; 4) + \Delta_{i,25} \\ SP_5(0; 3; 4) + \Delta_{i,35} \\ \infty + \Delta_{i,45} \\ SP_5(0; 5; 4) + \infty \end{Bmatrix}$
$SP_5(0; i; 5)$	0-5	$\min \begin{Bmatrix} SP_5(0; 1; 5) + \infty \\ SP_5(0; 2; 5) + \Delta_{i,21} \\ SP_5(0; 3; 5) + \Delta_{i,31} \\ SP_5(0; 4; 5) + \Delta_{i,41} \\ \infty + \Delta_{i,51} \end{Bmatrix}$	$\min \begin{Bmatrix} SP_5(0; 1; 5) + \Delta_{i,12} \\ SP_5(0; 2; 5) + \infty \\ SP_5(0; 3; 5) + \Delta_{i,32} \\ SP_5(0; 4; 5) + \Delta_{i,42} \\ \infty + \Delta_{i,52} \end{Bmatrix}$	$\min \begin{Bmatrix} SP_5(0; 1; 5) + \Delta_{i,13} \\ SP_5(0; 2; 5) + \Delta_{i,23} \\ SP_5(0; 3; 5) + \infty \\ SP_5(0; 4; 5) + \Delta_{i,43} \\ \infty + \Delta_{i,53} \end{Bmatrix}$	$\min \begin{Bmatrix} SP_5(0; 1; 5) + \Delta_{i,14} \\ SP_5(0; 2; 5) + \Delta_{i,24} \\ SP_5(0; 3; 5) + \Delta_{i,34} \\ SP_5(0; 4; 5) + \infty \\ \infty + \Delta_{i,54} \end{Bmatrix}$	∞

Tables 3.19-3.23. consists of the SP computation from all sources to all other sinks with exactly k edges in the matrix. Firstly, we compute for one edge, namely the path's cost for reaching each node, that is, the total processing time of each job. Then, for two edges, we compute the shortest paths with precisely two edges from all sources to all other sinks. This process is repeated until we reach the k edges as the path size. Following that, we calculate the minimum values of each column corresponding to each k edge to reach the final assignment matrix, which is illustrated in Table 3.24.

Table 3.24. The Summary of shortest path calculation of the *LB₁-Lifted* matrix for the makespan and the total flow time

To From	Edges	1	2	3	4	5
0	k=1	$\min \begin{Bmatrix} d_{01} \\ \infty \\ \infty \\ \infty \end{Bmatrix}$	$\min \begin{Bmatrix} \infty \\ d_{02} \\ \infty \\ \infty \end{Bmatrix}$	$\min \begin{Bmatrix} \infty \\ \infty \\ d_{03} \\ \infty \end{Bmatrix}$	$\min \begin{Bmatrix} \infty \\ \infty \\ \infty \\ d_{04} \end{Bmatrix}$	$\min \begin{Bmatrix} \infty \\ \infty \\ \infty \\ \infty \\ d_{05} \end{Bmatrix}$
0	k=2	$\min \begin{Bmatrix} \infty \\ SP_2(0; 1; 2) \\ SP_2(0; 1; 3) \\ SP_2(0; 1; 4) \\ SP_2(0; 1; 5) \end{Bmatrix}$	$\min \begin{Bmatrix} SP_2(0; 2; 1) \\ \infty \\ SP_2(0; 2; 3) \\ SP_2(0; 2; 4) \\ SP_2(0; 2; 5) \end{Bmatrix}$	$\min \begin{Bmatrix} SP_2(0; 3; 1) \\ SP_2(0; 3; 2) \\ \infty \\ SP_2(0; 3; 4) \\ SP_2(0; 3; 5) \end{Bmatrix}$	$\min \begin{Bmatrix} SP_2(0; 4; 1) \\ SP_2(0; 4; 2) \\ SP_2(0; 4; 3) \\ \infty \\ SP_2(0; 4; 5) \end{Bmatrix}$	$\min \begin{Bmatrix} SP_2(0; 5; 1) \\ SP_2(0; 5; 2) \\ SP_2(0; 5; 3) \\ SP_2(0; 5; 4) \\ \infty \end{Bmatrix}$
0	k=3	$\min \begin{Bmatrix} \infty \\ SP_3(0; 1; 2) \\ SP_3(0; 1; 3) \\ SP_3(0; 1; 4) \\ SP_3(0; 1; 5) \end{Bmatrix}$	$\min \begin{Bmatrix} SP_3(0; 2; 1) \\ \infty \\ SP_3(0; 2; 3) \\ SP_3(0; 2; 4) \\ SP_3(0; 2; 5) \end{Bmatrix}$	$\min \begin{Bmatrix} SP_3(0; 3; 1) \\ SP_3(0; 3; 2) \\ \infty \\ SP_3(0; 3; 4) \\ SP_3(0; 3; 5) \end{Bmatrix}$	$\min \begin{Bmatrix} SP_3(0; 4; 1) \\ SP_3(0; 4; 2) \\ SP_3(0; 4; 3) \\ \infty \\ SP_3(0; 4; 5) \end{Bmatrix}$	$\min \begin{Bmatrix} SP_3(0; 5; 1) \\ SP_3(0; 5; 2) \\ SP_3(0; 5; 3) \\ SP_3(0; 5; 4) \\ \infty \end{Bmatrix}$
0	k=4	$\min \begin{Bmatrix} \infty \\ SP_4(0; 1; 2) \\ SP_4(0; 1; 3) \\ SP_4(0; 1; 4) \\ SP_4(0; 1; 5) \end{Bmatrix}$	$\min \begin{Bmatrix} SP_4(0; 2; 1) \\ \infty \\ SP_4(0; 2; 3) \\ SP_4(0; 2; 4) \\ SP_4(0; 2; 5) \end{Bmatrix}$	$\min \begin{Bmatrix} SP_4(0; 3; 1) \\ SP_4(0; 3; 2) \\ \infty \\ SP_4(0; 3; 4) \\ SP_4(0; 3; 5) \end{Bmatrix}$	$\min \begin{Bmatrix} SP_4(0; 4; 1) \\ SP_4(0; 4; 2) \\ SP_4(0; 4; 3) \\ \infty \\ SP_4(0; 4; 5) \end{Bmatrix}$	$\min \begin{Bmatrix} SP_4(0; 5; 1) \\ SP_4(0; 5; 2) \\ SP_4(0; 5; 3) \\ SP_4(0; 5; 4) \\ \infty \end{Bmatrix}$
0	k=5	$\min \begin{Bmatrix} \infty \\ SP_5(0; 1; 2) \\ SP_5(0; 1; 3) \\ SP_5(0; 1; 4) \\ SP_5(0; 1; 5) \end{Bmatrix}$	$\min \begin{Bmatrix} SP_5(0; 2; 1) \\ \infty \\ SP_5(0; 2; 3) \\ SP_5(0; 2; 4) \\ SP_5(0; 2; 5) \end{Bmatrix}$	$\min \begin{Bmatrix} SP_5(0; 3; 1) \\ SP_5(0; 3; 2) \\ \infty \\ SP_5(0; 3; 4) \\ SP_5(0; 3; 5) \end{Bmatrix}$	$\min \begin{Bmatrix} SP_5(0; 4; 1) \\ SP_5(0; 4; 2) \\ SP_5(0; 4; 3) \\ \infty \\ SP_5(0; 4; 5) \end{Bmatrix}$	$\min \begin{Bmatrix} SP_5(0; 5; 1) \\ SP_5(0; 5; 2) \\ SP_5(0; 5; 3) \\ SP_5(0; 5; 4) \\ \infty \end{Bmatrix}$

Table 3.24. constitutes the *SP* computation from source 0 to all other sinks with exactly k edges in each cell by *LB₁ – Lifted*. However, different than *LB₁* the values of each cell in Table 3.16 are enhanced by lifting. Because the tours in *LB₁* might be broken by lifting. Thus, the *SP* values in each cell will be raised. After reaching this matrix, the further steps of *LB₁* is also valid for *LB₁ – Lifted*. Meaning that the lower bound for makespan, total flow time, total tardiness, and the number of tardy jobs in *LB₁ – Lifted* can be calculated as it is performed in *LB₁*.

3.7. Computational Analysis of Lower Bounds

Lower bounds are examined for makespan, total flow time, total tardiness, and the number of tardy jobs. The instance sets provided by Taillard, E. (1993) and Vallada et al. (2015) are utilized as benchmark problems in the experimentation of lower bounds.

Regarding the Taillard benchmark set:

Tables 3.25-3.28 represent the power of *LB₁* and *LB₁ – Lifted* for makespan total flow time, total tardiness, and the number of tardy jobs on the Taillard dataset, respectively.

Table 3.25. Performance of the Lower Bounds for Makespan on the Taillard Instances

Instance Set	LB_I	$LB_I\text{-Lifted}$
	$(\text{Opt.} - LB_I) / \text{Opt.}$	$(\text{Opt.} - LB_I\text{-Lifted}) / \text{Opt.}$
5_5_Avg	7.68%	3.15%
5_10_Avg	7.80%	6.26%
5_20_Avg	8.19%	5.04%
Instance Set	$(\text{Opt.} - LB_I) / \text{Opt.}$	$(\text{Opt.} - LB_I\text{-Lifted}) / \text{Opt.}$
20_5_Avg	25.04%	23.39%
20_10_Avg	15.45%	14.50%
20_20_Avg	13.06%	12.64%
50_5_Avg	34.28%	33.69%
50_10_Avg	17.99%	17.91%
50_20_Avg	14.36%	14.33%
100_5_Avg	40.33%	39.94%
100_10_Avg	23.91%	23.77%
100_20_Avg	15.97%	15.93%
200_10_Avg	28.70%	28.65%
200_20_Avg	15.57%	15.53%
Overall Avg.	19.17%	18.20%

Table 3.26. Performance of the Lower Bounds for Total Flow Time on the Taillard Instances

Instance Set	LB_1	LB_1 -Lifted
	$(\text{Opt.} - LB_1) / \text{Opt.}$	$(\text{Opt.} - LB_1\text{-Lifted}) / \text{Opt.}$
5_5_Avg	1.19%	0.70%
5_10_Avg	0.94%	0.75%
5_20_Avg	0.76%	0.31%
Instance Set	$(\text{Best Known} - LB_1) / \text{Best Known}$	$(\text{Best Known} - LB_1\text{-Lifted}) / \text{Best Known}$
20_5_Avg	13.21%	11.20%
20_10_Avg	6.79%	6.08%
20_20_Avg	5.51%	5.25%
50_5_Avg	23.13%	22.07%
50_10_Avg	10.21%	10.20%
50_20_Avg	7.64%	7.61%
100_5_Avg	31.01%	30.73%
100_10_Avg	16.70%	16.47%
100_20_Avg	10.27%	10.27%
200_10_Avg	22.82%	22.74%
200_20_Avg	11.20%	11.13%
Overall Avg.	11.53%	11.11%

Table 3.27. Performance of the lower bounds for Total Tardiness on the Taillard Instances

Instance Set	LB_I	LB_I -Lifted
	$(\text{Opt.} - LB_I) / \text{Opt.}$	$(\text{Opt.} - LB_I\text{-Lifted}) / \text{Opt.}$
5_5_Avg	4.31%	2.48%
5_10_Avg	59.40%	54.74%
5_20_Avg	69.62%	57.16%
Instance Set	$(\text{Best Known} - LB_I) / \text{Best Known}$	$(\text{Best Known} - LB_I\text{-Lifted}) / \text{Best Known}$
20_5_Avg	63.06%	57.18%
20_10_Avg	97.47%	97.36%
20_20_Avg	50.71%	50.51%
50_5_Avg	40.89%	39.19%
50_10_Avg	28.61%	28.54%
50_20_Avg	44.67%	44.57%
100_5_Avg	42.42%	42.06%
100_10_Avg	30.01%	29.71%
100_20_Avg	29.68%	29.67%
200_10_Avg	36.09%	36.00%
200_20_Avg	27.95%	27.87%
Overall Avg.	44.64%	42.65%

Table 3.28. Performance of the Lower Bounds for the Number of Tardy Jobs on the Taillard Instances

Instance Set	LB_1	$LB_1\text{-Lifted}$
	$(\text{Opt.} - LB_1) / \text{Opt.}$	$(\text{Opt.} - LB_1\text{-Lifted}) / \text{Opt.}$
5_5_Avg	2.50%	2.50%
5_10_Avg	23.33%	13.33%
5_20_Avg	30.00%	10.00%
Instance Set	$(\text{Best Known} - LB_1) / \text{Best Known}$	$(\text{Best Known} - LB_1\text{-Lifted}) / \text{Best Known}$
20_5_Avg	40.16%	36.27%
20_10_Avg	87.50%	87.50%
20_20_Avg	41.50%	41.50%
50_5_Avg	18.84%	17.66%
50_10_Avg	23.63%	23.63%
50_20_Avg	42.06%	42.06%
100_5_Avg	14.47%	14.23%
100_10_Avg	13.61%	13.48%
100_20_Avg	17.86%	17.86%
200_10_Avg	7.98%	7.92%
200_20_Avg	8.65%	8.59%
Overall Avg.	26.58%	24.04%

As seen in Table 3.25, the lower bounds for makespan are compared to the optimal solutions in all instances, as these optimal solutions are available in the literature (Lin and Ying (2016)). In contrast, as shown in Tables 3.26-3.28, while the lower bounds for total flow time, total tardiness, and the number of tardy jobs are compared to the optimal solutions in small instances, they are compared to the best-known solutions in large instances. The performance of lower bounds is quite reasonable compared with the mathematical model solutions, as seen in Tables 3.25-3.28. On average, LB_1 shows a **19.17%** gap for the makespan criterion as seen in Table 3.25. This value is improved to **18.20%** by the lifting mechanism. For the total flow time criterion, LB_1 has an **11.53%** gap, as seen in Table 3.26, which is improved to **11.11%** by the lifting mechanism. For total tardiness, LB_1 has a **44.64%** gap, as seen in Table 3.27, whereas it is improved to **42.65%** by the lifting mechanism. For the number of tardy jobs, LB_1 has a **26.58%** gap, as seen in Table 3.28, whereas it is improved to **24.04%** by the lifting mechanism. The LB_1 are generally reasonable and show gaps compared to the

optimal solutions in small instances and the best-known solutions in large instances, which are consistently reduced by the lifting mechanism across different criteria. The lifting mechanism improves the performance of LB_1 across various criteria, reducing gaps in makespan, total flow time, total tardiness, and the number of tardy jobs in the *Taillard* benchmark problems.

Regarding the VRF benchmark set:

Tables 3.29-3.32 represent the power of LB_1 and $LB_1 - Lifted$ for makespan total flow time, total tardiness, and the number of tardy jobs on the VRF dataset, respectively. The due dates used for minimizing total tardiness and the number of tardy jobs in the VRF instances are based on those generated in Section 3.8.4 of this thesis.

Table 3.29. Performance of the Lower Bounds for Makespan on the VRF Instances

Instance Set	LB ₁	LB ₁ -Lifted
	(Opt. – LB ₁) / Opt.	(Opt. – LB ₁ -Lifted) / Opt.
10_5_Avg	34.32%	30.48%
10_10_Avg	24.79%	21.60%
10_15_Avg	15.62%	13.82%
10_20_Avg	12.34%	11.30%
Instance Set	(Opt. – LB ₁) / Opt.	(Opt. – LB ₁ -Lifted) / Opt.
20_5_Avg	28.54%	26.87%
20_10_Avg	15.74%	14.71%
20_15_Avg	13.74%	13.38%
20_20_Avg	15.16%	14.21%
30_5_Avg	34.14%	33.11%
30_10_Avg	18.58%	17.79%
30_15_Avg	12.93%	12.74%
30_20_Avg	13.93%	13.63%
40_5_Avg	31.89%	30.80%
40_10_Avg	19.42%	19.17%
40_15_Avg	15.77%	15.63%
40_20_Avg	13.79%	13.66%
50_5_Avg	33.40%	32.59%
50_10_Avg	17.48%	16.86%
50_15_Avg	14.12%	14.06%
50_20_Avg	15.11%	15.02%
60_5_Avg	33.77%	33.04%
60_10_Avg	20.02%	19.52%
60_15_Avg	14.61%	14.47%
60_20_Avg	14.87%	14.75%
Overall Avg.	20.17%	19.30%

Table 3.30. Performance of the Lower Bounds for Total Flow Time on the VRF Instances

Instance Set	LB ₁	LB ₁ -Lifted
	(Opt. – LB ₁) / Opt.	(Opt. – LB ₁ -Lifted)/ Opt.
10_5_Avg	9.48%	6.93%
10_10_Avg	5.00%	3.83%
10_15_Avg	2.92%	2.68%
10_20_Avg	2.90%	2.84%
Instance Set	(Best Known – LB ₁) / Best Known	(Best Known – LB ₁ -Lifted)/ Best Known
20_5_Avg	14.41%	12.86%
20_10_Avg	5.74%	5.32%
20_15_Avg	5.13%	4.91%
20_20_Avg	6.36%	5.85%
30_5_Avg	23.29%	21.87%
30_10_Avg	9.82%	9.19%
30_15_Avg	5.90%	5.84%
30_20_Avg	7.14%	6.97%
40_5_Avg	20.82%	19.85%
40_10_Avg	11.83%	11.71%
40_15_Avg	10.00%	9.92%
40_20_Avg	8.28%	8.26%
50_5_Avg	23.69%	22.88%
50_10_Avg	11.67%	11.07%
50_15_Avg	9.58%	9.56%
50_20_Avg	10.07%	10.07%
60_5_Avg	24.97%	23.89%
60_10_Avg	14.28%	13.73%
60_15_Avg	10.39%	10.29%
60_20_Avg	10.68%	10.59%
Overall Avg.	11.01%	10.45%

Table 3.31. Performance of the Lower Bounds for Total Tardiness on the VRF Instances

Instance Set	LB _i	LB _i -Lifted
	(Opt. – LB _i) / Opt.	(Opt. – LB _i -Lifted) / Opt.
10_5_Avg	37.24%	32.05%
10_10_Avg	21.82%	17.51%
10_15_Avg	13.60%	12.36%
10_20_Avg	8.22%	7.92%
Instance Set	(Best Known – LB_i) / Best Known	(Best Known – LB_i-Lifted) / Best Known
20_5_Avg	55.77%	52.58%
20_10_Avg	26.19%	24.95%
20_15_Avg	23.31%	22.69%
20_20_Avg	24.84%	23.37%
30_5_Avg	73.27%	71.35%
30_10_Avg	44.08%	42.31%
30_15_Avg	33.68%	33.46%
30_20_Avg	32.72%	32.15%
40_5_Avg	78.75%	76.79%
40_10_Avg	55.79%	55.26%
40_15_Avg	46.86%	46.53%
40_20_Avg	40.38%	40.25%
50_5_Avg	85.13%	84.48%
50_10_Avg	58.27%	56.46%
50_15_Avg	45.23%	45.13%
50_20_Avg	48.17%	48.13%
60_5_Avg	90.52%	89.19%
60_10_Avg	68.09%	66.54%
60_15_Avg	53.83%	53.48%
60_20_Avg	55.12%	54.79%
Overall Avg.	46.70%	45.41%

Table 3.32. Performance of the Lower Bounds for the Number of Tardy Jobs on the VRF Instances

Instance Set	LB_1	LB_1 -Lifted
	$(Opt. - LB_1) / Opt.$	$(Opt. - LB_1-Lifted) / Opt.$
10_5_Avg	14.10%	14.10%
10_10_Avg	12.54%	10.54%
10_15_Avg	5.95%	5.95%
10_20_Avg	4.11%	2.68%
Instance Set	(Best Known $-LB_1$) / Best Known	(Best Known $-LB_1$ -Lifted)/ Best Known
20_5_Avg	36.79%	34.56%
20_10_Avg	26.66%	25.75%
20_15_Avg	21.14%	21.14%
20_20_Avg	18.98%	18.98%
30_5_Avg	56.83%	56.83%
30_10_Avg	43.77%	42.29%
30_15_Avg	33.65%	33.65%
30_20_Avg	38.82%	38.82%
40_5_Avg	68.57%	68.02%
40_10_Avg	56.81%	56.22%
40_15_Avg	50.35%	50.35%
40_20_Avg	48.29%	48.29%
50_5_Avg	76.31%	75.86%
50_10_Avg	61.29%	60.42%
50_15_Avg	52.98%	52.98%
50_20_Avg	57.90%	57.90%
60_5_Avg	82.81%	82.81%
60_10_Avg	70.64%	69.89%
60_15_Avg	63.45%	63.07%
60_20_Avg	64.87%	64.53%
Overall Avg.	44.54%	44.04%

As seen in Table 3.29, the lower bounds for makespan are compared to the optimal solutions in all instances, as these optimal solutions are available in literature (Lin and Ying (2016)). In contrast, as shown in Tables 3.30-3.32, while the lower bounds for total flow time, total tardiness, and the number of tardy jobs are compared to the optimal solutions in small instances, they are compared to the best-known solutions in

large instances. Like in the Taillard set, the performance of lower bounds is quite reasonable compared with the mathematical model solutions, as seen in Tables 3.29-3.32. On average, LB_1 shows a **20.17%** gap for the makespan criterion as seen in Table 3.29. This value is improved to **19.30%** by the lifting mechanism. For the total flow time criterion, LB_1 has an **11.01%** gap, as seen in Table 3.30, which is improved to **10.45%** by the lifting mechanism. For total tardiness, LB_1 has a **46.70%** gap, as seen in Table 3.31, whereas it is improved to **45.41%** by the lifting mechanism. For the number of tardy jobs, LB_1 has a **44.45%** gap, as seen in Table 3.32, whereas it is improved to **44.01%** by the lifting mechanism. The performance of LB_1 is generally reasonable and shows gaps compared to the optimal solutions in small instances and the best-known solutions in large instances, which are consistently reduced by the lifting mechanism across different criteria. The lifting mechanism improves the performance of LB_1 across various criteria, reducing gaps in makespan, total flow time, total tardiness, and the number of tardy jobs in the *VRP* benchmark problems.

Sections 3.6 and 3.7 address Research Question 3 of this thesis.

Research Question 3: Can we generate new *lower bounds* for the $(F_m|nwt|ServiceLevel)$ problems?

These sections discuss the $(F_m|nwt|C_{max})$, $(F_m|nwt|\sum C_i)$, $(F_m|nwt|\sum T_i)$, $(F_m|nwt|\sum U_i)$ problems, addressing Research Question 3 as follows. Two Shortest-Path guided **Lower Bound** algorithms are developed for the $(F_m|nwt|C_{max})$, $(F_m|nwt|\sum C_i)$, $(F_m|nwt|\sum T_i)$, and $(F_m|nwt|\sum U_i)$ problems: **LB_1** and **LB_1 -Lifted**. The computational results of the LBs are represented in the *Taillard* and *VRP* datasets.

3.8. Number of Tardy Jobs Minimization

Several features of objective criteria can be investigated regarding scheduling problems depending on the specific customer-based or production-based requirements. However, in many production environments, managers' success is frequently evaluated by looking at the proportion of on-time shipments; therefore, reducing the number of tardy jobs is an important goal. Since on-time shipments are essential for reducing costs and raising customer satisfaction for practically all production systems, reducing the number of tardy jobs is crucial. For each late task, manufacturers typically have to pay penalties. Regardless of how long a job is delayed, the penalty is always the same

in some production environments (Aldowaisan & Allahverdi (2012a)); Aldowaisan & Allahverdi (2012b)). For all these reasons, optimizing the number of tardy jobs plays a considerable role in scheduling problems. The number of tardy job minimization has a unique feature; different sequences can have the same number of tardy job values. In other words, various sequences can exist for various tardy job values.

Optimizing the number of tardy jobs in NWPFSs requires decent-quality due dates. Due dates are crucial factors in maximizing performance metrics connected to lateness. If no tardy job is encountered given an instance, the optimization problem has no relevance. The optimization problem also has no meaning the other way around -if all jobs are tardy in the given instance-. Even randomly generated due dates may be too loose and too tight, which may cause anomalies. Hence, having good quality due dates for jobs in optimizing lateness-related performance metrics is a must.

This section aims to optimize the performance criterion of the number of tardy jobs in NWPFSs denoted as $(F_m|nwt|\sum U_i)$. While investigating the number of tardy jobs minimization in NWPFSs from different solution methodologies perspectives, our initial goal is to develop a decent and beneficial due date generation mechanism for the benchmark problems. Thus, a novel due date generation mechanism for the jobs is developed, and the generated due dates are carried out while implementing proposed solution methodologies. This section of this thesis makes several contributions in literature, which can be listed as follows.

- Two mathematical model formulations, mixed-integer linear and constraint programming models, are adapted for the number of tardy jobs optimization in NWPFSs.
- A novel quick upper bound is constructed to have a fast and advantageous upper bound value for the problem and is referred to as the Sacrifice and Rearrange Heuristic (SRH).
- A new due date generation mechanism employing the SRH is proposed to acquire superior-quality due dates yielding meaningful objective function values in NWPFSs.
- A new demanding and challenging benchmark problem set with tight due dates is provided to the literature for lateness-related performance measures in NWPFSs.

The remainder of this section is structured as follows. Section 3.8.1 lists the literature review of both the NWPFSs including due date considerations and various flowshop settings. While the problem is described with the presentation of the mathematical model formulations in Section 3.8.2, the quick upper bound is explained in Section 3.8.3. In Section 3.8.4, the novel due date generation mechanism is depicted, and then the computational results of all solution methodologies are provided in Section 3.8.5.

3.8.1. Literature Review of Lateness-Related Performance Measurements in Flowshops

Several essential objective functions related to the due dates in shop floor settings deal with strict customer requests. We can categorize these types of objective functions in the literature as lateness-related objective measurements. These are tardiness, earliness, lateness, and the number of tardy jobs. Weighted and unweighted variants of these objective functions and the mean and/or maximum of these objective functions are always considered in different production environments. As previously mentioned, the main goal of this section is to explore one specific lateness-related performance measure: the *number of tardy jobs*. Thus, we first dive into the literature on various flowshop scheduling problems that aim to minimize tardy jobs. In line with this target, the literature review of the weighted and unweighted number of tardy jobs minimization on diversified flowshop settings is explored and provided below.

Earlier, Moore (1968) was the first study in the tardy job minimization literature on single-machine scheduling problems where the lateness is measured with a deferral cost function. Lenstra et al. (1977) proved the NP-hardness of the number of tardy jobs minimization in PFSPs, even for two-machine cases. Then, a class of single-machine scheduling problems with sequence-dependent setup times was explored in optimizing the maximum lateness and the number of tardy jobs by Uzsoy et al. (1992). The heuristics and dynamic programming procedures developed in that study were the first study of approximation algorithms revealing lateness-related measures in single-machine scheduling problems wherein sequence-dependent setup times and a precedence relationship exist. Later, (Bulfin & M'Hallah, 2003) enhanced a MILP model, a heuristic, and an exact algorithm for the two-machine flowshop scheduling problem in optimizing the weighted number of tardy jobs. Jolai (2005) performed a study on the problem of optimizing the number of tardy jobs on a single batch

processing machine where the batch processing machine can process some incompatible job families at the same time as a batch. In addition, a particular case in which a common due date for each job exists is evaluated. M'Hallah & Bulfin (2005) examined unrelated and identical processor scheduling problems to optimize the unweighted and weighted number of tardy jobs. According to their experimentation results, the exact method developed in that study can reach optimal solutions quickly. Chen (2009) performed another study to minimize the number of tardy jobs in a single-machine scheduling problem setting. The significant aspect of this study is that several regular maintenance periods are utilized while also deliberating the existence of non-resumable jobs. A heuristic and a branch and bound algorithm are carried out on this problem as solution methodologies. Choi & Lee (2009) explored minimizing the number of tardy jobs in a two-stage hybrid flow shop scheduling problem. Mazdeh, Hamidinia, & Karamouzian (2011) also analyzed the weighted number of tardy jobs minimization in a single-machine environment. However, their study considers batch production, delivering the batches to customers or the next workstation, revealing a delivery cost. A MILP model and a simulated annealing algorithm solve this problem. Varmazyar & Salmasi (2012) performed the first research study on the m -machine permutation flow shop scheduling problem (PFSP) with sequence-dependent setup times that consider the criterion of minimizing the number of tardy jobs. They approach the problem with a MILP model and several metaheuristics while generating tight due dates. Later, the same problem with additional time lags is studied by Dhouib et al. (2012), in which the time lags are specified as the time intervals between each subsequent job. In another study (Soroush, 2013), the single-machine scheduling problem was treated with stochastic processing times and due dates where the weighted number of tardy jobs is optimized. Following this, Allahverdi et al. (2016) developed several simulated annealing and genetic algorithms to minimize the number of tardy jobs for the first time in a two-stage assembly flowshop scheduling problem. In PFSPs, Utama (2021) illustrates the superiority of the Hybrid Whale Optimization Algorithm over the Particle Swarm Optimization and Generic Algorithms. The literature review of various flowshop settings that minimize both the number of tardy jobs and the weighted number of tardy jobs is reported in Tables 3.33 and 3.34.

Table 3.33. Various flowshop settings that minimize the (weighted) number of tardy jobs with due date considerations

Author(s)	Configuration	Objective Function	Solution Methodology	Processing Time Generation	Due Date Generation
Moore (1968)	$(1 \sum L_i)$	Number of tardy jobs	An algorithm	-	-
Uzsoy et al. (1992)	$(1 ST_{sd}, prec L_{max})$ $(1 ST_{sd}, prec \sum U_i)$	Maximum lateness, number of tardy jobs	Heuristics, dynamic programming procedures	-	-
Bulfin & M'Hallah (2003)	$(F_2 \sum w_i U_i)$	Weighted number of tardy jobs	MILP model, a heuristic, an exact algorithm	Uniform distribution [1,100]	Uniform distribution
Jolai (2005)	$(1 batch \sum U_i)$	Number of tardy jobs	Dynamic programming	-	-
M'Hallah & Bulfin (2005)	$(P \sum U_i)$ $(P \sum w_i U_i)$ $(R \sum U_i)$ $(R \sum w_i U_i)$	Number of tardy jobs, weighted number of tardy jobs	MILP models, an exact algorithm	Normal distribution	Uniform distribution
Chen (2009)	$(1 U_i)$	Number of tardy jobs	A heuristic and a branch-and-bound algorithm	Uniform distribution [1,10]	Uniform distribution using bounds (∞)
Choi & Lee, (2009)	Two-stage hybrid flowshop	Number of tardy jobs	Branch & bound, lower and upper bounds, two-phase heuristic algorithms	Uniform distribution [10,40]	Uniform distribution based on the formula of

Table 3.34. Various flowshop settings that minimize the (weighted) number of tardy jobs with due date considerations

Author(s)	Configuration	Objective Function	Solution Methodology	Processing Time Generation	Due Date Generation
Mazdeh et al. (2011)	(1 <i>batch</i> $\sum w_i U_i$ + <i>delivery cost</i>)	Weighted number of tardy jobs	MILP model, simulated annealing	Specific cases	Specific cases
Varmazyar & Salmasi (2012)	$(F_m \begin{smallmatrix} perm, \\ ST_{sd} \end{smallmatrix} \sum U_i)$	Number of tardy jobs	A MILP model, tabu search-based heuristics, imperialist competitive algorithm	Uniform distribution [1,20]	Based on a process proposed in the study
Dhouib et al. (2012)	$(F_m \begin{smallmatrix} perm, ST_{sd}, \\ time lags \end{smallmatrix} \sum U_i)$	Number of tardy jobs	Two MILP models, simulated annealing algorithm	Uniform distribution [20,100]	Uniform distribution using bounds (∞)
Soroush (2013)	(1 $\sum w_i U_i$)	Weighted number of tardy jobs	Various heuristics	Stochastic processing times	Stochastic due dates
Allahverdi et al. (2016)	Two-stage assembly flowshop	Number of tardy jobs	Simulated annealing, genetic algorithms	Uniform distribution [1,100]	Uniform distribution using bounds (∞)
Utama (2021)	$(F_m perm \sum U_i)$	Number of tardy jobs	Hybrid Whale Optimization Algorithm	CAR (Carlier, 1978) and HEL (Heller, 1960) benchmarks	Uniform distribution [1000,3000], [20,50] and [30,60]

As seen in Tables 3.33 and 3.34, the number of tardy job minimizations is a worthwhile research direction due to the minority of studies in various types of shop floor settings. Even for classical PFSPs, the lateness-related performance measurements are yet another possible study area. As mentioned previously, the main research direction of this section is to explore the *number of tardy job* minimizations in NWPFSFs. To advance along this path, the following section reviews and explains the literature on NWPFSFs related to lateness-based objective functions and the associated due dates.

Numerous studies focus on minimizing lateness-related objective functions with due date consideration in NWPFSFs. Ruiz & Allahverdi (2007) studied genetic algorithms by dominance relation for minimizing maximum lateness in NWPFSFs with sequence-independent setup times. Total tardiness and the number of tardy jobs minimization in NWPFSFs are addressed in the research studies of Aldowaisan & Allahverdi (2012a) and Aldowaisan & Allahverdi (2012b), respectively. These studies investigate different variations of simulated annealing algorithms applied to the problem at hand, and they are the first to uncover heuristics for these specific problems. The applied algorithms in the research of Aldowaisan & Allahverdi (2012a) are adapted for the sequence-independent setup times variant of the NWPFSF by the study of Aldowaisan & Allahverdi (2015). For the variant of the NWPFSF that relies on sequence dependency, Arabameri & Salmasi (2013) studied several tabu search and particle swarm-based metaheuristics to minimize weighted earliness and tardiness. Later, for total tardiness criteria, Ding et al. (2015) revealed that focusing on certain jobs, that are sensitive compared to the other jobs, aims to decrease the computational workload in algorithms, which is referred to as accelerated, and studied the accelerated version of the NEH and Iterated Greedy algorithms on the problem. Thus, they reported the superiority of the accelerated ones over the original ones. Another problem in which due date restrictions are considered a hard restriction for makespan minimization in NWPFSFs is explored (Samarghandi, 2015). They proposed a MILP and a particle swarm optimization model for the problem. Following that study, Samarghandi & Behrooz (2017) proposed mixed-integer linear programming, two quadratic mixed-integer programming models, two constraint programming models, and an exact solution algorithm for the same problem. Later, Ying et al. (2018) developed a MILP model and a two-phase enumeration algorithm for the NWPFSF with hard due date restrictions in makespan minimization. Recently, Yamada et al. (2021) proposed

constructive heuristics and iterated greedy algorithm-based metaheuristics to optimize total tardiness in NWPFPs with preventive maintenance. The literature review of NWPFPs, including due date considerations, is reported in Tables 3.35 and 3.36, and all studies are labeled according to the configuration, objective function, solution methodology, processing times generation, and due date generation.

Table 3.35. No-wait permutation flowshop settings that minimize various lateness-related objective functions with due date considerations

Author(s)	Configuration	Objective Function	Solution Methodology	Processing Time Generation	Due Date Generation
Ruiz & Allahverdi (2007)	$(F_m nwt, L_{max})$	Maximum Lateness	Genetic Algorithms, Dominance relations	Uniform distribution and Gaussian distribution	Uniform distribution Taillard
Aldowaisan & Allahverdi (2012a)	$(F_m nwt \sum T_i)$	Total tardiness	Simulated annealing algorithm, genetic algorithms	Uniform distribution	Uniform distribution using bounds (∞)
Aldowaisan & Allahverdi (2012b)	$(F_m nwt \sum U_i)$	Number of tardy jobs	Simulated annealing algorithm	Uniform distribution [1,100]	Uniform distribution using bounds (∞)
Arabameri & Salmasi (2013)	$(F_m nwt, w_i \sum E_i + w_i \sum T_i)$	Weighted earliness and tardiness	A MILP, Tabu search, and particle swarm optimization-based metaheuristics	Uniform distribution [1,20]	Uniform distribution using bounds (∞)
Aldowaisan & Allahverdi (2015)	$(F_m nwt, \sum T_i)$	Total tardiness	Simulated annealing and genetic algorithm	Uniform distribution [1,100]	Uniform distribution using bounds (∞)
Ding et al. (2015)	$(F_m nwt \sum T_i)$	Total tardiness	Accelerated NEH and accelerated iterated greedy algorithm	Taillard (Taillard, E. 1993) benchmark	Uniform distribution using bounds (∞)

Table 3.36. No-wait permutation flowshop settings that minimize various lateness-related objective functions with due date considerations

Author(s)	Configuration	Objective Function	Solution Methodology	Processing Time Generation	Due Date Generation
Samarghandi (2015)	$(F_m nwt, d_i C_{max})$	Makespan with due date Restriction	A MILP, a particle swarm optimization Algorithm	CAR (Carlier, 1978) and REC (Reeves, 1995) benchmarks	Uniform distribution based on the formula of Tari & Olfat, (2014)
Samarghandi & Behroozi (2017)	$(F_m nwt, d_i C_{max})$	Makespan With Due Date Restriction	MILP, two QMILP, and two CP Models, exact algorithm	As in Samarghandi (2015)	As in Samarghandi (2015)
Ying et al. (2018)	$(F_m nwt, d_i C_{max})$	Makespan With Due Date Restriction	MILP model, two-phase enumeration algorithm	As in Samarghandi & Behroozi (2017)	As in Samarghandi & Behroozi (2017)
Yamada et al. (2021)	$(F_m nwt \sum T_i)$	Total Tardiness	Constructive heuristics and iterated greedy algorithm-based metaheuristics	Uniform distribution [1,99]	Uniform distribution using bounds (∞)
This study	$(F_m nwt \sum U_i)$	Number of tardy jobs	A new due date generation mechanism, MILP, and CP models, a novel upper bound (Sacrifice & Rearrange Heuristic)	Taillard and VRF instances	Created by the proposed due date generation mechanism based on the Sacrifice & Rearrange Heuristic

As illustrated in Tables 3.33-3.36, it's a common practice in the scheduling literature to randomly generate processing times using a uniform distribution between $[1,99]$ and to calculate due dates based on the given formula:

$$[P(1 - T - R/2), P(1 - T + R/2)] \quad (\bowtie)$$

where P represents a lower bound of the makespan for a particular instance, while T and R stand for the tardiness factor and the due date's relative range. R indicates the extent to which the deadlines are spread out around T . The more reduced value of T , the more relaxed the due dates become, the easier it is to satisfy them. The studies that use the $(\bowtie)$ formula or its variants in generating the due dates, adjust the formula to fit the needs of the specific problem studied. This leads to both random generation of processing times and due dates, determination of the tardiness factor and relative range values, and needing a lower bound. Thus, benchmark problems that bring about too tight or loose due dates are constituted. Anomalies can arise from either tight or loose due dates. Hence, this leads to constructing trivial and/or incomparable benchmarks. Reproducing results in each paper in the literature on due date generation becomes challenging when the specific sets of mentioned instances are not provided (Vallada et al. (2008)). Some parameter setting is also required for given parameters in the formula $(\bowtie)$ such as T and R . This necessitates conducting multiple experiments for all enumeration of parameters within the provided intervals; the result is it is hard to make a comparison that is global in scope and fair in the assessment (Vallada et al. (2008)).

Moreover, some preliminary analysis is performed to illustrate anomalies in existing due dates on the Taillard benchmark set Taillard, E. (1993) when it is employed for lateness-related objective functions in NWPFPs. The due dates of these instances are taken from Minella et al. (2008). Based on the analysis of total tardiness and the number of tardy jobs minimization in some instances (Yüksel, D. (2019)), we observe zero tardiness. For instance, let us consider the 20-20 instance set given in Table 3.37; we observe zero tardiness in 4 instances due to very loose due dates. Almost 40% of the instances are trivial, contributing nothing to any benchmarking study. Therefore, this leads to a trivial and easily solvable set of instances in the benchmark set.

Table 3.37. Issues in Existing Due Dates of the Taillard Set

Instance No	Total Tardiness	Number of Tardy Jobs
20-20-01	0	0
20-20-02	216	1
20-20-03	0	0
20-20-04	0	0
20-20-05	3934	5
20-20-06	2541	4
20-20-07	364	1
20-20-08	844	2
20-20-09	258	1
20-20-10	0	0

It is demonstrated that the difficulty level associated with the number of tardy jobs minimization in PFSPs is likely influenced by the specific due date values (Hariri & Potts (1989)). The optimization problem is not meaningful if there are too few tardy jobs or no tardy jobs given the instance. The same is true when all the jobs are tardy. Loose due dates or tight due dates cause anomalies. Conversely, if we know the minimum number (k) of tardy jobs, there are $\binom{n}{k}$ possible cases. The number of tardy jobs values equal to half of the number of jobs, combinatorically speaking, yields the most challenging instances.

In short, the literature reviews the various flowshop settings that minimize the number of tardy jobs with due date considerations (see Table 3.33-3.34) and the no-wait permutation flowshop settings that minimize various lateness-related objectives with due date considerations (see Table 3.35-3.36). The aim is to highlight the gap in the literature regarding the minimization of tardy jobs and the challenges of due date generation in no-wait permutation flowshop settings. In the last rows of Table 3.36, this study is motivated to fill the literature gap in which it discovers the aspects of the number of tardy jobs minimization in NWPFSs with mathematical model formulations, an upper bounding heuristic, and a due date generation mechanism based on the developed upper bound for benchmark problems.

3.8.2. Mathematical Model Formulations

Two mathematical models, one belonging to the MILP class and the other to the CP class, have been adapted for this problem. The following parameters and decision variables, in addition to those outlined in Section 3.2, were utilized in the development of the mathematical models.

Parameters:	
ε	A small number
Decision Variables:	
U_i	1 if job i is tardy, 0 otherwise
V_p	1 if the job at position p is tardy, 0 otherwise

The adapted mathematical models are Model 3 – MILP and Model 5 – CP which are introduced in Section 3.2. For better clarity, they will be referred to as Model 1 - MILP and Model 2 - CP throughout Section 3.8.

Model 1. The MILP model			
Objective			
Minimize $\sum_{p \in N} V_p$			(3.55)
Constraints			
$\sum_{p=1}^N X_{ip} = 1$	$\forall i \in N$		(3.56)
$\sum_{i=1}^N X_{ip} = 1$	$\forall p \in N$		(3.57)
$S_p + \sum_{i=1}^N \sum_{l=1}^r P_{il} * X_{ip} \geq S_{p-1} + \sum_{i=1}^N \sum_{l=1}^r P_{il} *$	$\forall p \in \{2, \dots, N\}$		(3.58)
$X_{i,p-1} + \sum_{i=1}^N P_{ir} * X_{ip}$	$\forall r \in M$		
$S_p + \sum_{i=1}^N \sum_{r=1}^M X_{ip} * P_{ir} - \sum_{i=1}^N X_{ip} * DD_i \leq Q * V_p$	$\forall p \in N$		(3.59)
$S_p \geq 0, X_{ip}, V_p \in \{0,1\}$	$\forall i, p \in N$		(3.60)

The objective function (3.55) minimizes the total number of tardy jobs. Constraints (3.56) and (3.57) assign jobs to precisely one position, and one position is preserved for only one job. Constraint (3.58) controls the completion times of all jobs in the subsequent positions, ensuring that these completion times are larger than those of previous positions. The completion times are calculated by adding the starting times

of the jobs at positions plus the processing times until the associated machine. Constraint (3.59) holds the value of the binary variable, the tardiness of a job at a position. V_p is either 1 or 0 depending on the tardiness of jobs at position p . Constraint (3.60) is the sign restrictions and binary variables.

Note that to accurately determine the tardiness amount for each job, it is essential to extend the objective function as follows.

$$\text{Minimize } \sum_{p \in N} V_p + \varepsilon * \sum_{p \in N} (S_p + \sum_{i=1}^N \sum_{r=1}^M P_{ir} * X_{ip}) \quad (3.61)$$

The first part of the equation (3.61) minimizes the total number of tardy jobs, while the second part minimizes the total flow time of the jobs. The total flow time of each job is computed by adding the job's start time at each position to its processing time. This approach allows for the accurate computation of the total flow time and, consequently, the tardiness amount for each job. Therefore, to determine the accurate tardiness amount of each job, it is essential to extend the objective function as described in equation (3.61).

The value of ε is crucial in determining the number of tardy jobs in the objective function, as it also influences the minimization of job completion times at each position. The value of ε is determined based on the instance size being studied, ensuring that the sum of processing times for all jobs and the sum of maximum processing times for each job are proportioned by 100. The equation for ε is provided by equation (3.62):

$$\varepsilon = \frac{1}{(\sum_{i=1}^N \sum_{r=1}^M P_{ir} + \sum_{i=1}^N (i-1) * \max\{P_{1r}, P_{2r}, \dots, P_{ir} | \forall r\}) * 100}. \quad (3.62)$$

The MILP model examines the number of tardy jobs minimization in the NWPFSPs setting from the viewpoint of job positions and the starting time of each position. However, the CP model has a different perspective, yielding an essential function of the CP library, that is *allDifferent*. The CP model constructs a relationship between the jobs at each position using a binary value consideration in this function.

Model 2. The CP model		
Objective		
Minimize $\sum_{i \in N} U_i$		(3.63)
Constraints		
$allDifferent(X_1, X_2, \dots, X_n)$		(3.64)
$S_{X_{(i+1)},r} \geq S_{X_{(i)},r} + P_{X_{(i)},r}$	$\forall r \in M, \forall i \in \{1..N-1\}$	(3.65)
$S_{i,(r+1)} = S_{i,r} + P_{i,r}$	$\forall r \in \{1..M-1\}, \forall i \in N$	(3.66)
$U_i = \min\{1, \max\{(S_{i,M} + P_{i,M} - DD_i), 0\}\}$	$\forall i \in N$	(3.67)
$S_{i,r}, C_i, T_i, U_i \geq 0$	$\forall i \in N, \forall r \in M$	(3.68)

The objective function (3.63) minimizes the total number of tardy jobs. Constraint (3.64) assigns only one job to a position, and each is reserved for only one job. With this constraint, a new feature of CP, that is the *allDifferent* function is explored. Constraint (3.65) does not allow jobs to overlap. Then, constraint (3.66) ensures the no-wait restriction. Constraint (3.67) holds the value of the binary variable, which is a job's tardiness. U_i is either 1 or 0 depending on the tardiness of job i . Finally, the sign restrictions are ensured by constraint (3.68). The previous analysis of several mathematical models for the other objective functions (makespan, total flow time, and total tardiness) can be found in Yüksel et al. (2022a), and Yüksel et al. (2022b).

3.8.3. A Novel Quick Upper Bound: Sacrifice & Rearrange Heuristic

This section elaborates on the newly introduced and efficient quick upper-bounding method to minimize the number of tardy jobs on NWPFSs. The upper bound is referred to as the Sacrifice and Rearrange Heuristic (SRH), and its idea is rooted in classical improvement heuristics, like the *Variable Neighborhood Heuristic* (Hansen et al. (2010)). The Sacrifice and Rearrange Heuristic (SRH) investigates the neighborhoods of the current best solution, transitioning to a new solution only if the *Gain Function*'s conditions are satisfied. It fundamentally aims to decrease the number of tardy jobs by sacrificing the definite tardy jobs and rearranging the remaining jobs on the sequence iteratively.

Let,

$\Delta_{F_{ik}}$ = The time delay on the first machine between two consecutive jobs i and k ($i, k \in N$) (see Figures 3.1 and 3.9) and $|Tardy(\pi)|$ = The number of tardy jobs in sequence π .

First, we sequence the jobs based on the Earliest Due Date (EDD) rule (π_{EDD}). Amongst this sequence, we find and tag the first tardy job and its position ($X_p = j$). For this job and the subsequent jobs ($X_{p+1} = j + 1, \dots, n$), we calculate a *Gain Function*. The *Gain* amount is the total savings in the objective function if we move that job to the last position in the sequence. We carry the job that yields the highest positive gain to the last position. We continue this process until the *Gain* is positive. We announce the final sequence and the value of the number of tardy jobs. The motivation for the proposed SRH is the following. Once a tardy job is encountered in any position p , it will still be tardy even if we move to any subsequent positions $p + 1$, $p + 2$, $p + 3, \dots, N$. This leads to tardy jobs that are easy to move between further consequent positions. With the help of this idea, we can easily calculate the amount of *Gains* that can be attained by moving tardy jobs to subsequent positions. The computation of the *Gain Function* is the fundamental idea of the proposed heuristic. The pseudocode of the Sacrifice and Rearrange Heuristic (SRH) is provided in Algorithm 3.1.

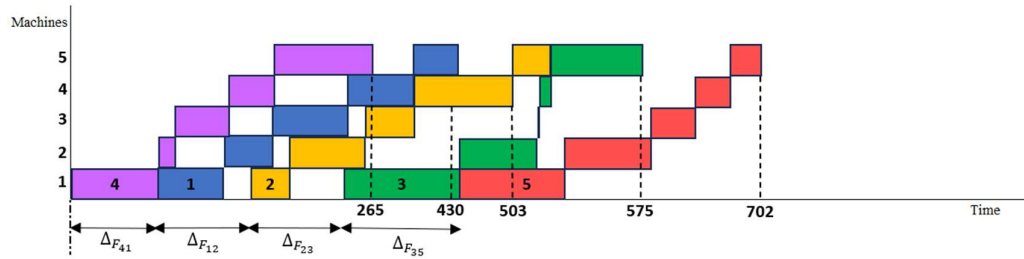
Algorithm 3.1. Sacrifice & Rearrange Heuristic

- S1.** Construct $\pi_{EDD_{S_1}} = \{1, \dots, n\} = \{1, \dots, X_{p-1} = j-1, X_p = j, X_{p+1} = j+1, \dots, n\}$
- S2.** Let j be the first tardy job that is at position p in the sequence π_{EDD} .
- S3.** Calculate C_{max} of the partial sequence $\pi_{Partial_{left}} = \{1, \dots, X_{p-1} = j-1\}$ until the first tardy job j at position p .
- S4.** Let $\pi = \{1, \dots, X_{p-1}, X_p = j, X_{p+1}, \dots, n\}$ and $\pi_{Partial_{right}} = \{X_p = j, X_{p+1}, \dots, X_k = n\}$
- Do until $k = 0$ or $Gain(j^*) = 0$
- $|Tardy(\pi_{Partial_{right}})| = k$ and
- $\pi_{Partial_{right}} = \{X_p = j, X_{p+1}, \dots, X_k = n\} = \{X_1 = j, X_2, \dots, X_k = n\}$
- Calculate **Gain Function**(X_l) = $[\Delta_{F_{l-1,l}} + \Delta_{F_{l,l+1}} - \Delta_{F_{l-1,l+1}}]^+ * (k-l) \quad \forall l =$
- $\{1, \dots, k\}$
- Let $j^* = \operatorname{argmax}_{\forall l=\{1,\dots,k\}} \{Gain(X_l)\}$ and $X_{p^*} = j^*$
- $\pi \leftarrow \pi' = \{1, \dots, X_{l-1} = j-1, \dots, X_{p^*-1}, X_{p^*+1}, \dots, X_n = j^*\}$
- $k \leftarrow k-1$
- $\pi_{Partial_{right}} \leftarrow \pi'_{Partial_{right}} \{\dots, X_{p^*-1}, X_{p^*+1}, \dots, X_{n-1}\}$
- S5.** Return $|Tardy(\pi)|$.
-

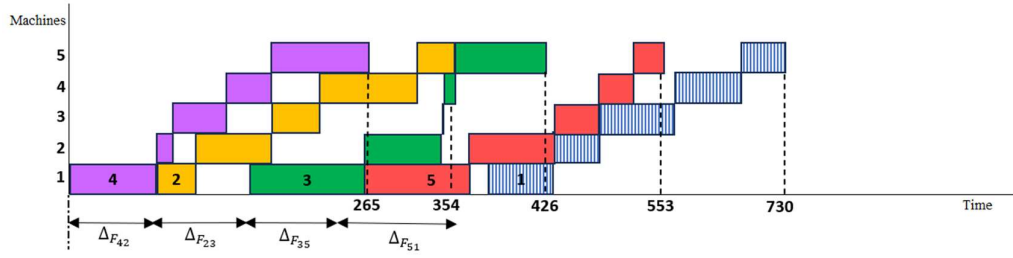
An example of SRH on $\pi_{EDD_{S_0}} = \{4, 1, 2, 3, 5\}$ is provided in Table 3.38. The SRH starts with the sequence $\pi_{EDD_{S_0}}$ and calculates the completion time and the tardiness of each job at each position p . Then, the *Gain Function* is constituted for jobs that are located after the first tardy job (the first tardy job is also included). The job with the highest *Gain* value is moved to the last position in the sequence. We iterate until there are no jobs in the *Gain Function* or the number of tardy jobs of the sequence $\pi_{Partial_{right}}$ is 0 ($|Tardy(\pi_{Partial_{right}})| = k = 0$). Gantt Charts of iterations of the example are illustrated in Figure 3.9. The jobs that are moved to the last position as definite tardy and the jobs that are left at the beginning of the sequence are illustrated with strips in Figure 3.9.

Table 3.38. Example of the Sacrifice and Rearrange Heuristic on $\pi_{EDD_{s_0}} = \{4, 1, 2, 3, 5\}$

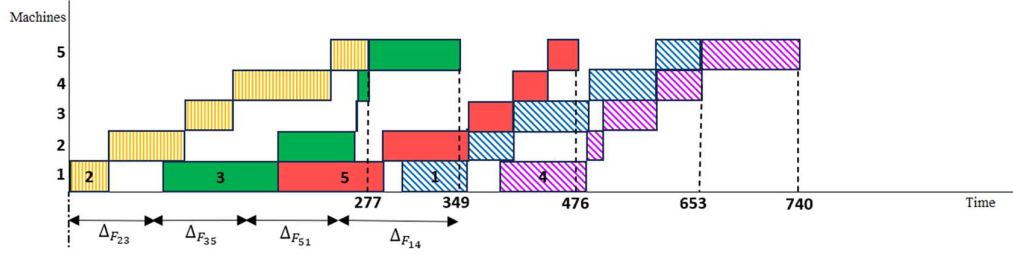
Iteration 1. Start with $\pi_{EDD_{s_0}} = \{4,1,2,3,5\}$				
		$\pi_{Partial_{right}}$	$ Tardy(\pi_{Partial_{right}}) = k$	$ Tardy(\pi) $
Sequence	{ 4, 1, 2, 3, 5 }			
CoP_p	[265,430,503,575,702]			
TP_p	[86, 116,177,239,361]	{4,1,2,3,5}	5	5
$Gain(X_p)$	[308,447, 146, 170, 0]			
Iteration 2. Move job 1 to the last position				
		$\pi_{Partial_{right}}$	$ Tardy(\pi_{Partial_{right}}) = k$	$ Tardy(\pi) $
Sequence	{ 4, 2, 3, 5, 1 }			
CoP_p	[265, 354,426,553,730]			
TP_p	[86, 28, 90,212,416]	{4,2,3,5,X}	4	5
$Gain(X_p)$	[231, 178,170, 0, X]			
Iteration 3. Move job 4 to the last position				
		$\pi_{Partial_{right}}$	$ Tardy(\pi_{Partial_{right}}) = k$	$ Tardy(\pi) $
Sequence	{ 2, 3, 5, 1, 4 }			
CoP_p	[277,349,476,653,740]			
TP_p	[0, 13,135,339,561]	{X, 3,5,X,X}	2	4
$Gain(X_p)$	[X, 170, 0, X, X]			
Iteration 4. Move job 3 to the last position				
		$\pi_{Partial_{right}}$	$ Tardy(\pi_{Partial_{right}}) = k$	$ Tardy(\pi) $
Sequence	{ 2, 5, 1, 4, 3 }			
CoP_p	[277,306,483,570,642]			
TP_p	[0, 0,169,391,306]	{X,X,X,X,X}	0	3
$Gain(X_p)$	[X, X, X, X, X]			



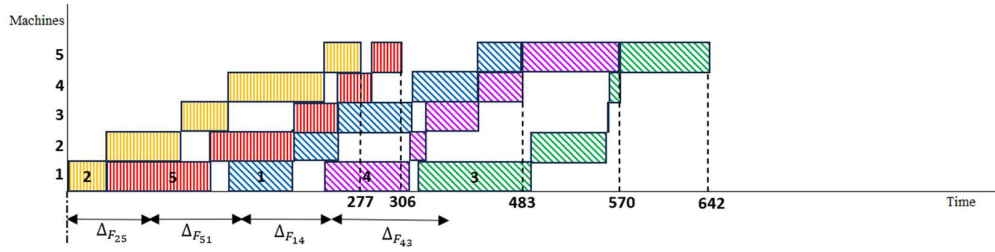
(a) Iteration 1



(b) Iteration 2



(c) Iteration 3



(d) Iteration 4

Figure 3.9. The Gantt Charts of Iterations that are provided in Table 3.20

Source: Author

Another critical feature of this SRH is the construction step's initial sequence (S1). In S1, we sequence the jobs based on the Earliest Due Date rule $\pi_{\text{EDD}_{S_0}} = \{1, \dots, n\}$, which is a classical construction heuristic (CH). However, the S1 of the algorithm can be enhanced depending on the construction strategy. We have also analyzed the pairwise differences of the due dates on the sequences constructed according to the EDD rule ($\pi_{\text{EDD}_{S_0}}$). Then, we note the three smallest differences. We swap jobs with the 1st smallest difference value, and then we obtain an alternative to the EDD CH ($\pi_{\text{EDD}_{S_1}}$). We follow this procedure for the two other most minor differences and obtain two more sequences ($\pi_{\text{EDD}_{S_2}}, \pi_{\text{EDD}_{S_3}}$). In total, we reached four various

constructed sequences. After that, we applied our *Sacrifice & Rearrange Heuristic* to these four different CHs.

An example of this swap strategy is provided below. Let us say that we have five jobs with the following due dates $DD_i = [314, 326, 336, 179, 341]$. We sort them and obtain the following sequence $\pi_{EDD_{s_0}} = \{4, 1, 2, 3, 5\}$ with the corresponding due dates $[179, 314, 326, 336, 341]$. Then, we compute the pairwise differences of due dates, as follows $[-, 135, \mathbf{12}, \mathbf{10}, \mathbf{5}]$. According to the pairwise differences, first, we swap jobs 3 and 5, since they have the smallest difference, that is 5. Then, we swap jobs 2 and 3 (the difference is 10), and jobs 1 and 2 (the difference is 12). In the end, we obtain three additional constructed sequences, as follows $\pi_{EDD_{s_1}} = \{4, 1, 2, \mathbf{5}, \mathbf{3}\}$, $\pi_{EDD_{s_2}} = \{4, 1, \mathbf{3}, \mathbf{2}, 5\}$, and $\pi_{EDD_{s_3}} = \{4, \mathbf{2}, \mathbf{1}, 3, 5\}$.

Another alternative sequence for S1 is that rather than sequencing jobs according to only the due dates, we can sequence them according to the earliest due date minus the total processing times of the job i (PT_i) values. Hereafter, that sequence can be named as $\pi_{E(DD-P)}_{s_0}$. Following the same procedure mentioned above, we can implement the pairwise difference control mechanism on this construction. Hence, we again obtain four additional constructed sequences. Let us say that we have five jobs with the following due dates and total processing times $DD_i = [314, 326, 336, 179, 341]$ and $PT_i = [353, 277, 234, 265, 270]$. Hence, the due date minus total processing time values is obtained as follows $DD_i - PT_i = [-39, 49, 102, -86, 71]$. A negative value indicates that the corresponding job is always tardy. We sort them and obtain the following sequence. $\pi_{E(DD-P)}_{s_0} = \{4, 1, 2, 5, 3\}$ with the corresponding due dates minus total processing times $[-86, -39, 49, 71, 102]$. Then, we compute the pairwise differences of due dates minus processing times $[-, \mathbf{47}, 88, \mathbf{22}, \mathbf{31}]$. According to the pairwise differences, first, we swap jobs 2 and 5, since they have the smallest difference. Then, we swap jobs 5 and 3, and we swap jobs 4 and 1. In the end, we obtain three additional constructed sequences, as follows $\pi_{E(DD-PT)}_{s_1} = \{4, 1, \mathbf{5}, \mathbf{2}, 3\}$, $\pi_{E(DD-P)}_{s_2} = \{4, 1, 2, \mathbf{3}, \mathbf{5}\}$, and $\pi_{E(DD-PT)}_{s_3} = \{\mathbf{1}, \mathbf{4}, 2, 5, 3\}$. By doing so, eight different initial sequences - 4 with EDD and 4 with E(DD-PT) - ($\pi_{EDD_{s_0}}, \pi_{EDD_{s_1}}, \pi_{EDD_{s_2}}, \pi_{EDD_{s_3}}, \pi_{E(DD-P)}_{s_0}, \pi_{E(DD-PT)}_{s_1}, \pi_{E(DD-PT)}_{s_2}, \pi_{E(DD-PT)}_{s_3}$) are achieved by CHs. Then, SRH can be implemented on these sequences. The variants of

constructive sequences that are generated from $\pi_{EDD_{s_0}}$ and $\pi_{E(DD-PT)_{s_0}}$ are obtained as depicted in Table 3.39.

Table 3.39. Generation of other variants on $\pi_{EDD_{s_0}} = \{4, 1, 2, 3, 5\}$ and

$$\pi_{E(DD-PT)_{s_0}} = \{4, 1, 2, 5, 3\}$$

$\pi_{EDD_{s_0}}$	4	1	2	3	5
DD_i	179	314	326	336	341
Pairwise differences of DD_i's	X	135	12	10	5
$\pi_{EDD_{s_1}}$	4	1	2	5	3
$\pi_{EDD_{s_2}}$	4	1	3	2	5
$\pi_{EDD_{s_3}}$	4	2	1	3	5
$\pi_{E(DD-PT)_{s_0}}$	4	1	2	5	3
$DD_i - PT_i$	-86	-39	49	71	102
Pairwise differences of $DD_i - PT_i$'s	X	47	88	22	31
$\pi_{E(DD-PT)_{s_1}}$	4	1	5	2	3
$\pi_{E(DD-PT)_{s_2}}$	4	1	2	3	5
$\pi_{E(DD-PT)_{s_3}}$	1	4	2	5	3

The Sacrifice and Rearrange Heuristic is applied to the remaining constructive sequences ($\pi_{EDD_{s_1}}$, $\pi_{EDD_{s_2}}$, $\pi_{EDD_{s_3}}$, $\pi_{E(DD-P)_{s_0}}$, $\pi_{E(DD-PT)_{s_1}}$, $\pi_{E(DD-PT)_{s_2}}$, $\pi_{E(DD-PT)_{s_3}}$) in the same way, it is explained in Table 3.39 and Figure 3.9, and the outcomes are reported in Table 3.40.

Table 3.40. Sacrifice and Rearrange Heuristics on Other Sequences

	Initial Sequence	Initial Number of Tardy Jobs	Sequence after SRH	Number of Tardy Jobs after SRH
$\pi_{EDD_{s_0}}$	{4,1,2,3,5}	5	{2,5,1,4,3}	3
$\pi_{EDD_{s_1}}$	{4,1,2,5,3}	5	{2,5,1,4,3}	3
$\pi_{EDD_{s_2}}$	{4,1,3,2,5}	5	{3,1,4,2,5}	4
$\pi_{EDD_{s_3}}$	{4,2,1,3,5}	5	{2,5,4,1,3}	3
$\pi_{E(DD-PT)_{s_0}}$	{4,1,2,5,3}	5	{2,5,1,4,3}	3
$\pi_{E(DD-PT)_{s_1}}$	{4,1,5,2,3}	5	{2,1,5,4,3}	4
$\pi_{E(DD-PT)_{s_2}}$	{4,1,2,3,5}	5	{2,5,1,4,3}	3
$\pi_{E(DD-PT)_{s_3}}$	{1,4,2,5,3}	5	{2,5,1,4,3}	3

Finally, we end up with the minimum number of tardy jobs value, which is 3, and report it as the outcome of the SRH.

3.8.4. A New Due Date Generation Mechanism and Benchmarking

Due dates are critical parameters in optimizing lateness-related performance measures, such as total tardiness, number of tardy jobs, earliness, weighted tardiness, etc. In scheduling research, it's a widespread approach to randomly create processing times by employing a uniform distribution within the range of 1 to 99. Similarly, due dates are generated randomly, following a uniform distribution by the provided formula ($\bowtie$). Moreover, various studies carry out various variants of the formula ($\bowtie$). The examples and issues were presented in Section 3.8.1. are

A new due date generation mechanism is created to overcome the aforementioned problems, as shown in Table 3.37 (Section 3.8.1). It is an advantageous mechanism for producing high-quality due dates for the NWPFSs. This mechanism also utilizes the *Sacrifice and Rearrange Heuristic*, yielding practical and challenging due dates. We use the SRH in the due date generation process to generate reasonable due dates for jobs. Hence, the quick upper bound is a part of the due date generation mechanism. The procedure is currently carried out on the first 240 instances of the VRF instance set (Vallada et al. (2015)), called small VRF instances. This dataset is used because it is one of the most used datasets in flowshop scheduling problems under the makespan

and total flow time criterion. However, to our knowledge, no due date is generated for this dataset in the literature. Thus, the proposed due date generation mechanism is carried out on the VRF dataset.

The due date generation mechanism is as follows. The VRF instance set consists of 240 instances with 24 classes of 10 jobs. We generate 100 random sequences and select the smallest makespan value. First, we assign the completion time of jobs in the sequence as their due dates ($a = 1$). Then, we apply the SRH mentioned in Section 3.8.3 to this sequence and compute the number of tardy jobs. If the average number of tardy jobs the SRH produces in each class is less than half the number of jobs in the instance class we are working on, we continue with the algorithm. Then, we decrease the value of the a 10%, which means that the due dates are cropped by an additional 10%, and apply the SRH again. If the average number of tardy jobs that the SRH produces in each class is less than half of the jobs in the instance set we are working on, we continue with the algorithm. Then, we crop the due dates by 10%, and hence, we have cropped 20% compared to the beginning. By controlling this, we will crop the due dates by 50%. In the end, if we generate the due dates with completion times cropped by a certain percentage, the average number of tardy jobs that the SRH produces is close to half the average number of jobs in each class. In this way, we are generating the most challenging instances. The pseudocode of the proposed due date generation mechanism is provided in Algorithm 3.2.

Algorithm 3.2. Due Date Generation Mechanism based on Sacrifice & Rearrange Heuristic

S0. There are 10 instances with job sizes 10,20,30,40,50,60 in the VRF set of instances. In total, 240 instances exist.

S1. Generate 100 random sequences as $\pi_1, \pi_2, \dots, \pi_{100}$ and calculate C_{max} values for each instance.

S2. Let $\pi^* = \operatorname{argmin}_{\pi \in \{\pi_1, \dots, \pi_{100}\}} \{C_{max}(\pi)\}$.

S3. Do until *average number of tardy jobs* $\cong \frac{\sum(10 * \text{JobSize})}{2}$

$a = 1$

$DD_{X_p=i} = C_{X_p=i} * a \quad \forall p \text{ in } \pi^*.$

Apply the **Sacrifice and Rearrange Heuristic** on π^*

Calculate *average number of tardy jobs* $= \frac{\sum |Tardy(\pi)|}{240}$

If *average number of tardy jobs* $< \frac{\sum(10*JobSize)}{2}$

$a \leftarrow a * \%10$

Else

Stop and return a .

S4. Return $DD_{X_p=i}C_{X_p=i} * a$ values with the a , which is obtained in S3.

3.8.5. Computational Analysis on Number of Tardy Jobs Minimization in NWPFSPs

The performed computational study is two-fold. First, tight due dates are generated with the proposed due date generation mechanism for the Vallada et al. (2015) benchmark problems. Then, the proposed mathematical models and the Sacrifice and Rearrange Heuristic are carried out on both the *Taillard, E. (1993) and Vallada et al. (2015)* benchmark sets.

To generate due dates, first, the value of the a should be decided. An experiment is performed, and the results are reported in Table 3.41.

Table 3.41. The Summary Table of Experimentation on a Value

	$a = 0.40$		$a = 0.50$		$a = 0.60$	
Instance Set	$\sum_{i \in N} U_i$	Job Size	$\sum_{i \in N} U_i$	Job Size	$\sum_{i \in N} U_i$	Job Size
10 jobs Avg.	8.2	10	7.2	10	6.1	10
20 jobs Avg.	14.1	20	11.6	20	9.3	20
30 jobs Avg.	19.3	30	18.4	30	12.6	30
40 jobs Avg.	24.3	40	19.7	40	15.4	40
50 jobs Avg.	29.5	50	23.8	50	18.7	50
60 jobs Avg.	34.3	60	27.4	60	21.3	60
Overall Average	21.6	35	18.0	35	13.9	35
Percent	62%		51%		40%	

According to our experimentation on the a value expressed in Table 3.41, we obtained 62%, 51%, and 40% of tardy jobs on average based on the corresponding a values of 0.40, 0.50, and 0.60, respectively. Since 51% is the closest to 50%, 0.50 is chosen as the a value. By setting a to 0.50, the due dates are generated, assuming that there will be approximately 50% tardy jobs when the set is employed for the number of tardy

jobs minimization in NWPFSF. Generated due dates of the problem instances are presented to the literature and provided in Appendices A1-A6 for 10, 20, 30, 40, 50, and 60 jobs, respectively. The text files containing instance details with the associated due dates of the benchmark set are provided at the following link: <https://data.mendeley.com/datasets/nfjs94x9wj/1>. The generated due dates are tight, yielding approximately 50% tardy jobs, so they can also be carried out for other lateness-related performance measurements in further studies. Namely, we provide a rigorous and challenging benchmark problem set characterized by tough due dates in the literature of minimizing lateness-related performance measures in NWPFSFs.

The computational analysis is conducted extensively to assess the performance of the MILP & CP model formulations and quick upper bound algorithms. The Taillard, E. (1993) and Vallada et al. (2015) instance sets are used as benchmark problems.

- ***The first benchmark problem*** - that is Taillard (1993) - comprises 11 collections of problems in which each collection has 10 instances and varying sizes of 20, 50, 100, and 200 jobs with 5, 10, and 20 machines. In addition to the above-mentioned instances, three more sets of small instances (30 instances in total) are carried out. These small instances, with sizes of 5-5, 5-10, and 5-20, are cropped from the Taillard set of 20-5, 20-10, and 20-20 instances by a sound truncation mechanism proposed by (Yüksel, 2019; Yüksel et al. 2020), which means that we carry out 140 problem instances of Taillard (1993) through this study.
- ***The second benchmark problem*** - Vallada et al. (2015) - comprises 24 small problems in which each collection has 10 instances and varying sizes of 10, 20, 30, 40, 50, and 60 jobs with 5, 10, 15, and 20 machines. Hence, 240 problem instances by the Vallada et al. (2015) are followed up through this study. Hereafter, Vallada et al. (2015) benchmark set will be called the VRF instances.

The comprehensive results, encompassing the text files containing instance details with the associated due dates, mathematical model outputs, and SRH outputs are provided in the following link: <https://data.mendeley.com/datasets/nfjs94x9wj/1>.

First, both benchmark problem sets are employed to execute the MILP and CP model formulations instance by instance. The experimentation is performed in the IBM ILOG CPLEX Optimization Studio 20.0. The experimentation of the optimality is presented on the cropped instances of the Taillard set, and the first 40 instances of the VRF set. The results are presented in Tables 3.42 and 3.43, respectively. The objective values, the average run time, and the average gap percentage per instance set are provided in columns. In contrast, the total number of tardy jobs, the overall average run time, and the overall average gap percentage are provided in the last row of the tables.

Table 3.42. Computational Analysis for MILP and CP on the Cropped Instances of Taillard Instances

Model 1 - MILP					Model 2 - CP				
Instance Set	Objective Value (Sum)	Time (Sec) (Average)	Gap % (Average)		Instance Set	Objective Value (Sum)	Time (Sec) (Average)	Gap % (Average)	
5-5	40	0.038	0%		5-5	40	0.038	0%	
5-10	15	0.063	0%		5-10	15	0.076	0%	
5-20	11	0.094	0%		5-20	11	0.131	0%	
Sum	66	Avg	0.065	Avg 0%	Sum	66	Avg	0.082	Avg 0%

Table 3.43. Computational Analysis for MILP and CP on the first 4 sets of VRF Instances

Model 1 - MILP					Model 2 - CP				
Instance Set	Objective Value (Sum)	Time (Sec) (Average)	Gap % (Average)		Instance Set	Objective Value (Sum)	Time (Sec) (Average)	Gap % (Average)	
10-5	58	0.612	0%		10-5	58	9.908	0%	
10-10	64	1.047	0%		10-10	64	45.072	0%	
10-15	72	1.320	0%		10-15	72	40.580	0%	
10-20	82	1.586	0%		10-20	82	27.257	0%	
Sum	276	Avg	1.141	Avg 0%	Sum	276	Avg	30.704	Avg 0%

Tables 3.42 and 3.43 demonstrate the verification of the mathematical models on the cropped instances of the Taillard set (30 instances), and the first 40 instances of the

VRF set. Both models reach the same objective value and have a 0% gap value on the cropped instances of the Taillard set and the first 40 instances of the VRF set, respectively.

After illustrating the verification of the mathematical models, the MILP and CP model formulations are executed instance-by-instance by a 3600-second time limit in both benchmark problem sets. The experimentation of the MILP and CP models in all instances of the Taillard and VRF benchmark sets are presented in Tables 3.44 and 3.45, respectively. The objective values, the average run time, and the average gap percentage per instance set are provided in the columns. In contrast, the last row of the tables provides the total number of tardy jobs, the overall average run time, and the overall average gap percentage.

Table 3.44. Computational Analysis for MILP and CP on the Taillard Instances

Model 1 - MILP				Model 2 - CP			
Instance Set	Objective Value (Sum)	Time (Sec) (Average)	Gap % (Average)	Instance Set	Objective Value (Sum)	Time (Sec) (Average)	Gap % (Average)
5-5	40	0.038	0%	5-5	40	0.038	0%
5-10	15	0.063	0%	5-10	15	0.076	0%
5-20	11	0.094	0%	5-20	11	0.131	0%
20-5	77	2792	7.98%	20-5	78	3600	100.00%
20-10	34	3600	85.11%	20-10	34	3600	100.00%
20-20	20	3600	75.91%	20-20	16	2788	70.00%
50-5	340	3600	44.51%	50-5	361	3600	100.00%
50-10	323	3600	73.36%	50-10	327	3600	100.00%
50-20	305	3600	99.42%	50-20	282	3600	100.00%
100-5	868	3600	86.10%	100-5	867	3600	100.00%
100-10	891	3600	97.59%	100-10	831	3600	100.00%
100-20	927	3600	99.77%	100-20	811	3600	100.00%
200-10	1971	3600	96.26%	200-10	1889	3600	100.00%
200-20	1964	3600	99.68%	200-20	1871	3600	100.00%
Sum	7786	Avg 2770.89	Avg 61.83%	Sum	7433	Avg 2770.60	Avg 76.43%

Table 3.45. Computational Analysis for MILP and CP on the Small VRF Instances

Model 1 - MILP				Model 2 - CP			
Instance Set	Objective Value (Sum)	Time (Sec) (Average)	Gap % (Average)	Instance Set	Objective Value (Sum)	Time (Sec) (Average)	Gap % (Average)
10-5	58	0.612	0%	10-5	58	9.908	0%
10-10	64	1.047	0%	10-10	64	45.072	0%
10-15	72	1.320	0%	10-15	72	40.580	0%
10-20	82	1.586	0%	10-20	82	27.257	0%
20-5	87	2368	4.17%	20-5	87	3600	55.92%
20-10	104	3468	15.96%	20-10	101	3600	45.65%
20-15	108	1849	3.82%	20-15	108	3600	35.39%
20-20	110	2785	10.86%	20-20	111	3600	31.86%
30-5	126	3600	38.43%	30-5	136	3600	71.10%
30-10	140	3600	56.58%	30-10	145	3600	61.95%
30-15	142	3600	51.94%	30-15	149	3600	54.80%
30-20	152	3600	51.87%	30-20	155	3600	54.71%
40-5	169	3600	54.41%	40-5	187	3600	78.39%
40-10	180	3600	74.92%	40-10	195	3600	77.30%
40-15	187	3600	66.97%	40-15	203	3600	70.03%
40-20	190	3600	60.83%	40-20	208	3600	64.91%
50-5	212	3600	67.75%	50-5	237	3600	85.26%
50-10	226	3600	77.65%	50-10	247	3600	80.20%
50-15	241	3600	75.19%	50-15	263	3600	77.16%
50-20	264	3600	71.25%	50-20	263	3600	71.23%
60-5	252	3600	74.05%	60-5	284	3600	88.04%
60-10	283	3600	79.84%	60-10	303	3600	81.20%
60-15	322	3600	81.58%	60-15	307	3600	80.78%
60-20	332	3600	78.66%	60-20	323	3600	78.29%
Sum	4103	Avg 2836	Avg 45.70%	Sum	4288	Avg 3005.12	Avg 55.96%

As seen in Table 3.44, the CP model results in better solutions than the MILP model in terms of the value of the objective function. In contrast, the MILP model reports a lower gap percentage level than the CP model on the Taillard benchmark set. Furthermore, the average run time of both models follows a similar trend to that of the Taillard benchmark set. The MILP model also reaches optimal solutions in the following instances: 20-5-03, 20-5-05, 20-5-06, and 20-5-08, whereas the CP model

reaches 20-20-01 20-20-03, and 20-20-10, other than the instances which they have found optimally in the cropped instances of Taillard.

Moving to Table 3.45, the MILP model results in better solutions than the CP model regarding the objective function and the gap percentage level on the VRF benchmark set. Furthermore, the average run time of both models follows a similar trend with only a 169-second difference on the VRF benchmark set. The MILP model also reaches optimal solutions in the following instances: 20-5-01 20-5-02, 20-5-03, 20-5-05, 20-5-06, 20-5-09, 20-5-10, 20-10-02, 20-10-08, 20-15-01, 20-15-04, 20-15-5, 20-15-06, 20-15-07, 20-15-08, 20-15-09, 20-20-02, 20-20-03, 20-20-06 and 20-20-09, other than the instances which they have found optimally in the first four sets of instances of VRF. However, the CP cannot find an optimal result other than the instances they have found optimally in the first four sets of instances of the VRF benchmark set.

After presenting the experimentation on modeling, the performance of the SRH can be illustrated as follows. The experimentation of the SRH is presented on the cropped instances of the Taillard set (30 instances), and the first 40 instances of the VRF set. The results are presented in Table 3.46, and Tables 3.47-3.48, respectively. The total objective values and the total run time per instance set are provided in columns, whereas the overall sum of the number of tardy jobs and the overall sum of the run times are provided in the last row of the tables.

Table 3.46. Computational Analysis for MILP, CP, and SRH on the Cropped Instances of Taillard Instances

	Model 1 - MILP		Model 2 - CP		SRH	
Instance Set	Objective Value (Sum)	Time (in Sec) (Sum)	Objective Value (Sum)	Time (in Sec) (Sum)	Objective Value (Sum)	Time (in Sec) (Sum)
5-5-01	3	0.05	3	0.04	3	0.02
5-5-02	3	0.05	3	0.04	3	0.02
5-5-03	5	0.03	5	0.03	5	0.03
5-5-04	3	0.02	3	0.04	3	0.02
5-5-05	3	0.08	3	0.04	3	0.02
5-5-06	5	0.02	5	0.02	5	0.01
5-5-07	4	0.03	4	0.07	4	0.02
5-5-08	5	0.06	5	0.01	5	0.03
5-5-09	4	0.03	4	0.07	4	0.02
5-5-10	5	0.02	5	0.02	5	0.03
5-5 Sum	40	0.38	40	0.38	40	0.22
5-10-01	1	0.09	1	0.08	1	0.17
5-10-02	1	0.05	1	0.07	1	0.02
5-10-03	1	0.05	1	0.05	1	0.02
5-10-04	1	0.06	1	0.06	1	0.02
5-10-05	1	0.03	1	0.06	1	0.02
5-10-06	2	0.09	2	0.07	3	0.02
5-10-07	1	0.08	1	0.07	1	0.03
5-10-08	1	0.03	1	0.09	2	0.04
5-10-09	3	0.08	3	0.12	3	0.02
5-10-10	3	0.06	3	0.10	3	0.02
5-10 Sum	15	0.63	15	0.76	17	0.37
5-20-01	1	0.11	1	0.13	1	0.03
5-20-02	1	0.05	1	0.13	1	0.05
5-20-03	1	0.08	1	0.15	1	0.02
5-20-04	0	0.03	0	0.10	0	0.02
5-20-05	2	0.08	2	0.20	2	0.03
5-20-06	2	0.05	2	0.14	2	0.03
5-20-07	1	0.17	1	0.11	1	0.04
5-20-08	1	0.22	1	0.12	1	0.02
5-20-09	1	0.08	1	0.11	1	0.03
5-20-10	1	0.08	1	0.11	1	0.02
5-20 Sum	11	0.94	11	1.31	11	0.30
Overall Sum	66	1.94	66	2.45	68	0.89

Table 3.47. Computational Analysis for MILP, CP, and SRH on the first 4 sets of VRF Instances

	Model 1 - MILP		Model 2 - CP		SRH	
Instance Set	Objective Value (Sum)	Time (in Sec) (Sum)	Objective Value (Sum)	Time (in Sec) (Sum)	Objective Value (Sum)	Time (in Sec) (Sum)
10-5-01	5	0.28	5	2.62	5	0.06
10-5-02	6	0.86	6	5.93	6	0.05
10-5-03	7	0.74	7	9.49	7	0.05
10-5-04	5	1.11	5	15.95	5	0.05
10-5-05	6	0.09	6	0.08	6	0.07
10-5-06	6	0.75	6	7.28	6	0.07
10-5-07	5	0.28	5	0.09	5	0.05
10-5-08	6	0.61	6	4.17	6	0.09
10-5-09	6	0.67	6	6.13	6	0.06
10-5-10	6	0.74	6	47.36	6	0.05
10-5 Sum	58	6.12	58	99.08	59	0.61
10-10-01	7	1.31	7	77.26	7	0.05
10-10-02	5	1.03	5	12.61	5	0.06
10-10-03	8	0.66	8	26.76	8	0.06
10-10-04	7	1.42	7	27.44	7	0.06
10-10-05	7	0.63	7	0.39	8	0.07
10-10-06	7	0.73	7	25.59	7	0.05
10-10-07	6	0.63	6	243.26	6	0.06
10-10-08	6	2.41	6	10.67	6	0.05
10-10-09	6	0.27	6	0.15	6	0.06
10-10-10	5	1.39	5	26.58	6	0.05
10-10 Sum	64	10.47	64	450.72	66	0.57

Table 3.48. Computational Analysis for MILP, CP, and SRH on the first 4 sets of VRF Instances

	Model 1 - MILP		Model 2 - CP		SRH	
Instance Set	Objective Value (Sum)	Time (in Sec) (Sum)	Objective Value (Sum)	Time (in Sec) (Sum)	Objective Value (Sum)	Time (in Sec) (Sum)
10-20-01	7	0.47	7	19.07	7	0.04
10-20-02	7	0.70	7	0.28	7	0.06
10-20-03	6	3.84	6	11.61	6	0.06
10-20-04	7	0.75	7	8.43	7	0.06
10-20-05	8	0.44	8	0.18	8	0.06
10-20-06	7	2.30	7	120.22	7	0.06
10-20-07	7	2.34	7	0.21	7	0.04
10-20-08	7	0.78	7	56.00	7	0.06
10-20-09	8	1.03	8	59.45	8	0.06
10-20-10	8	0.55	8	0.34	8	0.05
10-15 Sum	72	13.20	72	405.80	72	0.55
10-20-01	7	3.00	7	62.50	7	0.04
10-20-02	9	0.42	9	0.21	9	0.09
10-20-03	8	2.63	8	148.38	8	0.04
10-20-04	9	0.69	9	0.19	9	0.06
10-20-05	7	0.88	7	60.19	7	0.07
10-20-06	8	4.23	8	0.22	8	0.06
10-20-07	8	1.31	8	0.22	8	0.06
10-20-08	7	2.23	7	0.32	7	0.06
10-20-09	9	0.28	9	0.21	9	0.07
10-20-10	10	0.19	10	0.14	10	0.05
10-20 Sum	82	15.86	82	272.57	82	0.61
Overall Sum	276	45.66	276	1228.17	279	2.34

The SRH can reach optimal solutions in 28 out of 30 instances in the Taillard benchmark set as illustrated in Table 3.46. The instances in which SRH can not find

optimal solutions are 5-10-06 and 5-10-08. However, the number of tardy jobs value is only 1 unit higher than the optimal value in those instances. The total run time needed to find optimal solutions for MILP, CP, and SRH are 1.94, 2.45, and 0.89 seconds in the Taillard set.

The SRH can reach optimal solutions in 37 out of 40 instances in the VRF benchmark set, as illustrated in Tables 3.47 and 3.48. The instances in which SRH cannot find optimal solutions are 10-5-06, 10-10-05, and 10-10-10. However, the number of tardy jobs value is only 1 unit higher than the optimal value in those instances. In addition, the total run time needed to find optimal solutions for MILP, CP, and SRH are 45.66, 1228.17, and 2.34 seconds in the VRF set.

Following this, the performance of the quick upper bound, reached by the SRH, has been experimented with over different constructive heuristics. The SRH improves the four constructive heuristics with EDD, and the SRH improves the other four constructive heuristics with E(DD-PT). The results are reported in Tables 3.49 and 3.50 on the Taillard benchmark set.

Table 3.49. Performance of Sacrifice & Rearrange Heuristic with EDD on Taillard Benchmark Set

Sum over all instances in the set	$\pi_{EDD_{s_0}}$			$\pi_{EDD_{s_1}}$			$\pi_{EDD_{s_3}}$			$\pi_{EDD_{s_4}}$		
	$\pi_{EDD_{s_0}}$	SRH on $\pi_{EDD_{s_0}}$	Improvement in Objective Function	$\pi_{EDD_{s_1}}$	SRH on $\pi_{EDD_{s_1}}$	Improvement in Objective Function	$\pi_{EDD_{s_2}}$	SRH on $\pi_{EDD_{s_2}}$	Improvement in Objective Function	$\pi_{EDD_{s_3}}$	SRH on $\pi_{EDD_{s_3}}$	Improvement in Objective Function
5-5	50	40	10	50	41	9	50	42	8	50	40	10
5-10	30	17	13	29	18	11	26	18	8	37	20	17
5-20	23	17	6	28	17	11	27	18	9	29	18	11
20-5	179	94	85	179	98	81	179	94	85	179	87	81
20-10	151	53	98	151	53	98	149	55	94	150	54	96
20-20	110	32	78	113	34	79	118	34	84	113	31	82
50-5	482	354	128	482	355	127	482	355	127	482	354	128
50-10	480	316	164	481	317	164	480	317	163	480	317	163
50-20	485	263	222	485	263	222	485	262	223	485	263	222
100-5	985	834	151	985	836	149	985	835	150	985	836	149
100-10	984	789	195	984	789	195	984	791	193	984	787	197
100-20	983	723	260	983	722	261	983	722	261	984	723	261
200-10	1987	1769	218	1988	1770	218	1987	1769	218	1987	1770	217
200-20	1989	1707	282	1989	1703	286	1989	1707	282	1989	1708	281
Overall Sum	8918	7008	1910	8927	7016	1911	8924	7019	1905	8934	7019	1915
Improvement in Objective Function	21.42 %			21.41 %			21.35 %			21.43 %		

Table 3.50. Performance of Sacrifice & Rearrange Heuristic with E(DD-PT) on
Taillard Benchmark Set

Sum over all instances in the set	$\pi_{E(DD-PT)_{s_0}}$			$\pi_{E(DD-PT)_{s_1}}$			$\pi_{E(DD-PT)_{s_3}}$			$\pi_{E(DD-PT)_{s_4}}$		
	$\pi_{E(DD-PT)_{s_0}}$ SRH on	$\pi_{E(DD-PT)_{s_0}}$ Improvement in	Objective Function	$\pi_{E(DD-PT)_{s_1}}$ SRH on	$\pi_{E(DD-PT)_{s_1}}$ Improvement in	Objective Function	$\pi_{E(DD-PT)_{s_2}}$ SRH on	$\pi_{E(DD-PT)_{s_2}}$ Improvement in	Objective Function	$\pi_{E(DD-PT)_{s_3}}$ SRH on	$\pi_{E(DD-PT)_{s_3}}$ Improvement in	Objective Function
5-5	50	40	10	50	42	8	50	40	10	50	40	10
5-10	32	18	14	36	23	13	33	19	14	33	19	14
5-20	24	16	8	28	16	12	36	19	17	29	20	9
20-5	184	105	79	185	106	79	184	104	80	184	105	79
20-10	164	64	100	161	63	98	163	65	98	163	63	100
20-20	124	39	85	119	40	79	125	41	84	125	40	85
50-5	490	377	113	490	375	115	490	375	115	490	381	109
50-10	487	327	160	487	326	161	487	325	162	487	328	159
50-20	489	274	215	489	274	215	489	274	215	489	275	214
100-5	990	856	134	989	858	131	990	856	134	990	862	128
100-10	989	803	186	989	804	185	989	803	186	989	803	186
100-20	985	732	253	985	732	253	985	732	253	985	732	253
200-10	1990	1783	207	1990	1786	204	1990	1784	206	1990	1783	207
200-20	1989	1713	276	1989	1713	276	1989	1712	277	1989	1714	275
Overall Sum	8987	7147	1840	8987	7158	1829	9000	7149	1851	8993	7165	1828
Improvement in Objective Function	20.47 %			20.35 %			20.57 %			20.33 %		

According to Table 3.49, the SRH reached 21.42%, 21.41%, 21.35%, and 21.43% improvements in the $\pi_{EDD_{s_0}}$, $\pi_{EDD_{s_1}}$, $\pi_{EDD_{s_2}}$, and $\pi_{EDD_{s_3}}$ sequences of the Taillard benchmark problems, respectively. In addition, the SRH reached 20.47%, 20.35%, 20.57%, and 20.33% improvements on the $\pi_{E(DD-PT)_{s_0}}$, $\pi_{E(DD-PT)_{s_1}}$, $\pi_{E(DD-PT)_{s_2}}$, and $\pi_{E(DD-PT)_{s_3}}$ sequences of the Taillard benchmark problems, respectively, are reported in Table 3.50.

Then, the performance of the quick upper bound, reached by the SRH, was experimented with using different constructive heuristics on the VRF benchmark set.

The SRH improves the four constructive heuristics with EDD and the other four constructive heuristics with E(DD-PT) are improved by SRH. The results are reported in Tables 3.51 and 3.52 on the VRF benchmark set.

Table 3.51. Performance of Sacrifice & Rearrange Heuristic with EDD on VRF Benchmark Set

Sum over all instances in the set	$\pi_{EDD_{s_0}}$			$\pi_{EDD_{s_1}}$			$\pi_{EDD_{s_3}}$			$\pi_{EDD_{s_4}}$		
	$\pi_{EDD_{s_0}}$	SRH on $\pi_{EDD_{s_0}}$	Improvement in Objective Function	$\pi_{EDD_{s_1}}$	SRH on $\pi_{EDD_{s_1}}$	Improvement in Objective Function	$\pi_{EDD_{s_2}}$	SRH on $\pi_{EDD_{s_2}}$	Improvement in Objective Function	$\pi_{EDD_{s_3}}$	SRH on $\pi_{EDD_{s_3}}$	Improvement in Objective Function
10-5	100	63	37	100	64	36	100	62	38	100	65	35
10-10	100	69	31	100	72	28	100	70	30	100	70	30
10-15	100	74	26	100	74	26	100	76	24	100	75	25
10-20	100	82	18	100	83	17	100	85	15	100	83	17
20-5	200	103	97	200	107	93	200	104	96	200	103	97
20-10	200	113	87	200	113	87	200	114	86	200	115	85
20-1	200	125	75	200	127	73	200	124	76	200	125	75
20-20	200	124	76	200	127	73	200	127	73	200	125	75
30-5	300	145	155	300	146	154	300	149	151	300	146	154
30-10	300	159	141	300	161	139	300	159	141	300	161	139
30-15	300	167	133	300	168	132	300	166	134	300	166	134
30-20	300	163	137	300	164	136	300	164	136	300	163	137
40-5	400	188	212	400	191	209	400	190	210	400	187	213
40-10	400	193	207	400	196	204	400	195	205	400	192	208
40-15	400	201	199	400	203	197	400	200	200	400	202	198
40-20	400	205	195	400	205	195	400	207	193	400	207	193
50-5	500	228	272	200	226	274	500	228	272	500	227	273
50-10	500	234	266	500	237	263	500	236	264	500	234	266
50-15	500	248	252	500	243	257	500	246	254	500	248	252
50-20	500	243	257	500	244	256	500	246	254	500	244	256
60-5	600	267	333	600	270	330	600	265	335	600	268	332
60-10	600	272	328	600	274	326	600	273	327	600	274	326
60-15	600	279	321	600	277	323	600	281	319	600	281	319
60-20	600	277	323	600	277	323	600	279	321	600	277	323
Overall Sum	8400	4222	4178	8400	4249	4151	8400	4246	4154	8400	4238	4162
Improvement in Objective Function	49.74 %			49.42 %			49.45 %			49.55 %		

Table 3.52. Performance of Sacrifice & Rearrange Heuristic on the variants of Construction Heuristic with E(DD-PT) on VRF Benchmark Set

Sum over all instances in the set	$\pi_{E(DD-PT)s_0}$			$\pi_{E(DD-PT)s_1}$			$\pi_{E(DD-PT)s_3}$			$\pi_{E(DD-PT)s_4}$		
	$\pi_{E(DD-PT)s_0}$	SRH on $\pi_{E(DD-PT)s_0}$	Improvement in Objective	$\pi_{E(DD-PT)s_1}$	SRH on $\pi_{E(DD-PT)s_1}$	Improvement in Objective	$\pi_{E(DD-PT)s_2}$	SRH on $\pi_{E(DD-PT)s_2}$	Improvement in Objective	$\pi_{E(DD-PT)s_3}$	SRH on $\pi_{E(DD-PT)s_3}$	Improvement in Objective
10-5	100	64	36	100	65	35	100	65	35	100	64	36
10-10	100	68	32	100	70	30	100	67	33	100	70	30
10-15	100	74	26	100	75	25	100	74	26	100	74	26
10-20	100	84	16	100	83	17	100	83	17	100	85	15
20-5	200	105	95	200	105	95	200	107	93	200	105	95
20-10	200	118	82	200	120	80	200	116	84	200	116	84
20-15	200	130	70	200	126	74	200	128	72	200	128	72
20-20	200	127	73	200	126	74	200	128	72	200	126	74
30-5	300	151	149	300	153	147	300	152	148	300	151	149
30-10	300	161	139	300	163	137	300	162	138	300	161	139
30-15	300	166	134	300	165	135	300	165	135	300	165	135
30-20	300	165	135	300	164	136	300	168	132	300	166	134
40-5	400	194	206	400	194	206	400	195	205	400	195	205
40-10	400	199	201	400	198	202	400	200	200	400	200	200
40-15	400	202	195	400	202	195	400	203	197	400	202	198
40-20	400	211	189	400	210	190	400	210	190	400	209	191
50-5	500	237	263	500	235	265	500	237	263	500	236	264
50-10	500	243	257	500	243	257	500	243	257	500	241	259
50-15	500	248	252	500	251	249	500	249	251	500	255	245
50-20	500	248	252	500	248	252	500	247	253	500	248	252
60-5	600	275	325	600	273	327	600	272	328	600	275	325
60-10	600	284	319	600	280	320	600	278	322	600	279	321
60-15	600	286	314	600	286	314	600	289	311	600	285	315
60-20	600	283	317	600	285	315	600	283	317	600	285	315
Overall Sum	8400	4320	4080	8400	4320	4080	8400	4321	4079	8400	4321	4079
Improvement in Objective Function	48.57 %			48.57 %			48.56 %			48.56 %		

According to Table 3.51, the SRH reached 49.74%, 49.42%, 49.45%, and 49.55% improvements in the $\pi_{\text{EDD}_{s_0}}$, $\pi_{\text{EDD}_{s_1}}$, $\pi_{\text{EDD}_{s_2}}$, and $\pi_{\text{EDD}_{s_3}}$ sequences of the VRF benchmark problems, respectively. In addition, the SRH reached 48.57%, 48.57%, 48.56%, and 48.56% improvements on the $\pi_{\text{E}(\text{DD-P})_{s_0}}$, $\pi_{\text{E}(\text{DD-PT})_{s_1}}$, $\pi_{\text{E}(\text{DD-P})_{s_2}}$, and $\pi_{\text{E}(\text{DD-P})_{s_3}}$ sequences of the VRF benchmark problems, respectively, are reported in Table 3.52. Therefore, Tables 3.49-3.52 illustrate the outstanding performance of the SRH on different constructions. One upcoming inference of this experiment is that the performance of the SRH on VRF benchmark problems is relatively higher than that of the SRH on the Taillard benchmark problems achieved by the decent-quality due dates. Our study shows that, on average, it leads to a 20.91% improvement compared to the construction heuristics when applied to the Taillard benchmark problems. However, notably, it achieves an average enhancement of 49.05% over the construction heuristics on the VRF benchmark problems.

To conclude both experimentations on the models and the quick upper bound, the outcomes of the MILP and CP models were compared with the outcomes of the Sacrifice & Rearrange Heuristic (minimum of all variants of Sacrifice & Rearrange Heuristic). Tables 3.53 and 3.54 report the results of the Taillard and VRF benchmark problems in terms of the objective function value and the computation time in seconds, respectively.

Table 3.53. Comparison of Sacrifice & Rearrange Heuristic with the Mathematical Models on Taillard Benchmark Set

Instance Set	Model 1 - MILP		Model 2 – CP		SRH	
	Objective Value (Sum)	Time (in Secs) (Sum)	Objective Value (Sum)	Time (in Secs) (Sum)	Objective Value (Sum)	Time (in Secs) (Sum)
5-5	40	0.38	40	0.38	40	0.22
5-10	15	0.63	15	0.76	17	0.37
5-20	11	0.94	11	1.31	11	0.30
20-5	77	27923.00	78	36000.00	90	0.32
20-10	34	36000.00	34	36000.00	50	0.28
20-20	20	36000.00	16	27881.00	28	0.24
50-5	340	36000.00	361	36000.00	352	1.06
50-10	323	36000.00	327	36000.00	315	0.90
50-20	305	36000.00	282	36000.00	261	0.83
100-5	868	36000.00	867	36000.00	832	3.62
100-10	891	36000.00	831	36000.00	786	3.44
100-20	927	36000.00	811	36000.00	717	3.12
200-10	1971	36000.00	1889	36000.00	1767	12.56
200-20	1964	36000.00	1871	36000.00	1699	12.67
Overall Sum	7786	387924.94	7433	387883.45	6965	39.93

Table 3.54. Comparison of Sacrifice & Rearrange Heuristic with the Mathematical Models on VRF Benchmark Set

Instance Set	Model 1 - MILP		Model 2 – CP		SRH	
	Objective Value (Sum)	Time (in Secs) (Sum)	Objective Value (Sum)	Time (in Secs) (Sum)	Objective Value (Sum)	Time (in Secs) (Sum)
10-5	58	6.12	58	99.08	59	0.61
10-10	64	10.47	64	450.72	66	0.57
10-15	72	13.20	72	405.80	72	0.55
10-20	82	15.86	82	272.57	82	0.61
20-5	87	23689.00	87	36000.00	100	0.75
20-10	104	34680.00	101	36000.00	112	0.68
20-15	108	18491.00	108	36000.00	123	0.79
20-20	110	27859.00	111	36000.00	123	0.76
30-5	126	36000.00	136	36000.00	142	0.89
30-10	140	36000.00	145	36000.00	155	0.58
30-15	142	36000.00	149	36000.00	160	0.63
30-20	152	36000.00	155	36000.00	161	0.59
40-5	169	36000.00	187	36000.00	184	0.71
40-10	180	36000.00	195	36000.00	191	0.77
40-15	187	36000.00	203	36000.00	198	0.70
40-20	190	36000.00	208	36000.00	202	0.74
50-5	212	36000.00	237	36000.00	223	0.86
50-10	226	36000.00	247	36000.00	231	0.95
50-15	241	36000.00	263	36000.00	241	0.94
50-20	264	36000.00	263	36000.00	240	0.99
60-5	252	36000.00	284	36000.00	262	1.14
60-10	283	36000.00	303	36000.00	266	1.66
60-15	322	36000.00	307	36000.00	275	1.14
60-20	332	36000.00	323	36000.00	276	1.37
Overall Sum	4103	648364.66	4288	624028.17	4144	19.98

Table 3.53 illustrates that the total run time needed to reach feasible solutions for MILP, CP, and SRH are 387924.94, 387883.45, and 39.93 seconds in the Taillard set. Similarly, as seen in Table 3.54, the total run time needed to reach feasible solutions for MILP, CP, and SRH is 648364.66, 624028.17, and 19.98 seconds in the VRF set. Lastly, the SRH outperformed in 101 out of 140 instances, achieving a success rate of 72.1% on the Taillard instances. It also excelled in 133 out of 240 instances, achieving a 55.4% success rate on the VRF instances with new and strict due dates. In summary, the SRH surpasses the mathematical models by quickly reaching good feasible solutions and consistently achieving superior objective function values within a few seconds on both sets. The objective values of Tables 3.53-3.54 provided in the literature are the best-known solutions of the VRF benchmark set with new due dates for the number of tardy jobs minimization in NWPFSs. The best-known solutions for each instance are presented in the following link: <https://data.mendeley.com/datasets/nfjs94x9wj/1>.

Section 3.8 addresses Research Questions 1, 4 and 5 of this thesis.

Research Question 1: Can we generate more effective *mathematical models* (MILP and CP model formulations) for the $(F_m|nwt|ServiceLevel)$ problems?

Research Question 4: Can we develop effective heuristics as quick *upper bounds* for the $(F_m|nwt|ServiceLevel)$ problems?

Research Question 5: Can we create a *due date generation mechanism* and establish *strong due dates* to address lateness-related performance measures in the $(F_m|nwt|ServiceLevel)$ problems?

This section discusses the $(F_m|nwt|\sum U_i)$ problem, addressing Research Questions 1, 4, and 5 as follows. An *MILP and a CP model* are studied for the problem. Subsequently, a *novel quick upper bound*, the Sacrifice and Rearrange Heuristic, yields good-quality feasible solutions quickly. Moreover, a *new due date generation mechanism*, based on the Sacrifice and Rearrange Heuristic, is proposed for this problem set. The computational results for both *Taillard* and *VRF* datasets are reported.

4. CHAPTER: BI-CRITERIA NO-WAIT PERMUTATION FLOWSHOP SCHEDULING PROBLEM

This chapter explores the NWPFS in the context of bi-criteria minimization. Q-learning guided algorithms are developed to address the Bi-Criteria No-Wait Flowshop Scheduling Problem (BI-CRI NWPFS). Combining Deep Reinforcement Learning and meta-heuristic techniques represents a new research direction for enhancing the search capabilities of meta-heuristic methods in the context of production scheduling. Q-learning is a prominent reinforcement learning method whose utilization aims to direct the selection of actions, thus preventing the necessity for a random exploration in the iterative process of metaheuristics. The NWPFS is treated as a bi-criteria combinatorial optimization problem where total flow time and makespan are optimized simultaneously, as explained in Section 4.1. Section 4.2 reviews the literature on BI-CRI NWPFS and Q-learning-based algorithms for scheduling problems. Firstly, a deterministic MILP model is presented and detailed in Section 4.3. Then, Q-learning guided algorithms are developed: Bi-Criteria Iterated Greedy Algorithm with Q-learning (BC-IG_{QL}) and Bi-criteria Block Insertion Heuristic Algorithm with Q-learning (BC-BIH_{QL}). Moreover, the performance of the proposed Q-learning guided algorithms is compared over a collection of Bi-Criteria Genetic Local Search Algorithms (BC-GLS), Bi-Criteria Iterated Greedy Algorithm (BC-IG), Bi-Criteria Iterated Greedy Algorithm with a Local Search (BC-IG_{ALL}) and Bi-Criteria Variable Block Insertion Heuristic Algorithm (BC-VBIH). All metaheuristics are explained in Section 4.4. A comprehensive computational analysis is conducted in Section 4.5. The complete computational experiment performed on 480 problem instances of Vallada et al. (2015), which is known as the VRF benchmark set, indicates that the BC-BIH_{QL} and the BC-IG_{QL} algorithms outperform the BC-GLS, BC-IG, BC-IG_{ALL}, and BC-VBIH algorithms in comparative performance metrics. More specifically, the proposed BC-BIH_{QL} and BC-IG_{QL} algorithms can yield more non-dominated bi-criteria solutions with the most substantial competitiveness than the remaining algorithms.

4.1. Bi-Criteria Problem Setting

This section contributes to the $(F_m|nwt|C_{max}, \sum C_i)$ problem, which minimizes makespan and total flow time simultaneously in NWPFPs. Two objectives conflict with each other. The trade-off is be illustrated with an example. Concerning the definitions provided in Section 2, Figure 4.1 displays a Gantt Chart illustrating an example of the NWPFP with five jobs and five machines. The processing times for job $j \in N$ on machine $r \in M$ (P_{jr}) in this example are as follows.

$$P_{jr} = \begin{bmatrix} 77 & 39 & 14 & 11 & 83 \\ 94 & 31 & 21 & 2 & 13 \\ 9 & 46 & 15 & 36 & 84 \\ 57 & 18 & 10 & 30 & 46 \\ 29 & 93 & 85 & 89 & 20 \end{bmatrix}$$

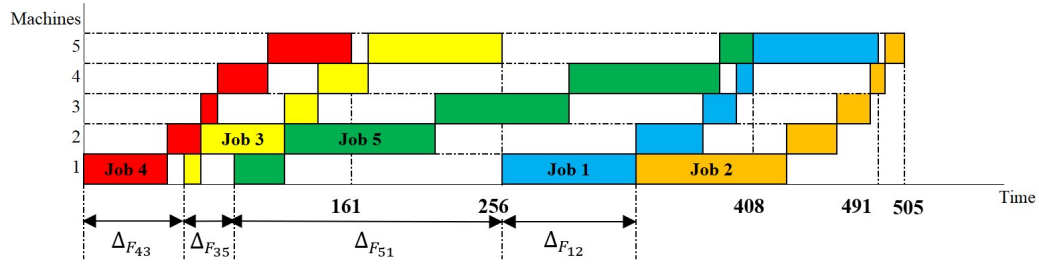


Figure 4.1. The Gantt Chart represents the example with 5 jobs on 5 machines in an NWPFP

Source: Yüksel et al. (2024)

Figure 4.1 depicts that the sequence $\pi = \{4,3,5,1,2\}$. The makespan and total flow time values are 505 and 1821 ($161+256+408+491+505$), respectively. Analyzing the trade-off between total flow time and makespan involves considering all permutations for this small example. Given five jobs, there are $5!$ different sequences which yield a unique makespan and a total flow time value representing a point solution in 2D. Among these, six sequences emerge as non-dominated solutions, forming the Pareto set for this instance. The dominated solution set and the Pareto frontier of the example are depicted in Figure 4.2.

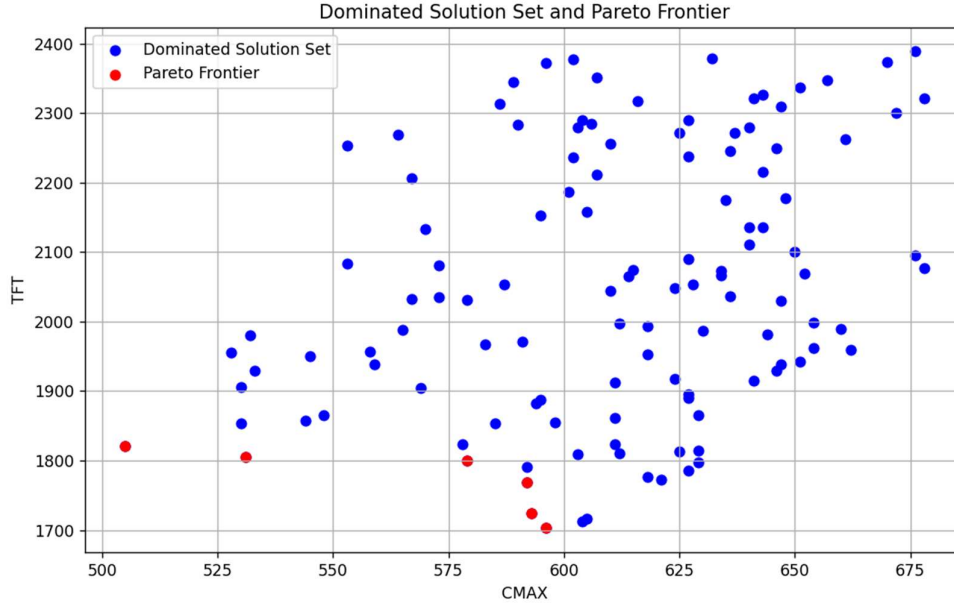


Figure 4.2. The dominated solution set and the Pareto frontier of the example with jobs five and five machines in an NWPFS

Source: Yüksel et al. (2024)

As depicted in Figure 4.2, there are $5!$ different sequences, resulting in 120 points in 2D, whereas 6 of them are non-dominated solutions consisting of the Pareto front for this example. The extreme points (1821,505) and (1703,596) represent the minimum values achievable for makespan or the total flow time, respectively. For a more comprehensive understanding of the trade-off between the objective functions, Figures 4.5 and 4.6 can be referred to, offering a more explicit presentation in more prominent instances.

4.2. Literature Review of Bi-Criteria NWPFSs and Q-learning-based Algorithms for Scheduling Problems

4.2.1. Bi-Criteria No-Wait Permutation Flowshop Scheduling Problems

A very early multi-objective study on NWPFSs was presented by Allahverdi and Aldowaisan (2004). Two algorithms, simulated annealing & genetic heuristics, are uncovered for the maximum lateness and the makespan minimization in this study, and the effectiveness of the proposed hybrid algorithms over literature is illustrated. Javadi

et al. (2008) originated a fuzzy multi-criteria LP model for optimizing the weighted mean earliness and the weighted mean completion time on problems with fuzzy objectives. Liu et al. (2008) studied an objective increment-based genetic algorithm for optimizing the total flow time and the makespan. Then, the simultaneous minimization of the weighted mean tardiness and the weighted mean completion time was studied by an immune algorithm in NWPFSs in Tavakkoli-Moghaddam et al. (2008). A particle swarm optimization algorithm was carried out in the study of Pan & Wang (2008b) for the bi-criteria NWPFSs, where the maximum tardiness and the makespan were minimized. However, the same objectives are optimized with a Differential Evolution (DE) algorithm in the study of Pan et al. (2009) while illustrating the superiority of DE over multi-criteria genetic algorithm in performance metrics. Next, the mean tardiness and the weighted sum of mean completion times were minimized in NWPFSs (Rahimi-Vahed et al. (2008)). A multi-criteria scatter search algorithm obtains the Pareto-optimal frontiers of this problem. Qian et al. (2009) created a DE-based memetic algorithm favoring several objective functions. Jenabi et al. (2010) suggested two MILP models and an iterated local search algorithm to optimize the total tardiness and the makespan. Khalili and Tavakkoli-Moghaddam (2012) developed algorithms for the variant in which the transportation times among machines were considered. Later, Naderi et al. (2012) minimized the total tardiness and the makespan criteria with the help of fuzzy programming and iterated local search approaches. The supremacy of the proposed algorithm was explicitly depicted over other algorithms. Keskin and Engin (2021) optimized the total flow time and the makespan by hybrid algorithms in NWPFSs with setup times.

Regarding hierarchical minimization, the minimization of makespan contingent on an upper limit for the mean flow time has been explored in Aydilek and Allahverdi (2012). In contrast, the other way around hierarchy (that is, optimization of total completion time contingent on the makespan) was studied by Allahverdi and Aydilek (2013). In this type of hierarchical minimization, the upper bound on the second objective is precalculated and fed to the algorithms to minimize the main objective. Later, Allahverdi et al. (2018) studied the total tardiness contingent on an upper bound on the makespan in NWPFSs. Next, Nagano et al. (2020) revealed an Iterated Greedy (IG) algorithm for the hierarchical minimization of makespan contingent on an upper limit for the total flow time. In a recent study, the total tardiness minimization, subject to an

upper bound on the total completion time, was carried out by Allahverdi et al. (2022). Similarly, Almeida and Nagano (2023) illustrated the effectiveness of the IG algorithms in optimizing the total flow time and makespan hierarchically in NWPFSPs with setup times.

Yet another bi-criteria point of view on the NWPFSPs is the consideration of energy efficiency. Recently, Yüksel et al. (2020b) studied total tardiness minimization and total energy consumption minimization in the NWPFSP setting. They revealed that their proposed discrete artificial bee colony algorithm (ABC) outperforms all others. Similar energy efficiency minimization ideas lie in Taşgetiren et al. (2019) and Yüksel et al. (2020a) for total flow time – total energy consumption and makespan – total energy consumption minimization problems, respectively. Moreover, Wu and Che (2020) developed an adaptive multiobjective Variable Neighborhood Search (VNS) algorithm to minimize the makespan and total energy consumption simultaneously.

As another industrial application of NWPFSPs, the no-wait hybrid/flexible flowshop environment is also widespread. The no-wait hybrid/flexible flowshop problem is identified as n jobs must be treated by k -stages where the jobs' stage visiting order is identical; namely, each job continues its processing from stage 1 until stage k (Pinedo, M. L. (2012)). The multi-criteria k -stage no-wait flexible flowshop scheduling problem was studied by Khalili, M. (2014) with the electromagnetism algorithm for optimizing the total tardiness and the makespan. Later, Jolai et al. (2013) developed three simulated annealing algorithms to optimize the maximum tardiness and the makespan in a two-stage case. Asefi et al. (2014) introduced a hybrid Non-dominated Sorting Genetic Algorithm (NSGA-II) and a VNS algorithm to optimize mean tardiness and makespan for a variant in which the setup times are sequence-dependent. Also, transportation time between machines can be encountered in many production systems. Thus, Khalili, M. (2012) and Naderi and Sadeghi (2012) studied the same problem with transportation times to optimize the total tardiness and the makespan. Finally, Tao and Liu (2019) developed genetic algorithms for the dual-resource constrained no-wait hybrid/flexible flowshop scheduling problems with unrelated parallel machines to optimize the makespan, the total expense, and the mean flow time.

All studies performed on BI-CRI NWPFPs are summed up in terms of the objective functions, the solution methodologies, and the configuration in chronological order in Tables 4.1-4.5. In the last row of Table 4.5, the specific features of this chapter are stated in the same setting.

Table 4.1. Bi-Criteria NWPFPs

Authors (Year)	Objective Criteria	Objective Criteria Analysis Type	Solution Procedure	Configuration
Allahverdi & Aldowaisan (2004)	Weighted Sum of Maximum Lateness and Makespan	Simultaneous	Hybrid SA and Hybrid Genetics	No-wait Permutation Flowshop
Javadi et al. (2008)	Weighted Mean Completion Time and Weighted Mean Earliness	Simultaneous	Fuzzy LP	No-wait Permutation Flowshop
Liu et al. (2008)	Total Flow Time and Makespan	Simultaneous	Genetic Algorithm	No-wait Permutation Flowshop
Tavakkoli-Moghaddam et al. (2008)	Weighted Mean Time and Weighted Mean Tardiness	Simultaneous	Immune Algorithm	No-wait Permutation Flowshop
Pan & Wang (2008b)	Maximum Tardiness and Makespan	Simultaneous	PSO Algorithm	No-wait Permutation Flowshop
Rahimi-Vahed et al. (2008)	Weighted Sum of Mean Completion Times and Mean Tardiness	Simultaneous	Multi-Criteria Scatter Search	No-wait Permutation Flowshop
Pan et al. (2009)	Maximum Tardiness and Makespan	Simultaneous	Novel DE Algorithm	No-wait Permutation Flowshop

Table 4.2. Bi-Criteria NWPFPs

Authors (Year)	Objective Criteria	Objective Criteria Analysis Type	Solution Procedure	Configuration
Qian et al. (2009)	Several Objectives	Simultaneous	Memetic Algorithm-Based DE	No-wait Permutation Flowshop
Jenabi et al. (2010)	Total Tardiness and Makespan	Simultaneous	Two MILP Models and Iterated Local Search	No-wait Permutation Flowshop
Aydilek and Allahverdi (2012)	Makespan contingent on an upper bound on Mean Flow Time	Hierarchical	Different Heuristics	No-wait Permutation Flowshop
Khalili and Tavakkoli- Moghaddam, (2012)	Total Weighted Tardiness and Makespan	Simultaneous	Electromagnetism Algorithm and Simulated Annealing algorithms	No-wait Permutation Flowshop
Khalili, M. (2012)	Total Tardiness and Makespan	Simultaneous	Electromagnetism Algorithm	k -Stage No-wait Hybrid Flowshop
Naderi and Sadeghi (2012)	Total Tardiness and Makespan	Simultaneous	Simulated Annealing	k -Stage No-wait Hybrid Flowshop
Naderi et al. (2012)	Total Tardiness and Makespan	Simultaneous	Fuzzy Programming and An Iterated Local Search Algorithm	No-wait Permutation Flowshop
Allahverdi and Aydilek (2013)	Total Completion Time contingent on an upper bound on makespan	Hierarchical	A Proposed Algorithms and Genetic Algorithm	No-wait Permutation Flowshop

Table 4.3. Bi-Criteria NWPFSFs

Authors (Year)	Objective Criteria	Objective Criteria Analysis Type	Solution Procedure	Configuration
Jolai et al. (2013)	Maximum Tardiness and Makespan	Simultaneous	Three Simulated Annealing Algorithms	2-Stage No-wait Hybrid Flowshop
Asefi et al. (2014)	Makespan and Mean Tardiness	Simultaneous	Hybrid NSGA-II and VNS Algorithm	k -Stage No-wait Hybrid Flowshop
Khalili, M. (2014)	Makespan and Total Tardiness	Simultaneous	Electromagnetism Algorithm	k -Stage No-wait Hybrid Flowshop
Allahverdi et al. (2018)	Total Tardiness contingent on an upper bound on Makespan	Hierarchical	A variant of the Simulated Annealing Algorithm	No-wait Permutation Flowshop
Tao and Liu (2019)	Makespan, Total Expense, Mean Flow Time	Simultaneous	Genetic Algorithm	k -Stage No-wait Hybrid
Taşgetiren et al. (2019)	Total Flow Time and Total Energy Consumption	Simultaneous	MILP model, Discrete ABC, Genetic Algorithm	No-wait Permutation Flowshop
Nagano et al. (2020)	Makespan contingent on an upper bound on Total Flow Time	Hierarchical	IG Algorithm	No-wait Permutation Flowshop
Wu and Che (2020)	Makespan and Total Energy Consumption	Simultaneous	MILP model, Adaptive Multiobjective VNS	No-wait Permutation Flowshop

Table 4.4. Bi-Criteria NWPFSs

Authors (Year)	Objective Criteria	Objective Criteria Analysis Type	Solution Procedure	Configuration
Yüksel et al. (2020a)	Makespan and Total Energy Consumption	Simultaneous	MILP model, Variable Block Insertion Heuristic, IG and IG _{ALL} algorithms	No-wait Permutation Flowshop
Yüksel et al. (2020b)	Total Tardiness and Total Energy Consumption	Simultaneous	MILP and CP models, Discrete ABC, Genetic Algorithm	No-wait Permutation Flowshop
Keskin and Engin (2021)	Makespan and Total Flow Time	Simultaneous	Hybrid Genetic Local and Global Search Algorithm, IG Algorithm	No-wait Permutation Flowshop with Setup Times
Allahverdi et al. (2022)	Total Tardiness contingent on an upper bound on Total Completion Time	Hierarchical	Constructive Algorithm	No-wait Permutation Flowshop
Almeida and Nagano (2023)	Total Flow Time contingent on an upper bound on Makespan	Hierarchical	Iterated Greedy Algorithms	No-wait Permutation Flowshop with Setup Times

Table 4.5. Bi-Criteria NWPFPSPs

Authors (Year)	Objective Criteria	Objective Criteria Analysis Type	Solution Procedure	Configuration
<i>This Thesis</i>	<i>Total Flow Time and Makespan</i>	<i>Simultaneous</i>	<i>MILP model, Augmented- Epsilon Constraint Algorithm, Iterated Greedy (IG) Algorithm with Q- learning, Block Insertion Heuristic Algorithm with Q- learning, Genetic Local Search Algorithms, Iterated Greedy Algorithm, Block Insertion Heuristic</i>	<i>No-wait Permutation Flowshop</i>

Regarding the literature review on the BI-CRI NWPFPSPs, simultaneous optimization of the total flow time and the makespan criterion resulted in the main focus of this thesis. The makespan describes the maximum completion time of the jobs, whereas the total flow time represents the sum of completion times for all jobs. Both are critical criteria in the industry. The makespan considers the efficient use of resources, whereas the total flow time is considered while rapidly meeting the demands (Aydilek and Allahverdi, 2012). However, the two are conflicting. Li et al. (2014) demonstrated the trade-off challenge between minimizing makespan and total completion time, resulting in minimizing makespan does not necessarily lead to minimizing total flow time, and conversely. Moreover, Li et al. (2019) and Ye et al. (2020) illustrated the minimization of each objective, makespan, and total flow time results in conflicting scheduling sequences, which correlate relatively little in PFSPs and NWPFPSPs, respectively. To deal with this trade-off, they proposed trade-off balancing heuristics regarding the

dominance evaluation, as in Pareto efficiency, for the corresponding scheduling problems.

4.2.1. Q-Learning-Based Algorithms for Scheduling Problems

Reinforcement Learning (RL) is a subfield of machine learning that focuses on sequential decision-making (Kaelbling et al. (1996)). Combining Deep Reinforcement Learning (DRL) and meta-heuristic techniques represents a new research direction for enhancing the search capabilities of meta-heuristic methods in the context of production scheduling (Talbi, E. G. (2016); Chen et al. (2020); Cai et al. (2023)). So, it is a growing field of research, with ongoing efforts to explore different ways to leverage the power of machine learning for solving complex scheduling problems. This integration offers various possibilities, such as utilizing machine learning for tasks like parameter configuration, fitness evaluation, solution initialization, and particularly, in the context of Adaptive Operator Selection, for operator selection (Karimi-Mamaghan et al. (2023)). The Q-learning algorithm is a notable method in RL. It is categorized as a model-free RL technique that relies on temporal differences (Watkins, C. J. C. H. (1989)). The Q-learning mechanism aims to direct the selection of actions, thus preventing the necessity for a random exploration in the iterative process.

There has been a notable expansion in applying reinforcement learning techniques within combinatorial optimization problems in recent years. Kizilay et al. (2020) embedded the Q-learning into the evolutionary algorithm to decide some parameters of mutation strategy, such as crossover and mutation rates, for various engineering design problems. An application of DRL-guided DE over constraint optimization is performed by Hu et al. (2023). In contrast, it is expanded to multi-objective combinatorial optimization problems as an application on the traveling salesperson problem by Li et al. (2023). DRL has also recently been carried out for satellite range scheduling problems by Ou et al. (2023) and for uncrewed surface vessel scheduling problems by Gao et al. (2023). Another fascinating field of application of Q-learning integration with metaheuristics is urban traffic light scheduling problems. Lin et al. (2023) and Wang et al. (2023) studied several metaheuristics with Q-learning. They

illustrated the competitiveness of the Q-learning-guided metaheuristic algorithms, among others, for urban traffic light scheduling problems.

Significantly, this trend is undergoing swift growth in manufacturing scheduling. The Q-learning is incorporated into the IG algorithms to minimize the makespan in PFSPs (Karimi-Mamaghan et al. (2023)). The perturbation mechanism uses the Q-learning framework to select the perturbation operator dynamically in IG algorithms. Li et al. (2022a) studied the Q-learning-guided ABC algorithm for the PFSPs and showed the competitiveness of the Q-learning-adapted ABC algorithm over other algorithms. Also, various metaheuristics are enhanced by Q-learning integration in distributed assembly PFSPs (Yu et al. (2023)). In the no-idle permutation flow shop scheduling problems (NIPFSP) field, Oztop et al. (2020) initially generated a VNS algorithm with Q-learning to optimize the makespan objective. After that, they enhanced the learning mechanism, integrated it into the IG algorithm, and iterated the local search mechanism (Öztop et al., 2022). Furthermore, Zhao et al. (2021) carried out an RL mechanism for the distributed assembly variant of NIPFSP to optimize the maximum assembly completion time. Within the domain of blocking permutation flowshop scheduling problems (BPFSP), Tasgetiren et al. (2024) suggest employing Q-learning-driven IG algorithms to minimize the makespan. While Ren et al. (2023) demonstrated the superior performance of the Q-learning-guided VNS algorithm on the disassembly line scheduling problems, Zhang et al. (2022) studied DRL on a proposed more complex multi-agent manufacturing system. Besides, the DRL framework is embedded in the robotic flow shop scheduling problem in consecutive studies by Lee and Kim (2022) and Kim and Lee (2023) for minimizing the makespan.

A DRL framework is employed to autonomously acquire a solution for the flexible job-shop scheduling problem (FJSSP) (Lei et al. (2022)). In this problem setting, the reinforcement agent must coordinate multiple actions concurrently, such as assigning operations to jobs and finding suitable machines among available machines. Hence, a multi-action DRL mechanism is carried out on the FJSSP setting to minimize the makespan. FJSSP is also considered an RL-based evolutionary algorithm by Li et al. (2022b). However, two objectives are optimized: makespan and total machine workload. Chang et al. (2022) also address the dynamic version of FJSSP by integrating a DRL framework. The proposed DRL framework minimized the penalties

for earliness and tardiness for the problem in which jobs arrive randomly. Next, Du et al. (2022) generated a knowledge-based RL for the FJSSP setting to minimize makespan and total electricity costs, whereas Chen et al. (2023) incorporated the Q-learning into an immune algorithm to be applied to fuzzy FJSSP. In the field of hybrid flow shop scheduling problems (HFSP), Cai et al. (2023) proposed integrating a novel shuffled frog-leaping algorithm with DRL to minimize makespan. The algorithm is implemented on a distributed assembly HFSP with fabrication, transportation, and assembly considerations.

In the NWPFSPs field, the studies integrating Q-learning are primarily implemented on the distributed version of the NWPFSPs (Zhao et al. (2022b)) and even on the distributed heterogeneous NWPFSPs (Zhao et al. (2022c)). For energy-efficiency considerations in PFSPs, Q-learning is embedded into metaheuristic algorithms for the multi-objective energy-efficient distributed NWPFSPs (Zhao et al. (2022b)), the multi-objective energy-efficient distributed blocking flow shop scheduling problems (BPFSP) (Zhao et al. (2022a); Bao et al. (2023)) and the multi-objective energy-efficient distributed assembly NWPFSPs (Zhao et al. (2023)).

RL methods have recently been extensively used in manufacturing scheduling applications, but their utilization in the NWPFSPs has been limited. Incorporating Q-learning into algorithms for addressing multi-objective optimization problems, particularly in production scheduling, represents a novel and promising research direction. This thesis is the first to integrate the Q-learning algorithm into metaheuristic approaches within a multi-objective/bi-objective NWPFSP setting where makespan and total flowtime are simultaneously minimized. We anticipate this thesis will serve as a valuable reference for implementing Q-learning in the context of multi-objective/bi-objective production scheduling problems in literature.

4.3. Mathematical Model

Model 2, as explained in Section 3.2, is examined for the $(F_m|nwt|C_{max}, \sum C_i)$ problem. Table 4.6 presents the notations used in the MILP model.

Table 4.6. Notations for MILP Model

Sets	
N	Job set
M	Machine set
Indices	
i, k	Job index ($1 \leq i, k \leq N$)
r	Machine index ($1 \leq r \leq M$)
p	Position index ($1 \leq p \leq N$)
Parameters	
P_{ir}	Processing time of job $i \in N$ on machine $r \in M$
Δ_{Fik}	The delay time between two successive jobs i and k ($i, k \in N$) on the first machine
Q	Big number
Decision Variables	
B_i	Beginning time of job i on the first machine
C_i	Completion time of job i on the last machine
δ_{ik}	{1 if job i is positioned immediately before job k , 0 otherwise}
C_{max}	Makespan
$\sum_{i \in N} C_i$	Total flow time

The MILP model originated for the NWPFSP to optimize the total flow time and the makespan simultaneously is derived from the Model 2 – MILP model from Section 3.2 and depicted below:

MILP Model.		
Minimize C_{max}		(4.1)
Minimize $\sum_{i \in N} C_i$		(4.2)
Subject to;		
$\sum_{i=1: i \neq k}^N \delta_{ik} \leq 1$	$\forall k \in N$	(4.3)
$\sum_{i=1: i \neq k}^N \delta_{ki} \leq 1$	$\forall k \in N$	(4.4)
$Q * (1 - \delta_{ik}) + B_k - B_i \geq \Delta_{F_{ik}}$	$\forall i, k \in N: i \neq k$	(4.5)
$\sum_{i=1: i \neq k}^N \sum_{k=1}^N \delta_{ik} = N - 1$		(4.6)
$C_{max} \geq C_i$	$\forall i \in N$	(4.7)
$C_i \geq B_i + \sum_{r \in M} P_{ir}$	$\forall i \in N$	(4.8)
$B_k \geq 0$	$\forall k \in N$	(4.9)
$\delta_{ik} \in \{0,1\}$	$\forall i, k \in N$	(4.10)

The objective functions (4.1)-(4.2) calculate the makespan and the total flow time to minimize them. Constraints (4.3)-(4.4) indicate the sequence relation of two consecutive jobs. Constraint (4.5) indicates that if two jobs are consecutive, then the difference between their starting times on the first machine will be at least the time delay of those jobs on the first machine ($\Delta_{F_{ik}}$). $\Delta_{F_{ik}}$ values for two consecutive jobs i and k are depicted in Figure 4.1. These values are predetermined values to the problem since they only rely on two successive jobs' processing times values (Wismer, D. A. (1972); Pan et al. (2008)). Constraint (4.6) restricts the binary variables to the number of jobs minus 1. Then, the makespan and each job's completion time values are calculated by constraints (4.7)-(4.8). Constraints (4.9)-(4.10) are the sign and variables indicating constraints. This model is different than the classical approach of Manne (1960) based on the point of view of the problem in such a way that this model considers sequencing jobs on the first machine regarding the $\Delta_{F_{ik}}$ values.

4.4. Metaheuristic Algorithms

Because of the no-wait limitation, two successive jobs in the sequence only hold a specific feature incident to themselves. The completion time of two successive jobs can be computed in advance. Hence, the difference between two successive jobs'

completion times relies entirely upon these two successive jobs (Wisner, D. A. (1972); Pan et al. (2008)), irrespective of the alignment of other jobs and their positions in the sequence. Thanks to this feature, a delay matrix where all $\Delta_{F_{\pi_{p-1}, \pi_p}}$ (see Figure 3.1 and Figure 4.1) values can be computed previously. Each of this matrix, namely the delay values for any schedule π (denotes the sequence of jobs) is calculated as given below:

$$\Delta_{F_{\pi_{p-1}, \pi_p}} = P_{\pi_{p-1}, 1} + \max \left\{ 0, \max_{2 \leq p \leq m} \left\{ \sum_{r=2}^p P_{\pi_{p-1}, r} - \sum_{r=1}^{p-1} P_{\pi_p, r} \right\} \right\} \quad (4.11)$$

$$\forall p = 2, \dots, N$$

Therefore, as indicated below, the makespan can be quickly calculated.

$$C_{max}(\pi) = \sum_{p=2}^N \Delta_{F_{\pi_{p-1}, \pi_p}} + \sum_{r=1}^M P_{\pi_N, r} \quad (4.12)$$

Then, the first job's completion time on the tailender machine is explicitly computed as given below:

$$C_{\pi_1, M}(\pi_1) = \sum_{r=1}^M P_{\pi_1, r} \quad (4.13)$$

Eventually, any job's completion time on the tailender machine can be assessed as provided below.

$$C_{\pi_p, M}(\pi_p) = \sum_{p=2}^N \Delta_{F_{\pi_{p-1}, \pi_p}} + \sum_{r=1}^M P_{\pi_1, r} \quad \forall p = 2, \dots, N \quad (4.14)$$

In conclusion, our second objective -total flow time- is computed as follows.

$$\sum_{i \in N} C_i(\pi) = \sum_{p=1}^N C_{\pi_p, M}(\pi_p) \quad (4.15)$$

For example, for the given sequence of $\pi = \{4, 3, 5, 1, 2\}$, that is illustrated in Figure 4.1;

$$C_{max}(\pi) = 505, \quad C_{\pi_1, M}(\pi_1) = 161, \quad C_{\pi_2, M}(\pi_2) = 256, \quad C_{\pi_3, M}(\pi_3) = 408, \\ C_{\pi_4, M}(\pi_4) = 491, C_{\pi_5, M}(\pi_5) = 505, \text{ and } \sum_{i \in N} C_i(\pi) = 1821.$$

Considering the second objective, the problem turned out to be a bi-criteria NWPFSFSP with simultaneous optimization of the makespan and the total flow time to approach a set of non-dominant solutions. Unlike single-criteria optimization problems, multi-criteria optimization problems are based on three primary notions while establishing the relations among the solutions found at various points in the output. These are *dominance relation*, a *set of Pareto-optimal solutions*, and a *set of non-dominant*

solutions (Deb, K. (2011)). These concepts are handled during solution approach implementation, e.g., running the MILP model and implementing the metaheuristics.

Two Q-learning guided algorithms are designed and proposed for the BI-CRI NWPFSFSP.

- Bi-Criteria Iterated Greedy Algorithm with Q-learning (BC-IG_{QL}),
- Bi-Criteria Block Insertion Heuristic with Q-learning (BC-BIH_{QL}),

The performance of the proposed Q-learning-guided algorithms (BC-IG_{QL} and BC-BIH_{QL}) are compared. Thus, *the competitiveness of two Q-learning-guided algorithms with each other* has been illustrated.

Then, the performance of the BC-IG_{QL} and BC-BIH_{QL} algorithms are compared with two traditional IG algorithms and a variant of the BIH algorithm to illustrate the Q-learning effect. These are annotated as follows.

- Bi-Criteria Iterated Greedy Algorithm (BC-IG)
- Bi-Criteria Iterated Greedy Algorithm with a Local Search (BC-IG_{ALL})
- Bi-Criteria Variable Block Insertion Heuristic (BC-VBIH)

By doing so, *the contribution of Q-learning over IG and BIH algorithms, with and without Q-learning*, has been illustrated.

Then, BC-IG_{QL} and BC-BIH_{QL} are compared with the five variants of Genetic Local Search algorithms, which are effective algorithms for scheduling problems in the literature. The effectiveness of Genetic Algorithms in this problem set has already been proven by Liu et al. (2008). Hence, Genetic Local Search Algorithms have been selected as the group of algorithms for comparison in this research. These are annotated as follows.

- Bi-Criteria Genetic Local Search Algorithm (BC-GLS) variants:
 - Bi-Criteria Genetic Local Search Algorithm with One-cut Crossover (BC-GLS_{ONECUT}),
 - Bi-Criteria Genetic Local Search Algorithm with Two-cut Crossover (BC-GLS_{TWOCUT}),

- Bi-Criteria Genetic Local Search Algorithm with Order Crossover (BC-GLS_{OX}),
- Bi-Criteria Genetic Local Search Algorithm with Partially Mapped Crossover (BC-GLS_{PMX}),
- Bi-Criteria Genetic Local Search Algorithm with Particle Crossover (BC-GLS_{PTL}),

By doing so, *the competitiveness of Q-learning-guided algorithms over existing scheduling algorithms* in the literature has been illustrated. To sum up, the effectiveness of the designed Q-learning guided algorithms in terms of various perspectives has been proved and demonstrated to the readers.

4.4.1. Iterated Greedy Algorithm

The conventional Iterated Greedy (IG) algorithm, as outlined by Ruiz and Stützle (2007), can be briefly described as follows. The typical IG comprises four primary components: initial solution generation, the destruction-construction phase, local search, and the acceptance criteria. IG_{ALL} algorithm is presented in the literature in the study of Dubois-Lacoste et al. (2017) for a single-objective PFSP in minimizing the makespan objective. The IG_{ALL} algorithm is an extended version of the IG algorithm. The distinction between IG_{ALL} and IG algorithms is that after the destruction of the sequence, an extra insertion local search is employed for the partial solutions in IG_{ALL} since it aims to optimize the destructed partial solutions. The acceptance criteria of a solution is determined by the dominance relationship in comparing two solutions. A Bi-Criteria Iterated Greedy Algorithm (BC-IG) and a Bi-Criteria Iterated Greedy Algorithm with a Local Search (BC-IG_{ALL}) are carried out for the NWPFSF.

4.4.2. Block Insertion Heuristic Algorithm

The foundational principles of the Block Insertion Heuristic (BIH) algorithm were created by Kirlik and Oguz (2012); Subramanian et al. (2014); Xu et al. (2014) and González and Vela (2015). The basic concept involves employing more extensive neighborhood move operators by *Block Move* instead of the swap and insertion move operators. Blocks are groups of interconnected consecutive jobs. The BIH comprises

four primary components: initial solution generation, the block move & block insertion phase, local search, and the acceptance criteria. Inspired by the aforementioned studies, a variable BIH algorithm (VBIH) is studied for PFSPs, single machine scheduling problems, and blocking PFSPs by the studies of Tasgetiren et al. (2016a), Tasgetiren et al. (2016b), and Kizilay et al. (2019), respectively. VBIH is a BIH variant that employs block sizes varying between minimum and maximum block size limits. The VBIH algorithm functions by extracting a block of jobs from the existing solution, Block Move, followed by conducting an insertion local search on this partial solution. Subsequently, it sequentially reintroduces the block into every potential position within the partial solution, that is, Block Insertion, ultimately selecting the most optimal one from these. The acceptance criteria of a solution is determined by the dominance relationship in comparing two solutions. A Bi-Criteria Variable Block Insertion Heuristic (BC-VBIH) algorithm is employed for the NWPFSP.

4.4.3. Q-Learning

Reinforcement Learning (RL), a domain within machine learning, employs a sequential strategy where an agent is trained to determine the most appropriate action to take in a specific state to reach an objective (Kaelbling et al. (1996)). The Q-learning technique, developed by Watkins, C. J. C. H. (1989), is a model-independent RL algorithm that utilizes temporal differences. After the agent executes a selected action, it receives either a reward or a penalty. Subsequently, it learns to select the optimal action by evaluating the available action choices based on the cumulative rewards they yield.

In Q-learning, a score $Q(s, a)$ is assigned to each state-action pair (s, a) within the state space (S) and action space (A) . This score represents the anticipated benefit of opting for a particular action in the state s , and is calculated as follows.

$$Q(s, a) := Q(s, a) + \theta [r + \gamma \max_{a'} Q(s', a') - Q(s, a)], \quad (4.16)$$

where

s : current state

a : action that is taken in the state s

s' : next state after acting a

a' : possible action in the state s'

θ : the learning rate of new information ($0 \leq \beta \leq 1$)

r : reward obtained after acting a

γ : discount factor of influence on the future reward $\max_{a'} Q(s', a')$

The Q-values of the selected actions for the parameters are modified using the Q-learning function during each iteration. In the subsequent iteration, the algorithm chooses the optimal action for each parameter based on the highest Q-value. Q-learning converges to the optimal $Q(s, a)$ values when each state-action pair is visited sufficiently (Watkins, C. J. C. H. (1989)).

Then, the ϵ -decay strategy is applied to move from exploration to exploitation gradually. The ϵ -greedy strategy is a widely used approach that balances exploration and exploitation. The ϵ -greedy approach, as outlined by Sutton & Barto (2018), effectively balances exploration and exploitation by assigning an ϵ probability to alternative actions, thus creating a harmonious equilibrium between the two. It assigns an exploration probability ϵ , which allows for selecting actions other than the one with the maximum Q-value. This approach helps to ensure a more comprehensive exploration of the solution space, increasing the chances of discovering optimal or near-optimal solutions.

$$a = \begin{cases} \max_a Q(s, a) & \text{with } 1 - \epsilon \text{ prob.} \\ \text{randomly or uniformly selected action from } A & \text{with } \epsilon \text{ prob.} \end{cases} \quad (4.17)$$

Typically, the value of ϵ is gradually decreased during the search process, employing a parameter known as β , which is referred to as ϵ -decay. Thus, the Q-learning process not only achieves a balance between exploration and exploitation but also enables the utilization of state-action pairs that have not been previously encountered.

4.4.4. Q-Learning Guided Metaheuristic Algorithms

This thesis uses RL, specifically the Q-learning algorithm, as a machine learning technique to implement Adaptive Operator Selection within the IG and BIH algorithms. The objective is to dynamically select suitable perturbation operators, specifically the *Destruction Size* and *Block Size* operators, at each search step by

considering the search's current state and the operators' historical performance. By employing the Q-learning algorithm, we aim to enhance the effectiveness of the Adaptive Operator Selection mechanism and improve the overall performance of the IG and BIH algorithms in finding optimal solutions. This integration is inspired by the study of Karimi-Mamaghan et al. (2023). They explicitly illustrated the efficiency of integrating the Q-learning algorithm into the IG algorithm in minimizing the single objective, that is, makespan, in PFSPs, in which Q-learning is performed on perturbation operators of the IG algorithm. Therefore, the details of the Q-learning integration can be found in Karimi-Mamaghan et al. (2023). Two Q-learning guided bi-criteria algorithms are proposed for the BI-CRI NWPFSs and annotated as follows:

- Bi-Criteria Iterated Greedy Algorithm with Q-learning (BC-IG_{QL})
- Bi-Criteria Block Insertion Heuristic with Q-learning (BC-BIH_{QL})

In the proposed Q-learning guided algorithms, we utilize the Q-learning algorithm to implement a learning mechanism within the IG and BIH algorithms and autonomously learn the parameter values of the algorithms. During each search phase, incorporating Q-learning into the IG and BIH algorithms revolves around choosing the most suitable perturbation operators, Destruction Size (dS), and Block Size (bS). This is done to enhance the algorithm's capability to explore more effectively. Consequently, the perturbation operators are defined as the set of possible actions. In essence, choosing the right perturbation operator is crucial for moving out of local optima and accelerating the progress towards reaching near-optimal solutions.

Both algorithms employ the following solution representation π^i , where i denotes the i^{th} solution and π denotes the sequence of jobs. Then, the single initial solution (the initial population of size $NP = 1$) is generated with an initial solution π^0 that is built according to the NEH heuristic (Nawaz et al. (1983)) in the BC-IG_{QL} and BC-BIH_{QL} algorithms. The archive set ϕ is vacant at initial construction; however, the non-dominant solutions discovered by the BC-IG_{QL} and BC-BIH_{QL} algorithms will be added to it.

4.4.4.1 Bi-Criteria Iterated Greedy Algorithm with Q-Learning

This thesis combines the Q-learning method with the IG algorithm, both of which were

described in earlier sections. Figure 4.3 depicts the overall structure of the suggested BC-IG_{QL} algorithm, incorporating its perturbation mechanism based on Q-learning.

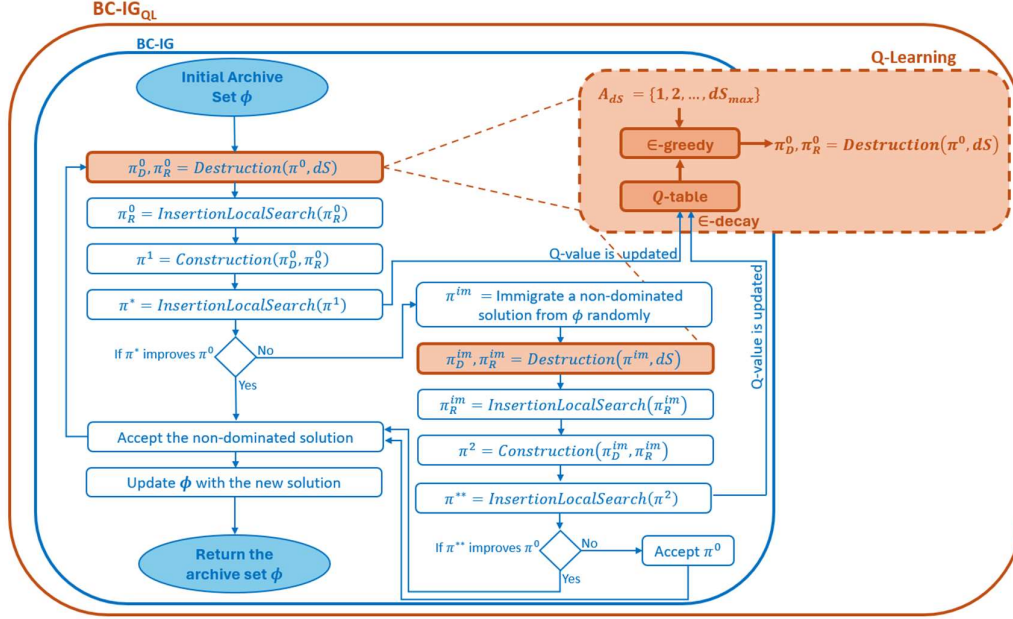


Figure 4.3. The overall structure of the Bi-Criteria IG algorithm with Q-learning

Source: Yüksel et al. (2024)

Figure 4.3 demonstrates that the BC-IG_{QL} algorithm encompasses four primary components: initial archive set generation, perturbation mechanism as the destruction-construction phase, local search, and acceptance criteria of the non-dominated solutions. Finally, it returns the non-dominated archive set. Notably, Q-learning has been incorporated into the perturbation mechanism. Q-learning is also integrated into the solutions that have immigrated randomly from the archive set. This figure is generated for the BC-IG_{QL} as inspired by the single objective version of the algorithm provided by Karimi-Mamaghan et al. (2023).

In the destruction phase of the BC-IG_{QL} algorithm, dS jobs are taken off an individual π^0 and collected in π_D^0 . The residuary jobs are saved in π_R^0 . After the destruction, an insertion local search (ILS), illustrated in Algorithm 4.2, is implemented on the partial solution π_R^0 to optimize the destroyed partial solutions. Regarding construction, we insert each job in π_D^0 is into π_R^0 . Because of the problem's multi-criteria nature, the

dominance rule of two solution points is carried out while checking that non-dominance. It's worth noting that while each complete solution is evaluated using the dominance rule (Deb, K. (2011)), similarly, we interpret the partial solutions using the dominance rule of partial solutions. In the end, we acquire a complete solution by choosing the non-dominant solutions within n solutions after inserting the last withdrawn job for n positions. Then, we re-employed the ILS for the complete solution required after the construction step to improve the quality of the complete solution further. Once an individual π^0 is dominated by the new solution π^* , the reward function of π^* is computed as follows.

$$Reward = 1/(C_{max}(\pi^*) + \sum_{j \in N} C_j(\pi^*)) \quad (4.18)$$

As the goal is to maximize the Q-value (cumulative reward), achieving a smaller value for $C_{max} + \sum_{j \in N} C_j$ leads to a more excellent reward value. When the current solution is enhanced within the inner do-while loop, as seen in Algorithm 4.1, each parameter's action yields an increased reward. Conversely, if the current solution doesn't get enhanced, the performed action of each parameter results in a diminished reward due to the increased $C_{max} + \sum_{j \in N} C_j$ value. The inspiration for the reward function is drawn from the research studies conducted by Oztop et al. (2020) and Öztop et al. (2022). They used a reward function of $1/C_{max}$ in their Q-learning-based IG algorithms to reduce the makespan in no-idle PFSPs. They illustrated its effectiveness compared to other existing algorithms.

Subsequently, using this immediate reward r and considering the historical performance of the operators, the Q-learning algorithm allocates credit to each operator. Namely, the Q-table is updated for the previous action using the reward value r . Depending on the ongoing state of the search and the credit assigned to each operator, the next perturbation operator to be employed is determined.

The Q-value is computed for every parameter-action pair within the Q-learning function. Equation (4.19) was formulated by substituting the parameter value for the state (s) value in Equation (4.16). In the BC-IG_{QL} algorithm, the perturbation parameter considered the state of the Q-learning is dS . The actions in this context refer to various perturbation operators with varying degrees of strength that can be applied during each search process step. To be more specific, a range of different values for

dS is regarded as the set of actions denoted as A_{dS} , expressed as follows: $A_{dS} = \{1, 2, \dots, dS_{max}\}$ (where dS_{max} is the maximum number of jobs that can be removed during the destruction phase). The objective behind varying the values of dS is to generate a set of perturbation operators characterized by different levels of exploration capabilities. As dS grows larger, more jobs are taken out of and reinserted into the job sequence, resulting in a more pronounced perturbation effect on the solution.

Our Q-learning function in BC-IG_{QL} is as follows.

$$Q(dS, a_{dS}) := Q(dS, a_{dS}) + \theta \left[r + \gamma \max_{a'_{dS} \in A_{dS}} Q(dS', a'_{dS}) - Q(dS, a_{dS}) \right], \quad (4.19)$$

where we assume that there is only one state for the parameter dS , a_{dS} is the action that is taken in the state dS , a'_{dS} is the possible action in the state dS' , θ is the learning rate of new information ($0 \leq \theta \leq 1$), r is the reward obtained after acting a_{dS} , and γ is the discount factor of influences of the future reward of $\max_{a'_{dS} \in A_{dS}} Q(dS', a'_{dS})$.

Finally, we update the archive set ϕ by π^* . Suppose the individual is not dominated by the new solution π^* . In that case, immigrate a non-dominated solution from the archive set ϕ randomly and apply destruction & construction and ILS. If we acquire a new non-dominated solution, the reward function and the Q-learning values of the new non-dominated solution are calculated for the new solution. Then, we update the archive set ϕ .

The epsilon-decay strategy is applied to move from exploration to exploitation gradually. A random action is selected, with a probability of ϵ , that is jumping probability to escape from local optima, promoting exploration. The action with the maximum Q-value is selected with a probability of $1-\epsilon$, prioritizing exploitation. By incorporating the ϵ -greedy strategy, the Q-learning algorithm can balance exploring new actions and exploiting the knowledge gained from previously explored actions. This approach ensures a more comprehensive solution to space exploration, increasing the chances of discovering optimal or near-optimal solutions. The complete pseudocode of the BC-IG_{QL} is provided in Algorithm 4.1.

Algorithm 4.1. Bi-Criteria Iterated Greedy Algorithm with Q-Learning

dS will be learned by Q-learning; $NP = 1$

$dS := \text{RandomChoice}(A^{dS})$

Construct the single initial solution π^0 by NEH heuristic

Update the archive set ϕ with π^0

Compute delay matrix d_{ij}

While (NotTermination) **do**

$\pi_D^0, \pi_R^0 = \text{Destruction}(\pi^0, dS)$ //remove dS jobs (π_D^0) from π^0 to form π_R^0

$\pi_R^0 = \text{InsertionLocalSearch}(\pi_R^0)$ //IG_{ALL}

$\pi^1 = \text{Construction}(\pi_D^0, \pi_R^0)$ //insert jobs of π_D^0 to their best position in π_R^0

$\pi^* = \text{InsertionLocalSearch}(\pi^1)$

if ($\pi^* > \pi^0$) **then do**

$r = 1/(C_{\max}(\pi^*) + \sum_{j \in N} C_j(\pi^*))$ //reward function

$\pi^0 = \pi^*$

// update Q-Table for the previous action using the reward value r .

$Q(dS, a_{dS}) := Q(dS, a_{dS}) + \theta \left[r + \gamma \max_{a'_{dS} \in A_{dS}} Q(dS', a'_{dS}) - Q(dS, a_{dS}) \right]$

Update the archive set ϕ with π^0

elseif

$\pi^{im} = \text{Immigrate a nondominated solution from } \phi \text{ randomly}$

$\pi_D^{im}, \pi_R^{im} = \text{Destruction}(\pi^{im}, dS)$

$\pi_R^{im} = \text{InsertionLocalSearch}(\pi_R^{im})$

$\pi^2 = \text{Construction}(\pi_D^{im}, \pi_R^{im})$

$\pi^{**} = \text{InsertionLocalSearch}(\pi^2)$

if ($\pi^{**} > \pi^0$)

$r = 1/(C_{\max}(\pi^{**}) + \sum_{j \in N} C_j(\pi^{**}))$

$\pi^0 = \pi^{**}$

$Q(dS, a_{dS}) := Q(dS, a_{dS}) + \theta \left[r + \gamma \max_{a'_{dS} \in A_{dS}} Q(dS', a'_{dS}) - Q(dS, a_{dS}) \right]$

elseif

$\pi^0 = \pi^0$

Update the archive set ϕ with π^0

endif

```

endif
 $\epsilon := \epsilon \cdot \beta$  // apply  $\epsilon$ -decay to gradually move from exploration to exploitation
if ( $\text{rand} \geq \epsilon$ ) then do //Choose an action using the  $\epsilon$ -greedy strategy
     $dS' := \text{argmax}_{a''_{dS} \in A_{dS}} Q(dS', a''_{dS})$  //The subsequent action is decided
    by Q-Table
elseif
     $dS' := \text{RandomChoice}(A_{dS})$  //The subsequent action for  $dS$  is decided
    randomly
endif
endwhile
return archive set  $\phi$ 
endprocedure

```

A productive insertion local search (ILS) is carried out as a local search in algorithms, as shown in Algorithm 4.2. For each job j , the job at π_p is taken out of the solution π^1 . Next, we insert π_p into n different places. A non-dominant solution π^* from n number of insertions is acquired. It is replaced with π^* , if it dominates individual π^0 .

Algorithm 4.2. Bi-Criteria Insertion Local Search Algorithm

```

for  $j = 1$  to  $n$  do
     $\pi_p$  = Remove a job randomly from  $\pi^1$  at position  $p$ 
     $\pi^*$  = Insert job  $\pi_p$  in Dominating Position in  $\pi^1$ 
    if ( $\pi^* > \pi^0$ ) then do
         $\pi^0 = \pi^*$ 
    end if
end for
return  $\pi^1$ 
endprocedure

```

4.4.4.2 Bi-Criteria Block Insertion Heuristic Algorithm with Q-Learning

The Bi-Criteria Block Insertion Heuristic with Q-learning (BC-BIH_{QL}) is the other algorithm that Q-learning guides. This thesis uses the Q-learning algorithm to implement Adaptive Operator Selection for the Block Size operator within the BIH algorithm. Figure 4.4 depicts the overall structure of the suggested BC-BIH_{QL} algorithm, incorporating its perturbation mechanism based on Q-learning.

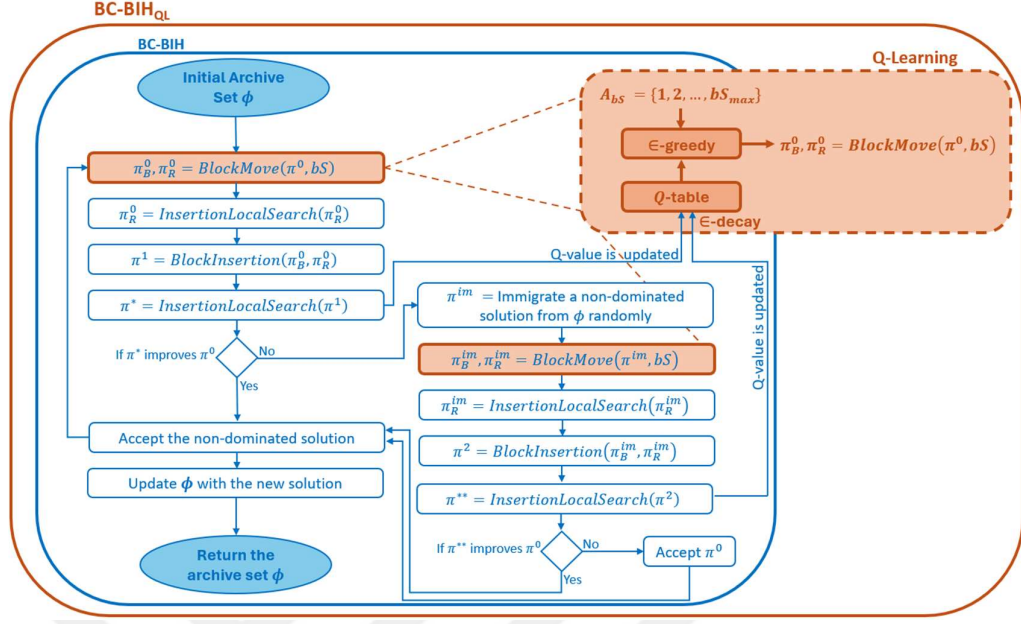


Figure 4.4. The overall structure of the Bi-Criteria BIH algorithm with Q-learning
Source: Yüksel et al. (2024)

Figure 4.4 demonstrates that the BC-BIH_{QL} algorithm encompasses four primary components: initial archive set generation, perturbation mechanism as the block move and block insertion phase, local search, and acceptance criteria of the non-dominated solutions. Ultimately, it concludes by returning the non-dominated archive set. The Q-learning is integrated into the perturbation mechanism throughout this procedure. Moreover, Q-learning is incorporated into solutions randomly selected from the archive set.

The Q-learning tries to learn the block size bS . A block bS of jobs is randomly taken from the individual π^0 and we keep them in π_B^0 in BC-BIH_{QL} algorithm. The residuary jobs are also held in π_R^0 . Then, the ILS, as illustrated in Algorithm 4.2, is performed on the partial solution π_R^0 . Then, $n - bS + 1$ block insertion movements are executed. Namely, we insert the block π_B^0 to the partial solution π_R^0 , successively. We acquire a complete solution by choosing the set of non-dominant solutions among $n - bS + 1$ solutions once the block has been put into the final position of the partial solution's π_R^0 . In the end, the unique ILS (Algorithm 4.2) mechanism is re-implemented to the complete solution assessed by the movements of block insertion. Given that the individual π^0 is dominated by the new solution π^* , the reward function of π^* is

computed, and the Q-table is updated for the previous action using the reward value r . In the BC-BIH_{QL} algorithm, the perturbation parameter considered in the state of Q-learning is bS . The actions referred to the range of different values for bS and denoted as $A_{bS} = \{1, 2, \dots, bS_{max}\}$ (where bS_{max} is the maximum number of jobs that can be removed during the block move phase).

Equation (4.19) of the BC-IG_{QL} is replaced by Equation (4.20) in the BC-BIH_{QL} algorithm.

$$Q(bS, a_{bS}) := Q(bS, a_{bS}) + \theta \left[r + \gamma \max_{a'_{bS} \in A_{bS}} Q(bS', a'_{bS}) - Q(bS, a_{bS}) \right] \quad (4.20)$$

where we posit that there exists only a state for parameter bS , a_{bS} is the action that is taken in the state bS , a'_{bS} is the possible action in the state bS' , θ is the learning rate of new information ($0 \leq \theta \leq 1$), r is the reward obtained after acting a_{bS} , and γ is the discount factor of influence of the future reward of $\max_{a'_{bS} \in A_{bS}} Q(bS', a'_{bS})$.

The residual of the algorithm is the same as in the BC-IG_{QL} (Algorithm 4.1). The complete pseudocode of the Bi-Criteria Block Insertion Heuristic with Q-learning (BC-BIH_{QL}) is provided in Algorithm 4.3.

Algorithm 4.3. Bi-Criteria Block Insertion Heuristic Algorithm with Q-Learning

bS will be learned by Q-learning; $NP = 1$

$bS := \text{RandomChoice}(A_{bS})$

Construct the single initial solution π^0 by NEH heuristic

Update the archive set ϕ with π^0

Compute delay matrix d_{ij}

While (*NotTermination*) **do**

$\pi_B^0, \pi_R^0 = \text{Block Move}(\pi^0, bS)$ //remove a block of bS jobs (π_B^0) from π^0 to form π_R^0

$\pi_R^0 = \text{InsertionLocalSearch}(\pi_R^0)$

$\pi^1 = \text{BlockInsertion}(\pi_B^0, \pi_R^0)$ //insert job block of π_B^0 to their best position in π_R^0

$\pi^* = \text{InsertionLocalSearch}(\pi^1)$ to $n - bS + 1$ positions

if ($\pi^* > \pi^0$) **then do**

$r = 1/(C_{\max}(\pi^*) + \sum_{j \in N} C_j(\pi^*))$ //reward function

$\pi^0 = \pi^*$

// update Q-Table for the previous action using the reward value r .

$Q(bS, a_{bS}) := Q(bS, a_{bS}) + \theta \left[r + \gamma \max_{a'_{bS} \in A_{bS}} Q(bS', a'_{bS}) - Q(bS, a_{bS}) \right]$

Update the archive set ϕ with π^0

elseif

$\pi^{im} = \text{Immigrate a nondominated solution from } \phi \text{ randomly}$

$\pi_B^{im}, \pi_R^{im} = \text{BlockMove}(\pi^{im}, bS)$

$\pi_B^{im} = \text{InsertionLocalSearch}(\pi_R^{im})$

$\pi^2 = \text{BlockInsertion}(\pi_B^{im}, \pi_R^{im})$

$\pi^{**} = \text{InsertionLocalSearch}(\pi^2)$

if ($\pi^{**} > \pi^0$)

$r = 1/(C_{\max}(\pi^{**}) + \sum_{j \in N} C_j(\pi^{**}))$

$\pi^0 = \pi^{**}$

$Q(bS, a_{bS}) := Q(bS, a_{bS}) + \theta \left[r + \gamma \max_{a'_{bS} \in A_{bS}} Q(bS', a'_{bS}) - Q(bS, a_{bS}) \right]$

elseif

$\pi^0 = \pi^0$

Update the archive set ϕ with π^0

```

endif
endif
 $\epsilon := \epsilon \cdot \beta$  // apply epsilon-decay to gradually move from exploration to exploitation
if ( $\text{rand} \geq \epsilon$ ) then d //Choose an action using the  $\epsilon$ -greedy strategy
     $bS' := \operatorname{argmax}_{a''_{bS} \in A_{bS}} Q(bS', a''_{bS})$  //The subsequent action is decided by
    Q-Table
elseif
     $bS' := \operatorname{RandomChoice}(A_{bS})$  //The subsequent action for  $bS$  is decided
    randomly
endif
endwhile
return archive set  $\phi$ 
endprocedure

```

4.4.5. Bi-Criteria Genetic Local Search Algorithms

The multi-objective genetic local search algorithm was introduced into the literature by Ishibuchi and Murata (1996), and the application of the algorithm to the flowshop scheduling problems is explained by the same authors in Ishibuchi and Murata (1998). This hybrid approach merges a multi-objective genetic algorithm (Ishibuchi and Murata, 1996) with an adapted local search procedure. We adopted this algorithm as a bi-criteria genetic local search algorithm for our problem. The makespan and the total flow time are two objectives in calculating the fitness function, the weighted sum of two. The pseudocode of the algorithm is provided in Algorithm 4.4.

Algorithm 4.4. Bi-Criteria Genetic Local Search Algorithm (BC-GLS)

set $dS = 4; NP = 30$

Construct parent population with size NP randomly (P)

Update the archive ϕ with NP individuals

Compute delay matrix d_{ij}

while (*NotTermination*) **do**

for $i = 1$ to NP **do**

 Sort the P in an increasing order

$CD_P =$ Calculate the crowding distances of P

$y =$ Randomly choose an individual from P ($i \neq y$)

 //Generate a child Q_i

$Q_i = \text{Crossover}(P_i, P_y)$ //Crossover Operator (Five variants of crossover operator are studied)

 Sort the Q in increasing order

$CD_Q =$ Calculate the crowding distances of Q

endfor

for $i = 1$ to NP **do**

if ($Q_i > P_i$) **then do**

$P_i = Q_i$

 Update the archive ϕ with P_i

elseif

$Q_i == P_i$

endif

if ($CD_P > CD_Q$)

$P_i = P_i$

elseif

$P_i = Q_i$

endif

endfor

 // Mutation

$nds =$ Randomly immigrate a nondominated solution from ϕ

$a, b, c =$ Randomly selected individuals from P

$nds = \text{InsertionLocalSearch}(nds)$

$P_a = nds$

$nds = \text{InsertionLocalSearch}(nds)$

```

 $P_b = nds$ 
 $nds = InsertionLocalSearch(nds)$ 
 $P_c = nds$ 
endwhile
endprocedure

```

As crossover operators, we have carried out five different crossovers: one-cut crossover (Mitchell, M. (1998)), two-cut crossover (Eshelman et al. (1989); Weise, T. (2009)), order crossover (Davis, L. (1994)), partially mapped crossover (Goldberg and Lingle (2014); Eiben and Smith (2015)), and particle crossover operator (Pan et al. (2008)). Therefore, five genetic local search algorithm variants are applied to the problem:

- Bi-Criteria Genetic Local Search Algorithm with One-cut Crossover (BC-GLS_{ONECUT}),
- Bi-Criteria Genetic Local Search Algorithm with Two-cut Crossover (BC-GLS_{TWOCUT}),
- Bi-Criteria Genetic Local Search Algorithm with Order Crossover (BC-GLS_{OX}),
- Bi-Criteria Genetic Local Search Algorithm with Partially Mapped Crossover (BC-GLS_{PMX}),
- Bi-Criteria Genetic Local Search Algorithm with Particle Crossover (BC-GLS_{PTL}),

Furthermore, the insertion phase is implemented as illustrated in Algorithm 4.2.

4.5. Computational Analysis of Mathematical Model and Algorithms

An extensive computational analysis is conducted for the performance assessment of the MILP model formulation and metaheuristic algorithms. The instances of Vallada et al. (2015) are employed as a benchmark set. The benchmark set - by Vallada et al. (2015) - consists of 24 collections of small problems and 24 collections of large problems. Each collection has ten instances. Small instances vary in the number of jobs to 10, 20, 30, 40, 50, and 60, and in machines 5, 10, 15, and 20. Large instances vary in the number of jobs to 100, 200, 300, 400, 500, 600, 700, and 800 and in the number of machines 20, 40, and 60. Therefore, we carry out 480 problem instances of

Vallada et al. (2015) through this study. Hereafter, the set of instances of Vallada et al. (2015) will be called *VRF benchmark problems*.

Initially, the MILP model is run by Mavrotas's augmented-epsilon constraint algorithm. However, the augmented epsilon-constraint algorithm cannot solve all instances in polynomial time. The augmented-epsilon constraint algorithm of Mavrotas (2009) runs by fixing the epsilon value to 10^{-2} for the first 40 instances of VRF instances (10-5, 10-10, 10-15, and 10-20 instances) in Optimization Studio of IBM ILOG CPLEX. These sets are solvable in polynomial time limits. In terms of mathematical modeling, the rest of the instances are executed by the augmented-epsilon constraint method with a specific time limit. A time limit of 9 minutes is established for each iteration in each instance of the remaining benchmark set. Consequently, each instance is executed for 3 hours, calculated as 20 iterations multiplied by 9 minutes ($20 * 9 \text{ min} = 180 \text{ min}$).

Then, the metaheuristics algorithms were programmed in C++ of Microsoft Visual Studio 2019. The experiment is performed on a computer with a Core i7 processor running at 2.60 GHz and 16 GB of RAM. Each instance was replicated 30 times. All algorithms were executed for $25nm$ milliseconds for the first four sets of VRF instances and $50nm$ milliseconds on small and large VRF benchmark problems in each replication. dS value of the BC-IG and BC-IG_{ALL} algorithms is set as 2 on the first four VRF instances and 4 on the remaining instances. The bS_{max} value of the BC-VBIH algorithm is set as 4 on the first four sets of VRF instances and 8 on the remaining instances. The Q-learning parameters of BC-IG_{QL} and BC-BIH_{QL} algorithms have been set as $\epsilon := 0.8$, $\beta := 0.996$, $\alpha := 0.6$, and $\gamma := 0.8$. These values have been determined by the parameter tuning study by Karimi-Mamaghan et al. (2023). Hence, we have adopted these values for our Q-learning-guided algorithms. In the BC-IG_{QL} algorithm, the action list of dS is taken as $A_{dS} \in \{1,2,3\}$ and in the BC-BIH_{QL} algorithm, the action list of bS is set as $A_{bS} \in \{1,2,3\}$.

The following performance indicators were used to interpret the BC-IG_{QL}, BC-BIH_{QL}, BC-GLS, BC-IG, BC-IG_{ALL}, and BC-VBIH algorithms' solution quality over the Pareto optimal set found by the MILP model. More importantly, the set of the Pareto optimal solutions of the augmented epsilon-constraint algorithm is compared with the

set of non-dominant solutions of metaheuristics.

- **Cardinality:**

The quantity of non-dominant solutions discovered: $R_p = |X|$.

- **The proportion of the Pareto-optimal solutions discovered:**

$$R_{pX} = \frac{|X \cap \text{Pareto}|}{|\text{Pareto}|},$$

- **Inverted generational distance:**

$IGD_X = \sum_{g \in P} \frac{d(g, X)}{|P|}$, in which the shortest Euclidean distance between two solutions is $d(g, X)$ (Coello et al. (2007)).

- **Distribution Spacing:**

$DS_X = \left[\frac{1}{|X|} \sum_{i \in X} (d_i - \bar{d})^2 \right]^{1/2} / \bar{d}$, where $\bar{d} = \sum_{i \in X} d_i / |X|$ and d_i indicates solution i 's shortest Euclidean distance from its nearest neighbor in X (Tan et al. (2006)). When low spacing values are encountered, the solutions in X are considered evenly distributed. Here, X is associated with the non-dominant solutions heuristic algorithms achieve.

To simplify the following computational tables, the computational results of different variants of the BC-GLS algorithms are provided at <https://data.mendeley.com/datasets/fhm8b3xyr2/1>. Only the BC-GLS_{PMX} variant, deemed the most effective, is retained in the thesis. Tables 4.7 and 4.8 summarize the comparison performance metrics of algorithms on the first 40 instances of the VRF dataset since they can be solved optimally by the MILP model.

Table 4.7. Computational Analysis for Cardinality on the First Four Sets of VRF Instances

Instance Set	BC-MILP	BC-IG _{QL}	BC-BIH _{QL}	BC-GLS _{PMX}	BC-IG	BC-IG _{ALL}	BC-VBIH
10-5	7.30	7.30	7.30	7.30	7.30	7.30	7.30
10-10	10.00	10.00	10.00	10.00	10.00	10.00	10.00
10-15	7.60	7.60	7.60	7.60	7.60	7.60	7.60
10-20	5.50	5.50	5.50	5.50	5.50	5.50	5.50
Average	7.60	7.60	7.60	7.60	7.60	7.60	7.60

Table 4.8. Computational Analysis for Proportion of Pareto Optimal Solution Set Discovered, Inverted General Distance, and Distribution Spacing on the First Four Sets of VRF Instances

Instance	BC-IG _{QL}			BC-BIH _{QL}		
Set	$R_{p_{IG_{QL}}}$	$IGD_{IG_{QL}}$	$DS_{IG_{QL}}$	$R_{p_{BIH_{QL}}}$	$IGD_{BIH_{QL}}$	$DS_{BIH_{QL}}$
10-5	1.000	0.000	0.749	1.000	0.000	0.749
10-10	1.000	0.000	0.825	1.000	0.000	0.825
10-15	1.000	0.000	0.676	1.000	0.000	0.676
10-20	1.000	0.000	0.483	1.000	0.000	0.483
Average	1.000	0.000	0.683	1.000	0.000	0.683
Instance	BC-GLS _{PMX}			BC-IG		
Set	$R_{p_{GLS_{PMX}}}$	$IGD_{GLS_{PMX}}$	$DS_{GLS_{PMX}}$	$R_{p_{IG}}$	IGD_{IG}	DS_{IG}
10-5	1.000	0.000	0.749	1.000	0.000	0.749
10-10	1.000	0.000	0.825	1.000	0.000	0.825
10-15	1.000	0.000	0.676	1.000	0.000	0.676
10-20	1.000	0.000	0.483	1.000	0.000	0.483
Average	1.000	0.000	0.683	1.000	0.000	0.683
Instance	BC-IG _{ALL}			BC-VBIH		
Set	$R_{p_{IG_{ALL}}}$	$IGD_{IG_{ALL}}$	$DS_{IG_{ALL}}$	$R_{p_{VBIH}}$	IGD_{VBIH}	DS_{VBIH}
10-5	1.000	0.000	0.749	1.000	0.000	0.749
10-10	1.000	0.000	0.825	1.000	0.000	0.825
10-15	1.000	0.000	0.676	1.000	0.000	0.676
10-20	1.000	0.000	0.483	1.000	0.000	0.483
Average	1.000	0.000	0.683	1.000	0.000	0.683

As demonstrated in Tables 4.7 and 4.8, all algorithms achieve the Pareto optimal solution set discovered by the MILP model on the first 40 instances of the VRF dataset. Furthermore, the 0 IGD values mean that all algorithms precisely acquire the exact set of Pareto optimal solutions. The low spacing value (0.683) implies the uniform divergence of the points of the set of the Pareto frontier.

After revealing that all algorithms reach 100% of the Pareto optimal solutions for instances that are solvable in polynomial time, the remaining instances are executed with the proposed Q-learning guided algorithms (BC-IG_{QL}, BC-BIH_{QL}) and the

algorithms for comparison (BC-GLS_{ONECUT}, BC-GLS_{TWOCUT}, BC-GLS_{SOX}, BC-GLS_{PMX}, and BC-GLS_{PTL}, BC-IG, BC-IG_{ALL} and BC-VBIH). In terms of mathematical modeling, the MILP model is executed with the augmented-epsilon constraint method with a time limit on each iteration of TFT grids for both benchmark sets. TFT's range is divided by 20 equal grids, and each iteration's time limit is set as 9 minutes, yielding that each instance is executed by 180 minutes. The metaheuristics were executed for 50nm milliseconds in each replication. Metaheuristics produce different non-dominated solution sets. To depict the performance evaluation of the metaheuristic algorithms, the non-dominated solution sets produced by all algorithms (BC-IG_{QL}, BC-BIH_{QL}, BC-GLS_{ONECUT}, BC-GLS_{TWOCUT}, BC-GLS_{SOX}, BC-GLS_{PMX}, BC-GLS_{PTL}, BC-IG, BC-IG_{ALL} and BC-VBIH) on the 50-05-01 instance of the small VRF set and the 400-60-02 instance of large VRF are illustrated in Figures 4.5 and 4.6, respectively.

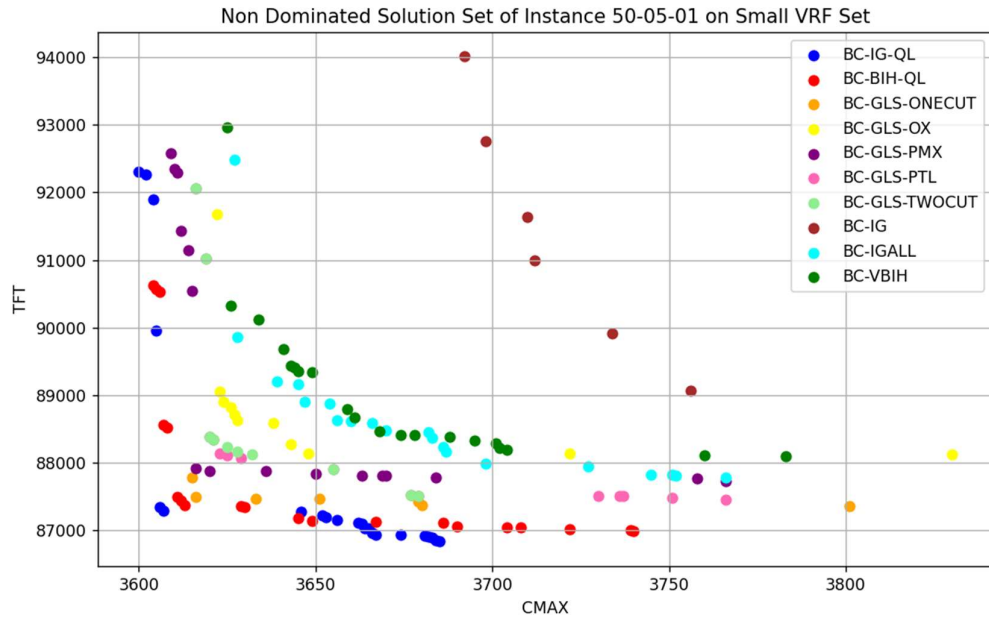


Figure 4.5. Non-Dominated Solution Set of Instance 50-05-01 on the Small VRF Instances

Source: Yüksel et al. (2024)

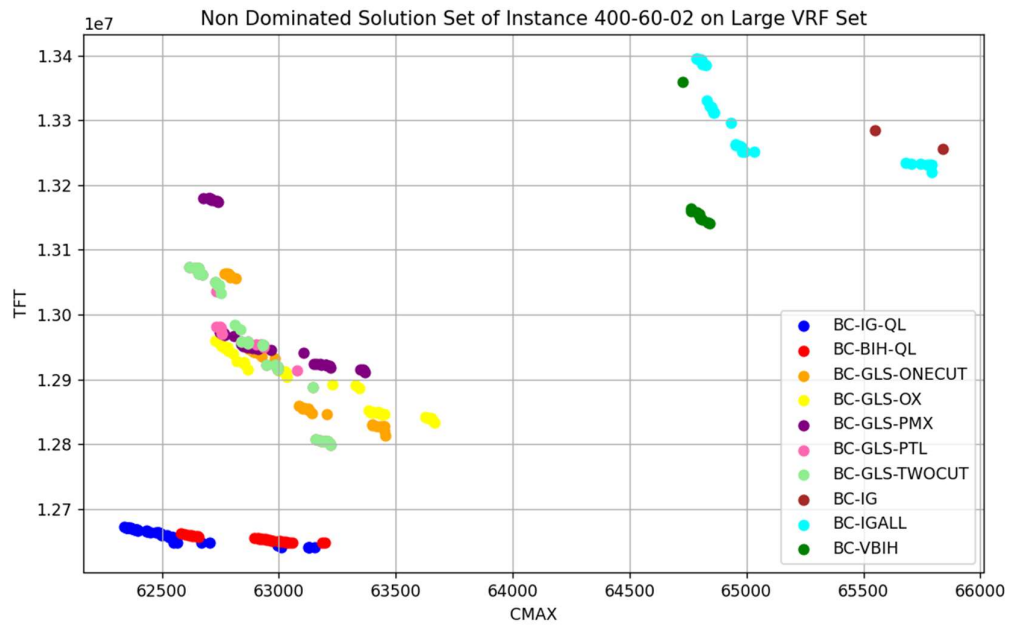


Figure 4.6. Non-Dominated Solution Set of Instance 400-60-02 on the Large VRF Instances

Source: Yüksel et al. (2024)

As is seen from Figures 4.5-4.6, the makespan increases while the total flow time decreases and vice versa. In other words, neither can be minimized simultaneously due to the trade-offs between the two objectives. More importantly, Tables 4.9-4.18 are good illustrative examples to present the superiority of the BC-IG_{QL} and BC-BIH_{QL} algorithms over other algorithms on both small and large VRF Benchmark sets, explicitly illustrated in the further explanations below.

Performance indicators for cardinality and distribution spacing are also utilized to compare performance on the large benchmark set. Additionally, one more performance indicator, coverage of two sets, is used to interpret the solution excellence of BC-IG_{QL}, BC-BIH_{QL}, and other algorithms.

- **Coverage of Two Sets** (Ruben and Stützle (2007)) :

$C(X, Y) = \frac{|\{y \in Y; \exists x \in X: x \succcurlyeq y\}|}{|Y|}$, where the value of $C(X, Y)$ is 1 if 100% of solutions of Y are dominated weakly by certain solutions of X .

The benchmark suite analyzes each algorithm's cardinality performance indicator, and Tables 4.9 and 4.10 report these results for the small VRF instances and the large VRF instances, respectively.

Table 4.9. Computational Analysis for Cardinality Metric on Small VRF Instances

Instance Set	 BC-MILP 	 BC-IG_{QL} 	 BC-BIH_{QL} 	 BC-GLS_{PMX} 	 BC-IG 	 BC-IG_{ALL} 	 BC-VBIH
10-5	7.30	7.30	7.30	7.30	7.30	7.30	7.30
10-10	10.00	10.00	10.00	10.00	10.00	10.00	10.00
10-15	7.60	7.60	7.60	7.60	7.60	7.60	7.60
10-20	5.50	5.50	5.50	5.50	5.50	5.50	5.50
20-5	2.00	14.30	14.40	14.40	9.90	14.40	14.40
20-10	2.00	12.10	12.10	12.10	8.70	12.10	12.10
20-15	2.00	14.60	14.60	14.60	12.60	14.60	14.60
20-20	2.40	11.50	11.50	11.50	8.80	11.50	11.50
30-5	1.90	21.40	21.00	20.30	9.80	18.60	19.80
30-10	1.80	19.50	19.70	18.80	7.10	19.60	19.20
30-15	2.00	18.40	18.60	18.50	6.80	19.20	19.20
30-20	1.80	19.90	20.00	20.40	8.10	20.40	20.00
40-5	1.70	23.20	23.70	18.90	7.00	14.20	13.30
40-10	1.60	22.00	23.50	21.10	9.00	19.60	18.90
40-15	1.80	22.70	23.00	21.40	8.80	19.90	17.10
40-20	1.90	19.10	19.80	19.80	7.30	20.00	19.90
50-5	1.50	26.10	22.00	15.90	7.90	14.50	16.50
50-10	2.00	21.00	19.50	17.70	5.70	16.00	13.60
50-15	1.70	20.30	20.20	18.70	7.30	14.10	13.40
50-20	1.70	29.70	30.30	28.60	9.50	19.70	16.40
60-5	1.80	17.90	17.80	12.70	7.40	14.30	15.70
60-10	1.30	22.10	20.00	18.80	7.10	12.20	14.90
60-15	1.40	23.20	22.40	15.40	6.90	13.60	15.70
60-20	1.60	20.30	20.20	19.60	7.30	11.30	13.90
Average	2.76	17.90	17.70	16.23	8.00	14.59	14.60

Table 4.10. Computational Analysis for Cardinality Metric on Large VRF Instances

Instance Set	BC-IG _{QL}	BC-BIH _{QL}	BC-GLS _{PMX}	BC-IG	BC-IG _{ALL}	BC-VBIH
100-20	24.80	20.90	18.30	6.30	13.10	16.80
100-40	25.20	28.60	21.50	8.10	16.00	17.80
100-60	25.80	31.60	20.70	7.90	17.40	18.00
200-20	23.10	25.80	19.00	3.90	15.90	17.80
200-40	32.80	29.00	22.00	5.30	20.50	23.30
200-60	31.10	30.60	21.20	5.10	20.70	18.70
300-20	22.60	30.00	27.80	3.40	19.80	16.50
300-40	32.70	25.80	30.60	3.00	20.00	21.80
300-60	30.80	37.40	32.50	4.10	16.40	21.30
400-20	31.10	31.20	25.70	4.50	18.60	20.40
400-40	24.50	25.20	35.70	3.90	16.50	17.80
400-60	29.10	32.00	33.70	3.50	22.00	17.50
500-20	29.90	36.60	32.60	3.40	20.60	25.00
500-40	37.20	36.00	33.50	3.60	22.10	20.10
500-60	32.30	28.10	33.20	3.20	25.90	17.80
600-20	44.00	32.80	28.90	3.50	24.30	25.30
600-40	45.10	44.00	26.70	2.70	21.30	19.70
600-60	40.10	38.50	34.00	2.40	16.70	24.70
700-20	42.80	45.40	24.20	2.80	23.90	28.80
700-40	51.40	48.10	32.30	3.20	20.70	19.20
700-60	46.00	49.60	44.20	3.30	18.70	33.20
800-20	44.10	44.60	33.80	3.10	21.90	30.70
800-40	43.80	62.10	36.00	2.70	19.60	20.90
800-60	47.20	52.70	31.10	2.70	21.80	21.80
Average	34.90	36.11	29.13	3.98	19.77	21.45

As shown in Table 4.9, the average cardinality of the MILP for the small VRF instances is 2.76. Moreover, the average cardinality of the BC-IG_{QL}, BC-BIH_{QL}, BC-GLS_{PMX}, BC-IG, BC-IG_{ALL}, and BC-VBIH algorithms for the small VRF instances are provided as 17.90, 17.70, 16.23, 8.00, 14.59, and 14.60 in Table 4.9, respectively. This means the BC-IG_{QL} and BC-BIH_{QL} algorithms outperform all other algorithms

regarding non-dominated solutions' cardinality for the small VRF instances. On average, they find 1.141 times the number of non-dominant solutions than the remaining algorithms.

The average cardinality of the BC-IG_{QL}, BC-BIH_{QL}, BC-GLS_{PMX}, BC-IG, BC-IG_{ALL}, and BC-VBIH algorithms for the large VRF instances are provided as 34.90, 36.11, 29.13, 3.98, 19.77, and 21.45 in Table 4.10, respectively. This implies that the BC-IG_{QL} and BC-BIH_{QL} algorithms again outperform all other algorithms regarding non-dominated solutions' cardinality for the large VRF instances. BC-IG_{QL} and BC-BIH_{QL} algorithms find 1.110 times the number of non-dominant solutions than the remaining algorithms on average.

Each algorithm's distribution spacing performance indicator is analyzed on the benchmark suite. Tables 4.11 and 4.12 report these results for the small VRF instances and large VRF instances, respectively.

Table 4.11. Computational Analysis for Distribution Metric on Small VRF Instances

Instance Set	$DS_{BC-IG_{QL}}$	$DS_{BC-BIH_{QL}}$	$DS_{BC-GLS_{PMX}}$	DS_{BC-IG}	$DS_{BC-IG_{ALL}}$	$DS_{BC-VBIH}$
10-5	0.749	0.749	0.749	0.749	0.749	0.749
10-10	0.825	0.825	0.825	0.825	0.825	0.825
10-15	0.676	0.676	0.676	0.676	0.676	0.676
10-20	0.483	0.483	0.483	0.483	0.483	0.483
20-5	1.110	1.298	1.298	1.178	1.298	1.298
20-10	0.966	0.966	0.966	0.951	0.966	0.966
20-15	0.810	0.810	0.810	1.055	0.810	0.810
20-20	1.077	1.077	1.077	1.147	1.077	1.077
30-5	1.306	1.553	1.276	1.106	1.291	2.009
30-10	1.583	1.289	1.417	1.048	1.376	1.488
30-15	1.042	1.080	1.007	0.815	1.024	1.003
30-20	1.386	1.229	1.209	1.185	1.212	1.201
40-5	1.469	1.679	1.916	0.845	1.394	1.474
40-10	1.528	1.570	1.297	0.959	1.252	1.333
40-15	1.298	1.219	1.394	0.728	1.365	1.371
40-20	1.354	1.468	1.261	0.723	1.384	1.691
50-5	1.797	1.386	1.487	0.976	1.503	1.630
50-10	1.472	1.154	1.540	0.891	1.621	1.117
50-15	1.524	1.453	1.000	0.838	1.227	1.735
50-20	1.439	1.526	1.278	0.918	1.347	1.316
60-5	1.682	1.552	1.314	1.150	1.403	1.281
60-10	1.174	1.598	0.942	0.995	1.663	1.708
60-15	1.375	1.277	1.439	0.922	1.205	1.466
60-20	1.081	1.497	1.120	0.836	1.269	1.417
Average	1.217	1.226	1.158	0.917	1.184	1.255

Table 4.12. Computational Analysis for Distribution Metric on Large VRF Instances

Instance Set	$DS_{BC-IG_{QL}}$	$DS_{BC-BIH_{QL}}$	$DS_{BC-GLS_{PMX}}$	DS_{BC-IG}	$DS_{BC-IG_{ALL}}$	$DS_{BC-VBIH}$
100-20	1.412	1.366	1.402	0.963	1.673	1.067
100-40	1.495	1.011	1.334	1.047	1.295	1.450
100-60	1.241	1.309	1.354	1.053	1.321	1.523
200-20	1.310	1.508	1.307	0.572	1.462	1.497
200-40	1.625	1.635	1.351	0.428	1.486	1.552
200-60	1.377	2.001	1.687	0.701	1.679	1.380
300-20	1.635	1.811	1.725	0.599	1.331	1.394
300-40	1.448	1.405	1.662	0.665	1.763	1.083
300-60	1.370	1.636	1.906	0.449	1.942	1.408
400-20	1.852	1.833	1.675	0.640	1.500	1.281
400-40	1.997	1.626	2.060	0.585	1.635	1.632
400-60	1.607	1.924	1.893	0.272	1.466	1.579
500-20	1.755	1.708	1.702	0.568	1.410	1.757
500-40	2.237	2.335	1.699	0.499	1.542	1.777
500-60	1.762	2.029	1.628	0.207	1.396	1.362
600-20	3.143	2.067	1.907	0.446	1.418	1.735
600-40	2.288	2.841	1.675	0.397	1.379	1.695
600-60	1.755	1.609	1.957	0.266	1.181	1.350
700-20	2.077	2.357	1.677	0.222	1.727	1.517
700-40	2.215	2.644	1.605	0.413	1.535	1.581
700-60	2.147	1.681	2.176	0.456	1.687	1.739
800-20	1.953	1.975	1.672	0.354	1.280	1.507
800-40	2.164	2.467	1.554	0.106	1.763	1.623
800-60	2.450	2.051	2.115	0.251	1.532	1.464
Average	1.846	1.868	1.697	0.507	1.517	1.498

According to Tables 4.11 and 4.12, all algorithms have lower values concerning the distribution spacing metric. The lower DS value indicates that the solutions found by all algorithms are uniformly distributed in both small and large VRF benchmark problems. To assess the distribution of the solution set, this performance metric is combined with the average quality (Tan et al. (2006)).

The coverage performance metric of each algorithm is analyzed on the VRF benchmark suites. Tables 4.13, 4.14, and 4.15 report these results for small VRF instances, and Tables 4.16, 4.17, and 4.18 report for large VRF instances, respectively.

Table 4.13. Computational Analysis for Coverage Metric on Small VRF Instances

Instance Set	C(BC-BIH _{QL} , BC-IG _{QL})	C(BC-IG _{QL} , BC-BIH _{QL})	C(BC-BIH _{QL} , BC-GLS _{PMX})	C(BC-GLS _{PMX} , BC-BIH _{QL})	C(BC-IG _{QL} , BC-GLS _{PMX})	C(BC-GLS _{PMX} , BC-IG _{QL})
10-5	1.000	1.000	1.000	1.000	1.000	1.000
10-10	1.000	1.000	1.000	1.000	1.000	1.000
10-15	1.000	1.000	1.000	1.000	1.000	1.000
10-20	1.000	1.000	1.000	1.000	1.000	1.000
20-5	1.000	0.993	1.000	1.000	0.993	1.000
20-10	1.000	1.000	1.000	1.000	1.000	1.000
20-15	1.000	1.000	1.000	1.000	1.000	1.000
20-20	1.000	1.000	1.000	1.000	1.000	1.000
30-5	0.903	0.876	0.864	0.742	0.851	0.717
30-10	0.964	0.971	0.962	0.918	0.987	0.949
30-15	0.986	0.980	0.949	0.975	0.944	0.973
30-20	0.968	0.986	0.964	1.000	0.982	1.000
40-5	0.643	0.597	0.808	0.262	0.819	0.331
40-10	0.822	0.813	0.760	0.734	0.811	0.725
40-15	0.925	0.895	0.865	0.754	0.866	0.771
40-20	0.916	0.866	0.854	0.871	0.783	0.887
50-5	0.449	0.487	0.909	0.041	0.908	0.071
50-10	0.590	0.667	0.694	0.325	0.630	0.317
50-15	0.682	0.701	0.602	0.605	0.587	0.569
50-20	0.631	0.712	0.645	0.430	0.708	0.429
60-5	0.533	0.416	0.988	0.000	0.914	0.079
60-10	0.665	0.398	0.882	0.063	0.757	0.200
60-15	0.660	0.313	0.787	0.135	0.691	0.157
60-20	0.372	0.596	0.732	0.218	0.712	0.241
Average	0.821	0.803	0.886	0.670	0.873	0.684

Table 4.14. Computational Analysis for Coverage Metric on Small VRF Instances

Instance	C(BC-BIH_{QL}, BC-IG)	C(BC-IG, BC-BIH_{QL})	C(BC-IG_{QL}, BC-IG)	C(BC-IG, BC-IG_{QL})	C(BC-BIH_{QL}, BC-IG_{ALL})	C(BC-IG_{ALL}, BC-BIH_{QL})
10-5	1.000	1.000	1.000	1.000	1.000	1.000
10-10	1.000	1.000	1.000	1.000	1.000	1.000
10-15	1.000	1.000	1.000	1.000	1.000	1.000
10-20	1.000	1.000	1.000	1.000	1.000	1.000
20-5	1.000	0.174	1.000	0.175	1.000	1.000
20-10	1.000	0.331	1.000	0.331	1.000	1.000
20-15	1.000	0.412	1.000	0.412	1.000	1.000
20-20	1.000	0.678	1.000	0.678	1.000	1.000
30-5	1.000	0.000	1.000	0.000	0.903	0.595
30-10	1.000	0.000	1.000	0.000	0.973	0.965
30-15	1.000	0.005	1.000	0.005	0.917	1.000
30-20	1.000	0.000	1.000	0.000	0.969	1.000
40-5	1.000	0.000	1.000	0.000	0.955	0.030
40-10	1.000	0.000	1.000	0.000	0.920	0.461
40-15	1.000	0.000	1.000	0.000	0.904	0.699
40-20	1.000	0.000	1.000	0.000	0.871	0.805
50-5	1.000	0.000	1.000	0.000	1.000	0.000
50-10	1.000	0.000	1.000	0.000	0.996	0.010
50-15	1.000	0.000	1.000	0.000	0.943	0.073
50-20	1.000	0.000	1.000	0.000	0.881	0.173
60-5	1.000	0.000	1.000	0.000	1.000	0.000
60-10	1.000	0.000	1.000	0.000	1.000	0.000
60-15	1.000	0.000	1.000	0.000	1.000	0.000
60-20	1.000	0.000	1.000	0.000	0.981	0.003
Average	1.000	0.233	1.000	0.233	0.967	0.576

Table 4.15. Computational Analysis for Coverage Metric on Small VRF Instances

Instance	C(BC-IG _{QL} , BC-IG _{ALL})	C(BC-IG _{ALL} , BC-IG _{QL})	C(BC-BIH _{QL} , BC-VBIH)	C(BC-VBIH, BC-BIH _{QL})	C(BC-IG _{QL} , BC-VBIH)	C(BC-VBIH, BC-IG _{QL})
10-5	1.000	1.000	1.000	1.000	1.000	1.000
10-10	1.000	1.000	1.000	1.000	1.000	1.000
10-15	1.000	1.000	1.000	1.000	1.000	1.000
10-20	1.000	1.000	1.000	1.000	1.000	1.000
20-5	0.993	1.000	1.000	1.000	0.993	1.000
20-10	1.000	1.000	1.000	1.000	1.000	1.000
20-15	1.000	1.000	1.000	1.000	1.000	1.000
20-20	1.000	1.000	1.000	1.000	1.000	1.000
30-5	0.896	0.600	0.935	0.500	0.911	0.504
30-10	0.972	0.961	0.967	0.938	0.972	0.941
30-15	0.909	0.997	0.914	0.997	0.909	0.997
30-20	0.982	0.995	0.985	0.955	0.982	0.935
40-5	0.965	0.018	0.984	0.016	0.971	0.014
40-10	0.900	0.435	0.928	0.215	0.936	0.207
40-15	0.856	0.704	0.973	0.344	0.925	0.337
40-20	0.825	0.786	0.902	0.501	0.868	0.512
50-5	1.000	0.000	1.000	0.000	1.000	0.000
50-10	1.000	0.000	1.000	0.000	0.995	0.000
50-15	0.994	0.078	0.987	0.000	0.987	0.003
50-20	0.902	0.147	0.920	0.082	0.973	0.025
60-5	1.000	0.000	1.000	0.000	1.000	0.000
60-10	1.000	0.000	1.000	0.000	1.000	0.000
60-15	1.000	0.000	1.000	0.000	1.000	0.000
60-20	0.970	0.014	1.000	0.000	1.000	0.000
Average	0.965	0.572	0.979	0.523	0.976	0.520

Table 4.16. Computational Analysis for Coverage Metric on Large VRF Instances

Instance	C(BC-BIH_{QL}, BC-IG_{QL})	C(BC-IG_{QL}, BC-BIH_{QL})	C(BC-BIH_{QL}, BC-GLS_{PMX})	C(BC-GLS_{PMX}, BC-BIH_{QL})	C(BC-IG_{QL}, BC-GLS_{PMX})	C(BC-GLS_{PMX}, BC-IG_{QL})
100-20	0.324	0.553	0.848	0.058	0.894	0.078
100-40	0.487	0.477	0.410	0.454	0.485	0.338
100-60	0.394	0.534	0.492	0.333	0.446	0.402
200-20	0.441	0.448	0.975	0.000	1.000	0.000
200-40	0.409	0.511	1.000	0.000	1.000	0.000
200-60	0.377	0.561	1.000	0.000	1.000	0.000
300-20	0.158	0.653	0.997	0.000	1.000	0.000
300-40	0.665	0.264	0.974	0.000	0.984	0.000
300-60	0.375	0.568	1.000	0.000	1.000	0.000
400-20	0.214	0.716	0.759	0.086	0.799	0.080
400-40	0.470	0.408	0.886	0.009	0.851	0.051
400-60	0.182	0.777	0.907	0.073	0.959	0.000
500-20	0.404	0.475	0.606	0.172	0.617	0.175
500-40	0.488	0.418	0.706	0.124	0.677	0.056
500-60	0.419	0.564	0.750	0.029	0.694	0.044
600-20	0.627	0.345	0.307	0.482	0.287	0.407
600-40	0.419	0.526	0.396	0.239	0.407	0.283
600-60	0.458	0.500	0.571	0.077	0.475	0.090
700-20	0.354	0.573	0.120	0.736	0.183	0.659
700-40	0.448	0.562	0.410	0.266	0.477	0.206
700-60	0.506	0.429	0.476	0.251	0.495	0.170
800-20	0.475	0.517	0.296	0.446	0.254	0.374
800-40	0.395	0.640	0.396	0.247	0.456	0.291
800-60	0.393	0.483	0.324	0.239	0.271	0.218
Average	0.412	0.521	0.650	0.180	0.655	0.163

Table 4.17. Computational Analysis for Coverage Metric on Large VRF Instances

Instance	C(BC-BIH _{QL} , BC-IG)	C(BC-IG, BC-BIH _{QL})	C(BC-IG _{QL} , BC-IG)	C(BC-IG, BC-IG _{QL})	C(BC-BIH _{QL} , BC-IG _{ALL})	C(BC-IG _{ALL} , BC-BIH _{QL})
100-20	1.000	0.000	1.000	0.000	1.000	0.000
100-40	1.000	0.000	1.000	0.000	1.000	0.000
100-60	1.000	0.000	1.000	0.000	1.000	0.000
200-20	1.000	0.000	1.000	0.000	1.000	0.000
200-40	1.000	0.000	1.000	0.000	1.000	0.000
200-60	1.000	0.000	1.000	0.000	1.000	0.000
300-20	1.000	0.000	1.000	0.000	1.000	0.000
300-40	1.000	0.000	1.000	0.000	1.000	0.000
300-60	1.000	0.000	1.000	0.000	1.000	0.000
400-20	1.000	0.000	1.000	0.000	1.000	0.000
400-40	1.000	0.000	1.000	0.000	1.000	0.000
400-60	1.000	0.000	1.000	0.000	1.000	0.000
500-20	1.000	0.000	1.000	0.000	1.000	0.000
500-40	1.000	0.000	1.000	0.000	1.000	0.000
500-60	1.000	0.000	1.000	0.000	1.000	0.000
600-20	1.000	0.000	1.000	0.000	1.000	0.000
600-40	1.000	0.000	1.000	0.000	1.000	0.000
600-60	1.000	0.000	1.000	0.000	1.000	0.000
700-20	1.000	0.000	1.000	0.000	1.000	0.000
700-40	1.000	0.000	1.000	0.000	1.000	0.000
700-60	1.000	0.000	1.000	0.000	1.000	0.000
800-20	1.000	0.000	1.000	0.000	0.858	0.000
800-40	1.000	0.000	1.000	0.000	1.000	0.000
800-60	1.000	0.000	1.000	0.000	1.000	0.000
Average	1.000	0.000	1.000	0.000	0.994	0.000

Table 4.18. Computational Analysis for Coverage Metric on Large VRF Instances

Instance	C(BC-IG _{QL} , BC-IG _{ALL})	C(BC-IG _{ALL} , BC-IG _{QL})	C(BC-BIH _{QL} , BC-VBIH)	C(BC-VBIH, BC-BIH _{QL})	C(BC-IG _{QL} , BC-VBIH)	C(BC-VBIH, BC-IG _{QL})
10-5	1.000	0.000	1.000	0.000	1.000	0.000
10-10	1.000	0.000	1.000	0.000	1.000	0.000
10-15	1.000	0.000	1.000	0.000	1.000	0.000
10-20	1.000	0.000	1.000	0.000	1.000	0.000
20-5	1.000	0.000	1.000	0.000	1.000	0.000
20-10	1.000	0.000	1.000	0.000	1.000	0.000
20-15	1.000	0.000	1.000	0.000	1.000	0.000
20-20	1.000	0.000	1.000	0.000	1.000	0.000
30-5	1.000	0.000	1.000	0.000	1.000	0.000
30-10	1.000	0.000	1.000	0.000	1.000	0.000
30-15	1.000	0.000	1.000	0.000	1.000	0.000
30-20	1.000	0.000	1.000	0.000	1.000	0.000
40-5	1.000	0.000	1.000	0.000	1.000	0.000
40-10	1.000	0.000	1.000	0.000	1.000	0.000
40-15	1.000	0.000	1.000	0.000	1.000	0.000
40-20	1.000	0.000	1.000	0.000	1.000	0.000
50-5	1.000	0.000	1.000	0.000	1.000	0.000
50-10	1.000	0.000	1.000	0.000	1.000	0.000
50-15	0.986	0.000	0.939	0.000	1.000	0.000
50-20	1.000	0.000	1.000	0.000	1.000	0.000
60-5	1.000	0.000	1.000	0.000	1.000	0.000
60-10	0.872	0.000	0.767	0.000	0.830	0.000
60-15	1.000	0.000	1.000	0.000	1.000	0.000
60-20	1.000	0.000	1.000	0.000	1.000	0.000
Average	0.994	0.000	0.988	0.000	0.993	0.000

Regarding the measure of coverage, the BC-BIH_{QL} algorithm dominates 82.1% of the solutions found by the BC-IG_{QL} algorithm. In contrast, the BC-IG_{QL} algorithm dominates 80.3% of the solutions found by the BC-BIH_{QL} algorithm in the small VRF benchmark problems, as illustrated in Table 4.13. Furthermore, the BC-BIH_{QL}

algorithm dominates 41.2% of the solutions found by the BC-IG_{QL} algorithm, and it is 52.1% the other around in the large VRF benchmark problems, as depicted in Table 4.16. This implies that the BC-BIH_{QL} and BC-IG_{QL} algorithms are pretty competitive on both parts of the VRF benchmark suite. Thus, the *competitiveness of two Q-learning-guided algorithms with each other* has been illustrated.

In the comparison of the BC-BIH_{QL} algorithm and the other algorithms, the BC-BIH_{QL} algorithm dominates 88.6%, 100.00%, 96.7%, and 97.9% of the solutions reached by the BC-GLS_{PMX}, BC-IG, BC-IG_{ALL}, and BC-VBIH algorithms, respectively, in the small VRF benchmark suite as stated in Tables 4.13, 4.14, and 4.15. The coverage significantly decreases when moving in the opposite direction of the intended coverage comparison. A similar trend is also seen in comparing the BC-IG_{QL} algorithm with others. The BC-IG_{QL} algorithm dominates 87.3%, 100.0%, 96.5%, and 97.6% of the solutions reached by the BC-GLS_{PMX}, BC-IG, BC-IG_{ALL}, and BC-VBIH algorithms, respectively, in the small VRF benchmark suite as stated in Tables 4.13, 4.14, and 4.15. The coverage is notably reduced when making a comparison in the opposite direction.

In the large VRF benchmark suite, as stated in Table 4.16, 4.17, and 4.18, the BC-BIH_{QL} algorithm dominates 65.0%, 100.00%, 99.4%, and 98.8% of the solutions reached by the BC-GLS_{PMX}, BC-IG, BC-IG_{ALL}, and BC-VBIH algorithms, respectively. The coverage is substantially less in the opposite direction of coverage. Again, the same pattern is observed between the BC-IG_{QL} and other algorithms. The BC-IG_{QL} algorithm dominates 65.5%, 100.00%, 99.4%, and 99.3% of the solutions reached by the BC-GLS_{PMX}, BC-IG, BC-IG_{ALL}, and BC-VBIH algorithms, respectively, in the large VRF benchmark suite as stated in Table 4.16, 4.17, and 4.18. The coverage is significantly lower in the reverse comparison. This approach demonstrates *the competitiveness of Q-learning-guided algorithms compared to existing scheduling algorithms* documented in the literature.

It is essential to mention that integrating Q-learning into the algorithms helps improve the quality of the non-dominated solutions found in the algorithms without Q-learning, as seen in Tables 4.13-4.18. The BC-IG_{QL} algorithm dominates almost 97% and 99% of the solutions reached by the BC-IG, BC-IG_{ALL}, and BC-VBIH algorithms in small and large datasets. Similarly, The BC-BIH_{QL} algorithm dominates almost 98% and

99% of the solutions reached by the BC-IG, BC-IG_{ALL}, and BC-VBIH algorithms in small and large datasets. Briefly, among all the features that have been compared, the Q-learning-guided algorithms demonstrate the highest level of competitiveness compared with the ones without Q-learning. By doing so, *the contribution of Q-learning over IG and BIH algorithms, with and without Q-learning*, has been illustrated.

To sum up, the effectiveness of the designed Q-learning guided algorithms in terms of various perspectives has been proved and demonstrated to the readers. Among all the features that have been compared, the Q-learning-guided algorithms demonstrate the highest level of competitiveness in both segments of the VRF benchmark suite concerning cardinality, distribution spacing, and coverage metrics. More significantly, the BC-BIH_{QL} and BC-IG_{QL} algorithms markedly dominate the solutions reached by all algorithms by cardinality and coverage metrics within both sections of the VRF benchmark suite. This means that BC-BIH_{QL} and BC-IG_{QL} algorithms outperform the variants of the BC-GLS algorithm, BC-IG, BC-IG_{ALL}, and BC-VBIH algorithms for part of both diversion and quality of the non-dominant solutions discovered in a holistic view.

Chapter 4 addresses Research Questions 6 and 7 of this thesis.

Research Question 6: Can we extend these studies in a *bi-objective framework* in such a way that two service levels are carried out simultaneously; i.e., $(F_m|nwt|ServiceLevel_1, SerciveLevel_2)$?

Research Question 7: Can we create *Q-learning guided metaheuristic algorithms* for the bi-objective $(F_m|nwt|ServiceLevel_1, SerciveLevel_2)$ problem?

The NWPFSF is treated as a ***bi-criteria combinatorial optimization*** problem where total flow time and makespan are optimized simultaneously in Chapter 4. Thus, this chapter discusses the $(F_m|nwt|C_{max}, \sum C_i)$ problem, addressing Research Questions 6 and 7 as follows. A bi-criteria MILP model is presented to illustrate the ***trade-off between the objectives***, and then ***Q-learning guided algorithms*** are developed for $(F_m|nwt|C_{max}, \sum C_i)$ problem. The computational results are evaluated using the ***VRF*** dataset.

As the research output from this section, I had the opportunity to present the preliminary results of this chapter at the 15th International Conference on Multi-objective Programming and Goal Programming 2023 (MOPGP 2023 Conference), which occurred in a hybrid format from October 2th to 4th, 2023, in Izmir, Türkiye. The abstract of this presentation is published in the MOPGP 2023 Book of Abstracts (Yüksel et al. (2023)). The title of the abstract is “***Bi-Criteria Optimization of Makespan and Total Flow Time in No-Wait Flowshop***”, and it is accessible online at the following link: <https://mopgp.org/wp-content/uploads/2023/09/Mopgp-Book-2023.pdf>.

Subsequently, the complete computational results of this chapter were published in the Journal of Swarm and Evolutionary Computation (Yüksel et al. (2024)) on the 28th of May 2024. The title of the article is “***Q-Learning Guided Algorithms For Bi-Criteria Minimization Of Total Flow Time And Makespan In No-Wait Permutation Flowshops***” and accessible online at the following link:

<https://www.sciencedirect.com/science/article/abs/pii/S221065022400155X>.

I also had a chance to give a seminar on this article at the University of Twente on the 21st of May 2024. The seminar is organized by the Department of Industrial Engineering and Business Information Systems (IEBIS). The announcement of the seminar is accessible on the following link:

<https://www.utwente.nl/en/bms/iebis/seminar/240521/>.

5. CHAPTER: DISCUSSION, CONCLUSION AND FUTURE RESEARCH DIRECTIONS

A range of solution methods exist to tackle combinatorial optimization problems. Significant approaches encompass the creation of lower bounding schemes, mathematical model formulations such as Mixed-Integer Linear Programming (MILP) and Constraint Programming (CP), upper bounds, and metaheuristics. These solution methods establish the basis for numerous viewpoints in the field. This thesis tackled several essential research questions within NWPFSPs, with a particular emphasis on developing new solution techniques for $(F_m|nwt|ServiceLevel)$ problems.

The outlined research questions of this thesis were as follows:

Research Questions:

1. Can we generate more effective mathematical models (MILP and CP model formulations) for the $(F_m|nwt|ServiceLevel)$ problems?
2. Can we enhance the mathematical models developed for the $(F_m|nwt|ServiceLevel)$ problems by incorporating valid inequalities?
3. Can we generate new lower bounds for the $(F_m|nwt|ServiceLevel)$ problems?
4. Can we develop effective heuristics as quick upper bounds for the $(F_m|nwt|ServiceLevel)$ problems?
5. Can we create a due date generation mechanism and establish strong due dates to address lateness-related performance measures in $(F_m|nwt|ServiceLevel)$ problems?
6. Can we extend these studies in a bi-objective framework in such a way that two service levels are carried out simultaneously; i.e., $(F_m|nwt|ServiceLevel_1, ServiceLevel_2)$?
7. Can we create Q-learning guided metaheuristic algorithms for the bi-objective $(F_m|nwt|ServiceLevel_1, ServiceLevel_2)$ problem?

This thesis contributes to NWPFSs in two main areas: *single-objective* optimization and *bi-criteria* optimization. For *single-objective* optimization (Chapter 3), new solution techniques are developed to separately minimize makespan (maximum completion time) (C_{max}), the total flow time ($\sum C_i$), the total tardiness ($\sum T_i$), and the number of tardy jobs ($\sum U_i$) in NWPFSs. For *bi-criteria* optimization (Chapter 4), innovative solution techniques are implemented to minimize both makespan simultaneously (C_{max}), and total flow time ($\sum C_i$) in the BI-CRI NWPFS. The instance sets from Taillard, E. (1993) and Vallada et al. (2015) are used as benchmark problems throughout this thesis.

Sections 3.2 and 3.3 tackle *Research Question 1* of this thesis. These sections focus on the problems $(F_m|nwt|C_{max})$, $(F_m|nwt|\sum C_i)$, and $(F_m|nwt|\sum T_i)$, addressing Research Question 1 by introducing *five new mathematical models* to the literature on NWPFSs. The computational results for these problems are presented using the *Taillard* and *VRF* datasets. Among the models, Model 3 and Model 4 stand out as the most effective, delivering superior results for makespan, total flow time, and total tardiness. They are proven to be highly efficient in flowshops, providing high-quality feasible solutions within a one-hour time limit.

Sections 3.4 and 3.5 address *Research Question 2* of this thesis. In these sections, the problems $(F_m|nwt|C_{max})$, $(F_m|nwt|\sum C_i)$, and $(F_m|nwt|\sum T_i)$ are discussed, focusing on the introduction of new *Chvátal-Gomory valid inequalities* to improve the performance of a mathematical model for NWPFSs. These inequalities were tested on the *Taillard* dataset for the mentioned problems. The computational results reveal that incorporating these valid inequalities significantly enhanced the model's performance, notably reducing the gap percentage in the first three sets of instances.

Sections 3.6 and 3.7 address *Research Question 3* of this thesis. These sections explore the problems $(F_m|nwt|C_{max})$, $(F_m|nwt|\sum C_i)$, $(F_m|nwt|\sum T_i)$, and $(F_m|nwt|\sum U_i)$. They present two Shortest Path guided *Lower Bounds* algorithms — *LB₁* and *LB₁-Lifted* — developed to enhance solutions for these problems. The computational results, based on the *Taillard* and *VRF* datasets, demonstrate that the lifting mechanism significantly improves the performance of *LB₁*. It effectively reduces gaps in makespan, total flow time, total tardiness, and the number of tardy jobs across these benchmark problems.

Section 3.8 tackles *Research Questions 1, 4, and 5* of this thesis. In this section, the focus is on the $(F_m | nwt | \sum U_i)$ problem. It examines both an *MILP* and a *CP* model for this problem. A *novel quick upper bound*, the Sacrifice and Rearrange Heuristic, is introduced to provide high-quality feasible solutions rapidly. Additionally, a *new due date generation mechanism*, based on this heuristic, is proposed to address the lateness-related performance measures. The generated due dates are tight, resulting in approximately half of the tasks being completed tardy. Hence, they can be efficiently utilized for additional research on assessing performance metrics related to lateness in NWPFSs. Indeed, this mechanism presents a set of demanding and challenging benchmark problems known for their tight due dates in the literature of NWPFSs. The computational results of this section are presented for the *Taillard* and *VRF* datasets. This thesis assesses the Sacrifice and Rearrange Heuristic's performance across various construction heuristics. Our results show that, on average, this heuristic provides a 20.91% improvement compared to the construction heuristics when utilized with the *Taillard* benchmark problems. Remarkably, it achieves an even more substantial enhancement of 49.05% on average over the construction heuristics when applied to the *VRF* benchmark problems. The SRH surpasses 101 out of 140 instances regarding the objective value, corresponding to 72.1% of the Taillard instances. It also outperforms 133 out of 240 instances regarding the objective value, which accounts for 55.4% of the VRF set with new due dates. In conclusion, the SRH outperforms the mathematical models by rapidly obtaining reasonable feasible solutions, consistently delivering superior objective function values in a few seconds for both sets. SRH performs notably better on the VRF benchmark problems than on the Taillard benchmark problems, where it meets challenging benchmark problems with strict due dates. Significantly, in this thesis, the best-known solutions of the VRF benchmark problem set considering new tough due dates are provided for the literature on the number of tardy jobs minimization of NWPFSs.

Chapter 4 tackles Research Questions 6 and 7 of this thesis. It approaches the NWPFS as a *bi-criteria combinatorial optimization problem*, where both total flow time and makespan are optimized simultaneously. This chapter focuses on the $(F_m | nwt | C_{max}, \sum C_i)$ problem, addressing *Research Questions 6 and 7* by presenting a bi-criteria MILP model that demonstrates the *trade-offs between these objectives*. Bi-objective NWPFS's primary purpose is to procure Pareto solutions and

approximate non-dominant solutions for simultaneously optimizing these objectives. Additionally, *Q-learning-guided algorithms* are developed for the $(F_m|nwt|C_{max}, \sum C_i)$ problem, with computational results assessed using the *VRF* dataset. The effectiveness of the Q-learning-guided algorithms has been validated and presented from multiple perspectives. Among the various features compared, these algorithms show the greatest competitiveness within both segments of the VRF benchmark suite, excelling in metrics such as cardinality, distribution spacing, and coverage. Moreover, the BC-IG_{QL} algorithm dominates almost 98% % of the solutions reached by the other algorithms without Q-learning. Similarly, The BC-BIH_{QL} algorithm dominates almost 98.5% of the solutions reached by the other algorithms without Q-learning. Namely, among all the features that have been compared, the Q-learning-guided algorithms (the BC-IG_{QL} and BC-BIH_{QL} algorithms) demonstrate the highest level of competitiveness in a comprehensive assessment of non-dominated solutions.

In summary, the subsequent sections of this thesis address the research questions as follows:

- Sections 3.2 and 3.3 provide answers to Research Question 1.
- Sections 3.4 and 3.5 focus on Research Question 2.
- Sections 3.6 and 3.7 investigate Research Question 3
- Section 3.8 covers Research Questions 1, 4, and 5.
- Chapter 4 addresses Research Questions 6 and 7.

There are several potential future research directions. Regarding *mathematical models*, the focus should be on improving both the objective values and the gap percentages. Given that CP is a promising area of research, enhancing CP models, particularly in terms of constraint generation procedures, would be valuable. Another area for future research is to apply these models to different objective criteria, such as maximum lateness, with or without strict due date restrictions.

Concerning *valid inequalities*, the valid inequality generation procedures can be applied to other mathematical models for NWPFSs. They will eventually enhance the performance of the other models in terms of the run time or percent deviations from the best integer solutions found. The lower and upper bounds can be integrated into the mathematical models and the generated valid inequalities. In that way, the

performance of the valid inequalities may improve much higher. Furthermore, the small set of instances is easily solvable in seconds. Hence, to compare the performance of the valid inequalities more sensitive in terms of the run time, a larger size small set of instances, such as ten jobs, can be studied to experiment with the parameters of the valid inequalities.

Several methods may enhance the performance of the *lower bounds*, given that there are various approaches to breaking the tours within the shortest paths provided by these bounds. The lower bound can be integrated into various solution methods, such as mathematical models and upper-bounding heuristics. Given their effectiveness, these generated lower bounds can also be incorporated into Branch and Bound algorithms.

Regarding the *number of tardy jobs minimization*, the remaining mathematical models can be utilized for this criterion. The proposed Sacrifice and Rearrange Heuristic can also inspire the generation of upper bounds for other lateness-related performance measurements in NWPFSs. This idea can also be an inspiration source for other flowshop scheduling problems in which the total number of tardy jobs is minimized. The novel quick upper bounding method is a shortcut feasible solution generation procedure. More specifically, the Gain Function can be directly integrated with the computation of new upper bounds for performance measurements, such as the weighted number of tardy jobs. The generated due dates create demanding and challenging benchmark problems known for their tight deadlines. Therefore, these benchmarks, which provide valuable insights for future studies focusing on lateness-related performance measures, can be directly used in future studies.

Regarding the *bi-criteria optimization* of NWPFSs, different MILP or CP model constructions can be generated for this problem since there is enough room to enhance the performance of the models. The *Q-learning procedure* might be implemented for other operators of the algorithms and integrated into various multi-criteria algorithms to amplify the performance of the proposed ones. The outcomes of this chapter encourage us to discover many more bi-criteria NWPFSs to reveal the trade-off between other conflicting objectives, such as the makespan & maximum tardiness, mean completion time & maximum earliness, total flow time & number of late/tardy jobs, etc. These problems might be treated with other Q-learning-driven algorithms.

6. REFERENCES

- Aldowaisan, T., & Allahverdi, A. (2003). New heuristics for no-wait flowshops to minimize makespan. *Computers & Operations Research*, 30(8), 1219-1231. [https://doi.org/10.1016/S0305-0548\(02\)00068-0](https://doi.org/10.1016/S0305-0548(02)00068-0)
- Aldowaisan, T., & Allahverdi, A. (2004). New heuristics for m-machine no-wait flowshop to minimize total completion time. *Omega*, 32(5), 345-352. <https://doi.org/10.1016/j.omega.2004.01.004>
- Aldowaisan, T., & Allahverdi, A. (2012a). Minimizing total tardiness in no-wait flowshops. *Foundations of Computing and Decision Sciences*, 37(3), 149. <https://doi.org/10.2478/v10209-011-0009-6>
- Aldowaisan, T., & Allahverdi, A. (2012b). No-wait flowshop scheduling problem to minimize the number of tardy jobs. *The International Journal of Advanced Manufacturing Technology*, 61(1), 311–323. <https://doi.org/10.1007/S00170-011-3659-X>
- Aldowaisan, T., & Allahverdi, A. (2015). No-Wait Flowshops to Minimize Total Tardiness with Setup Times. *Intelligent Control and Automation*, 6(01), 38–44. <https://doi.org/10.4236/ICA.2015.61005>
- Almeida, F. S. D., & Nagano, M. S. (2023). Heuristics to optimize total completion time subject to makespan in no-wait flow shops with sequence-dependent setup times. *Journal of the Operational Research Society*, 74(1), 362-373. <https://doi.org/10.1080/01605682.2022.2039569>
- Allahverdi, A., & Aldowaisan, T. (2004). No-wait flowshops with bicriteria of makespan and maximum lateness. *European Journal of Operational Research*, 152(1), 132-147. [https://doi.org/10.1016/S0377-2217\(02\)00646-X](https://doi.org/10.1016/S0377-2217(02)00646-X)
- Allahverdi, A., & Aydilek, H. (2013). Algorithms for no-wait flowshops with total completion time subject to makespan. *The International Journal of Advanced Manufacturing Technology*, 68, 2237-2251. <https://doi.org/10.1007/s00170-013-4836-x>
- Allahverdi, A., Aydilek, A., & Aydilek, H. (2016). Minimizing the number of tardy jobs on a two-stage assembly flowshop. *Journal of Industrial and Production Engineering*, 33(6), 391-403. <https://doi.org/10.1080/21681015.2016.1151466>

- Allahverdi, A., Aydilek, H., & Aydilek, A. (2018). No-wait flowshop scheduling problem with two criteria; total tardiness and makespan. *European Journal of Operational Research*, 269(2), 590-601. <https://doi.org/10.1016/j.ejor.2017.11.070>
- Allahverdi, A., Aydilek, H., & Aydilek, A. (2022). An algorithm for a no-wait flowshop scheduling problem for minimizing total tardiness with a constraint on total completion time. *International Journal of Industrial Engineering Computations*, 13(1), 43-50. <https://doi.org/10.5267/J.IJIEC.2021.8.003>
- Asefi, H., Jolai, F., Rabiee, M., & Tayebi Araghi, M. E. (2014). A hybrid NSGA-II and VNS for solving a bi-objective no-wait flexible flowshop scheduling problem. *The International Journal of Advanced Manufacturing Technology*, 75, 1017-1033. <https://doi.org/10.1007/s00170-014-6177-9>
- Arabameri, S., & Salmasi, N. (2013). Minimization of weighted earliness and tardiness for no-wait sequence-dependent setup times flowshop scheduling problem. *Computers & Industrial Engineering*, 64(4), 902-916. <https://doi.org/10.1016/j.cie.2012.12.023>
- Aydilek, H., & Allahverdi, A. (2012). Heuristics for no-wait flowshops with makespan subject to mean completion time, *Applied Mathematics and Computation*, 219(1), 351–359. <https://doi.org/10.1016/j.amc.2012.06.024>
- Azaiez, M. N., Gharbi, A., Kacem, I., Makhlouf, Y., & Masmoudi, M. (2021, December). Two-stage no-wait hybrid flow shop scheduling with inter-stage flexibility: A mathematical model. In *2021 IEEE International Conference on Networking, Sensing and Control (ICNSC)* (Vol. 1, pp. 1-5). IEEE. DOI: [10.1109/ICNSC52481.2021.9702162](https://doi.org/10.1109/ICNSC52481.2021.9702162)
- Bao, H., Pan, Q., Ruiz, R., & Gao, L. (2023). A collaborative iterated greedy algorithm with reinforcement learning for energy-aware distributed blocking flow-shop scheduling. *Swarm and Evolutionary Computation*, 83, 101399. <https://doi.org/10.1016/j.swevo.2023.101399>
- Bulfin, R. L., & M'Hallah, R. (2003). Minimizing the weighted number of tardy jobs on a two-machine flow shop. *Computers & Operations Research*, 30(12), 1887-1900. [https://doi.org/10.1016/S0305-0548\(02\)00114-4](https://doi.org/10.1016/S0305-0548(02)00114-4)
- Cai, J., Lei, D., Wang, J., & Wang, L. (2023). A novel shuffled frog-leaping algorithm with reinforcement learning for distributed assembly hybrid flow shop scheduling. *International Journal of Production Research*, 61(4), 1233-1251. <https://doi.org/10.1080/00207543.2022.2031331>
- Carrier, J. (1978). Ordonnancements a contraintes disjonctives. *RAIRO-Operations Research*, 12(4), 333-350.

- Chang, J., Yu, D., Hu, Y., He, W., & Yu, H. (2022). Deep reinforcement learning for dynamic flexible job shop scheduling with random job arrival. *Processes*, 10(4), 760. <https://doi.org/10.3390/pr10040760>
- Chen, W. J. (2009). Minimizing number of tardy jobs on a single machine subject to periodic maintenance. *Omega*, 37(3), 591-599. <https://doi.org/10.1016/j.omega.2008.01.001>
- Chen, R., Yang, B., Li, S., & Wang, S. (2020). A self-learning genetic algorithm based on reinforcement learning for flexible job-shop scheduling problem. *Computers & industrial engineering*, 149, 106778. <https://doi.org/10.1016/J.CIE.2020.106778>
- Choi, H. S., & Lee, D. H. (2009). Scheduling algorithms to minimize the number of tardy jobs in two-stage hybrid flow shops. *Computers & Industrial Engineering*, 56(1), 113-120. <https://doi.org/10.1016/j.cie.2008.04.005>
- Chen, X. L., Li, J. Q., & Xu, Y. (2023). Q-learning based multi-objective immune algorithm for fuzzy flexible job shop scheduling problem considering dynamic disruptions. *Swarm and Evolutionary Computation*, 83, 101414. <https://doi.org/10.1016/j.swevo.2023.101414>
- Coello, C. A. C. (2007). Evolutionary algorithms for solving multi-objective problems. 2nd Edition, Springer New York, NY. <https://doi.org/10.1007/978-0-387-36797-2>
- Dauzère-Pérès, S. (1997, September). An efficient formulation for minimizing the number of late jobs in single-machine scheduling. In *1997 IEEE 6th International Conference on Emerging Technologies and Factory Automation Proceedings, EFTA'97* (pp. 442-445). IEEE.
- Davis, L. (1994). Handbook of Genetic Algorithms, Van Nostrand Reinhold.
- Deb, K. (2011). Multi-objective optimisation using evolutionary algorithms: an introduction. In *Multi-objective evolutionary optimisation for product design and manufacturing* (pp. 3-34). London: Springer London.
- Dhouib, E., Teghem, J., & Loukil, T. (2013). Minimizing the number of tardy jobs in a permutation flowshop scheduling problem with setup times and time lags constraints. *Journal of Mathematical Modelling and Algorithms in Operations Research*, 12, 85-99. <https://doi.org/10.1007/S10852-012-9180-X>
- Ding, J., Song, S., Zhang, R., Gupta, J. N., & Wu, C. (2015). Accelerated methods for total tardiness minimisation in no-wait flowshops. *International Journal of Production Research*, 53(4), 1002-1018. : <https://doi.org/10.1080/00207543.2014.932935>

- Du, Y., Li, J. Q., Chen, X. L., Duan, P. Y., & Pan, Q. K. (2022). Knowledge-based reinforcement learning and estimation of distribution algorithm for flexible job shop scheduling problem. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 7(4), 1036-1050. <https://doi.org/10.1109/TETCI.2022.3145706>
- 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. <https://doi.org/10.1016/j.cor.2016.12.021>
- Eiben, A. E., & Smith, J. E. (2015). Introduction to evolutionary computing. Springer-Verlag Berlin Heidelberg.
- Eshelman, L. J., Caruana, R. A., & Schaffer, J. D. (1989, December). Biases in the crossover landscape. In *Proceedings of the third international conference on Genetic algorithms* (pp. 10-19).
- Fink, A., & Voß, S. (2003). Solving the continuous flow-shop scheduling problem by metaheuristics. *European Journal of Operational Research*, 151(2), 400-414. [https://doi.org/10.1016/S0377-2217\(02\)00834-2](https://doi.org/10.1016/S0377-2217(02)00834-2)
- Gao, M., Gao, K., Ma, Z., & Tang, W. (2023). Ensemble meta-heuristics and Q-learning for solving unmanned surface vessels scheduling problems. *Swarm and Evolutionary Computation*, 82, 101358. <https://doi.org/10.1016/j.swevo.2023.101358>
- Ghassemi Tari, F. G., & Olfat, L. (2014). Heuristic rules for tardiness problem in flow shop with intermediate due dates. *The International Journal of Advanced Manufacturing Technology*, 71, 381-393. <https://doi.org/10.1007/s00170-013-5478-8>
- Gicquel, C., & Minoux, M. (2015). Multi-product valid inequalities for the discrete lot-sizing and scheduling problem. *Computers & Operations Research*, 54, 12-20. <https://doi.org/10.1016/j.cor.2014.08.022>
- Graham, R. L., Lawler, E. L., Lenstra, J. K., & Kan, A. R. (1979). Optimization and approximation in deterministic sequencing and scheduling: a survey. In *Annals of Discrete Mathematics* (Vol. 5, pp. 287-326). Elsevier. [https://doi.org/10.1016/S0167-5060\(08\)70356-X](https://doi.org/10.1016/S0167-5060(08)70356-X)
- Goldberg, D. E., & Lingle, R. (2014, January). Alleles, loci, and the traveling salesman problem. In *Proceedings of the first international conference on genetic algorithms and their applications* (pp. 154-159). Psychology Press.

- González, M. A., & Vela, C. R. (2015). An efficient memetic algorithm for total weighted tardiness minimization in a single machine with setups. *Applied Soft Computing*, 37, 506-518. <https://doi.org/10.1016/j.asoc.2015.07.050>
- Hamdi, I., & Toumi, S. (2019). MILP models and valid inequalities for the two-machine permutation flowshop scheduling problem with minimal time lags. *Journal of Industrial Engineering International*, 15, 223-229. <https://doi.org/10.1007/s40092-019-00331-1>
- Hansen, P., Mladenović, N., & Moreno Perez, J. A. (2010). Variable neighbourhood search: methods and applications. *Annals of Operations Research*, 175, 367-407. <https://doi.org/10.1007/s10479-009-0657-6>
- Hariri, A. M. A., & Potts, C. N. (1989). A branch and bound algorithm to minimize the number of late jobs in a permutation flow-shop. *European Journal of Operational Research*, 38(2), 228-237. [https://doi.org/10.1016/0377-2217\(89\)90108-2](https://doi.org/10.1016/0377-2217(89)90108-2)
- Hassan, M. A., Kacem, I., Martin, S., & Osman, I. M. (2016, April). Valid inequalities for unrelated parallel machines scheduling with precedence constraints. In *2016 International Conference on Control, Decision and Information Technologies (CoDIT)* (pp. 677-682). IEEE.
- Heller, J. (1960). Some numerical experiments for an $M \times J$ flow shop and its decision-theoretical aspects. *Operations Research*, 8(2), 178-184. <https://doi.org/10.1287/opre.8.2.178>
- Hu, Z., Gong, W., Pedrycz, W., & Li, Y. (2023). Deep reinforcement learning assisted co-evolutionary differential evolution for constrained optimization. *Swarm and Evolutionary Computation*, 83, 101387. <https://doi.org/10.1016/j.swevo.2023.101387>
- Ishibuchi, H., & Murata, T. (1996, May). Multi-objective genetic local search algorithm. In *Proceedings of IEEE international conference on evolutionary computation* (pp. 119-124). IEEE. <https://doi.org/10.1109/ICEC.1996.542345>
- Ishibuchi, H., & Murata, T. (1998). A multi-objective genetic local search algorithm and its application to flowshop scheduling. *IEEE transactions on systems, man, and cybernetics, part C (applications and reviews)*, 28(3), 392-403. <https://doi.org/10.1109/5326.704576>
- Johnson, S. M. (1954). Optimal two- and three-stage production schedules with setup times included. *Naval Research Logistics Quarterly*, 1(1), 61-68. <https://doi.org/10.1002/nav.3800010110>
- Javadi, B., Saidi-Mehrabad, M., Haji, A., Mahdavi, I., Jolai, F., & Mahdavi-Amiri, N. (2008). No-wait flow shop scheduling using fuzzy multi-objective linear

- programming. *Journal of the Franklin Institute*, 345(5), 452-467.
<https://doi.org/10.1016/J.JFRANKLIN.2007.12.003>
- Jenabi, M., Naderi, B., & Ghomi, S. F. (2010, September). A bi-objective case of no-wait flowshops. In *2010 IEEE Fifth International Conference on Bio-Inspired Computing: Theories and Applications (BIC-TA)* (pp. 1048-1056). IEEE.
<https://doi.org/10.1109/BICTA.2010.5645110>
- Jolai, F. (2005). Minimizing number of tardy jobs on a batch processing machine with incompatible job families. *European Journal of Operational Research*, 162(1), 184-190. <https://doi.org/10.1016/j.ejor.2003.10.011>
- Jolai, F., Asefi, H., Rabiee, M., & Ramezani, P. (2013). Bi-objective simulated annealing approaches for no-wait two-stage flexible flow shop scheduling problem. *Scientia Iranica*, 20(3), 861-872.
<https://doi.org/10.1016/j.scient.2012.10.044>
- Kaczmarczyk, W. (2020). Valid inequalities for proportional lot-sizing and scheduling problem with fictitious microperiods. *International Journal of Production Economics*, 219, 236-247. <https://doi.org/10.1016/j.ijpe.2019.06.005>
- Kaelbling, L. P., Littman, M. L., & Moore, A. W. (1996). Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4, 237-285.
<https://doi.org/10.1613/JAIR.301>
- Karimi-Mamaghan, M., Mohammadi, M., Padeloup, B., & Meyer, P. (2023). Learning to select operators in meta-heuristics: An integration of Q-learning into the iterated greedy algorithm for the permutation flowshop scheduling problem. *European Journal of Operational Research*, 304(3), 1296-1330.
<https://doi.org/10.1016/J.EJOR.2022.03.054>
- Keha, A. B., Khowala, K., & Fowler, J. W. (2009). Mixed integer programming formulations for single machine scheduling problems. *Computers & Industrial Engineering*, 56(1), 357-367. <https://doi.org/10.1016/j.cie.2008.06.008>
- Keskin, K., & Engin, O. (2021). A hybrid genetic local and global search algorithm for solving no-wait flow shop problem with bi criteria. *SN Applied Sciences*, 3(6), 628. <https://doi.org/10.1007/S42452-021-04615-3>
- Khalili, M. (2014). A multi-objective electromagnetism algorithm for a bi-objective hybrid no-wait flowshop scheduling problem. *The International Journal of Advanced Manufacturing Technology*, 70(9), 1591-1601.
<https://doi.org/10.1007/s00170-013-5376-0>
- Khalili, M. (2012). Multi-objective no-wait hybrid flowshop scheduling problem with transportation times. *International Journal of Computational Science and Engineering*, 7(2), 147-154. <https://doi.org/10.1504/IJCSE.2012.048094>

- Khalili, M., & Tavakkoli-Moghaddam, R. (2012). A multi-objective electromagnetism algorithm for a bi-objective flowshop scheduling problem. *Journal of Manufacturing Systems*, 31(2), 232-239. <https://doi.org/10.1016/j.jmsy.2011.08.002>
- Kim, H. J., & Lee, J. H. (2023). Look-ahead based reinforcement learning for robotic flow shop scheduling. *Journal of Manufacturing Systems*, 68, 160-175. <https://doi.org/10.1016/J.JMSY.2023.02.002>
- Kirlik, G., & Oguz, C. (2012). A variable neighborhood search for minimizing total weighted tardiness with sequence dependent setup times on a single machine. *Computers & Operations Research*, 39(7), 1506-1520. <https://doi.org/10.1016/j.cor.2011.08.022>
- Kis, T. (2005). A branch-and-cut algorithm for scheduling of projects with variable-intensity activities. *Mathematical programming*, 103, 515-539. <https://doi.org/10.1007/s10107-004-0551-6>
- Kizilay, D., Tasgetiren, M. F., Pan, Q. K., & Gao, L. (2019). A variable block insertion heuristic for solving permutation flow shop scheduling problem with makespan criterion. *Algorithms*, 12(5), 100. <https://doi.org/10.3390/a12050100>
- Kizilay, D., Tasgetiren, M. F., Oztop, H., Kandiller, L., & Suganthan, P. N. (2020, July). A differential evolution algorithm with q-learning for solving engineering design problems. In *2020 IEEE Congress on Evolutionary Computation (CEC)* (pp. 1-8). IEEE. <https://doi.org/10.1109/CEC48606.2020.9185743>
- Labidi, M., Kooli, A., Ladhari, T., Gharbi, A., & Suryahatmaja, U. S. (2018). A computational study of the two-machine no-wait flow shop scheduling problem subject to unequal release dates and non-availability constraints. *IEEE Access*, 6, 16294-16304. DOI: [10.1109/ACCESS.2018.2815598](https://doi.org/10.1109/ACCESS.2018.2815598)
- Lee, J. H., & Kim, H. J. (2022). Reinforcement learning for robotic flow shop scheduling with processing time variations. *International Journal of Production Research*, 60(7), 2346-2368. <https://doi.org/10.1080/00207543.2021.1887533>
- Lei, K., Guo, P., Zhao, W., Wang, Y., Qian, L., Meng, X., & Tang, L. (2022). A multi-action deep reinforcement learning framework for flexible Job-shop scheduling problem. *Expert Systems with Applications*, 205, 117796. <https://doi.org/10.1016/j.eswa.2022.117796>
- Lenstra, J. K., Kan, A. R., & Brucker, P. (1977). Complexity of machine scheduling problems. In *Annals of discrete mathematics* (Vol. 1, pp. 343-362). Elsevier. [https://doi.org/10.1016/S0167-5060\(08\)70743-X](https://doi.org/10.1016/S0167-5060(08)70743-X)
- Li, W., Mitchell, V. L., & Nault, B. R. (2014). Inconsistent objectives in operating room scheduling. In *IIE Annual Conference. Proceedings* (p. 727). Institute of

Industrial and Systems Engineers (IISE).

- Li, W., Nault, B. R., & Ye, H. (2019). Trade-off balancing in scheduling for flow shop production and perioperative processes. *European Journal of Operational Research*, 273(3), 817-830. <https://doi.org/10.1016/j.ejor.2018.08.048>
- Li, H., Gao, K., Duan, P. Y., Li, J. Q., & Zhang, L. (2022a). An improved artificial bee colony algorithm with Q-learning for solving permutation flow-shop scheduling problems. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 53(5), 2684-2693. <https://doi.org/10.1109/TSMC.2022.3219380>
- Li, R., Gong, W., & Lu, C. (2022b). A reinforcement learning based RMOEA/D for bi-objective fuzzy flexible job shop scheduling. *Expert Systems with Applications*, 203, 117380. <https://doi.org/10.1016/j.eswa.2022.117380>
- Li, S., Wang, F., He, Q., & Wang, X. (2023). Deep reinforcement learning for multi-objective combinatorial optimization: A case study on multi-objective traveling salesman problem. *Swarm and Evolutionary Computation*, 83, 101398. <https://doi.org/10.1016/j.swevo.2023.101398>
- Lin, S. W., & Ying, K. C. (2016). Optimization of makespan for no-wait flowshop scheduling problems using efficient matheuristics. *Omega*, 64, 115-125. <https://doi.org/10.1016/j.omega.2015.12.002>
- Lin, Z., Gao, K., Wu, N., & Suganthan, P. N. (2023). Scheduling eight-phase urban traffic light problems via ensemble meta-heuristics and Q-learning based local search. *IEEE Transactions on Intelligent Transportation Systems*. <https://doi.org/10.1109/TITS.2023.3296387>
- Liu, Y. G., Zhu, X., & Li, X. P. (2008, July). A new hybrid genetic algorithm for the bi-criteria no-wait flowshop scheduling problem with makespan and total flow time minimization. In *2008 International Conference on Machine Learning and Cybernetics* (Vol. 2, pp. 883-888). IEEE. <https://doi.org/10.1109/ICMLC.2008.4620529>
- Mahdavi Mazdeh, M., Hamidinia, A., & Karamouzian, A. (2011). A mathematical model for weighted tardy jobs scheduling problem with a batched delivery system. *International Journal of Industrial Engineering Computations*, 2(3), 491-498. <https://doi.org/10.5267/j.ijiec.2011.04.003>
- Manne, A. S. (1960). On the job-shop scheduling problem. *Operations research*, 8(2), 219-223. <https://doi.org/10.1287/opre.8.2.219>
- Mavrotas, G. (2009). Effective implementation of the ϵ -constraint method in multi-objective mathematical programming problems. *Applied mathematics and computation*, 213(2), 455-465. <https://doi.org/10.1016/J.AMC.2009.03.037>

- Merchan, A. F., & Maravelias, C. T. (2016). Preprocessing and tightening methods for time-indexed MIP chemical production scheduling models. *Computers & Chemical Engineering*, 84, 516-535. <https://doi.org/10.1016/j.compchemeng.2015.10.003>
- M'Hallah, R., & Bulfin, R. L. (2005). Minimizing the weighted number of tardy jobs on parallel processors. *European Journal of Operational Research*, 160(2), 471-484. <https://doi.org/10.1016/j.ejor.2003.06.027>
- Minella, G., Ruiz, R., & Ciavotta, M. (2008). A review and evaluation of multiobjective algorithms for the flowshop scheduling problem. *INFORMS Journal on Computing*, 20(3), 451-471. <https://doi.org/10.1287/ijoc.1070.0258>
- Mitchell, M. (1998). *An introduction to genetic algorithms*. MIT press.
- Moore, J. M. (1968). An n job, one machine sequencing algorithm for minimizing the number of late jobs. *Management Science*, 15(1), 102-109. <https://doi.org/10.1287/mnsc.15.1.102>
- Naderi, B., & Ruiz, R. (2010). The distributed permutation flowshop scheduling problem. *Computers & Operations Research*, 37(4), 754-768. <https://doi.org/10.1016/j.cor.2009.06.019>
- Naderi, B., & Sadeghi, H. (2012). A multi-objective simulated annealing algorithm for solving the flexible no-wait flowshop scheduling problem with transportation times. *Journal Optimization in Industrial Engineering*, 5, 33-41.
- Naderi, B., Aminnayeri, M., Piri, M., & Ha'iri Yazdi, M. H. (2012). Multi-objective no-wait flowshop scheduling problems: models and algorithms. *International Journal of Production Research*, 50(10), 2592-2608. <https://doi.org/10.1080/00207543.2010.543937>
- Nagano, M. S., de Almeida, F. S., & Miyata, H. H. (2021). An iterated greedy algorithm for the no-wait flowshop scheduling problem to minimize makespan subject to total completion time. *Engineering Optimization*, 53(8), 1431-1449. <https://doi.org/10.1080/0305215X.2020.1797000>
- Nattaf, M., Horváth, M., Kis, T., Artigues, C., & Lopez, P. (2019). Polyhedral results and valid inequalities for the continuous energy-constrained scheduling problem. *Discrete Applied Mathematics*, 258, 188-203. <https://doi.org/10.1016/j.dam.2018.11.008>
- Nawaz, M., Ensco Jr, E. E., & Ham, I. (1983). A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. *Omega*, 11(1), 91-95. [https://doi.org/10.1016/0305-0483\(83\)90088-9](https://doi.org/10.1016/0305-0483(83)90088-9)
- Ou, J., Xing, L., Yao, F., Li, M., Lv, J., He, Y., ... & Zhang, G. (2023). Deep reinforcement learning method for satellite range scheduling problem. *Swarm and*

- Öztop, H., Tasgetiren, M. F., Kandiller, L., & Pan, Q. K. (2020, July). A novel general variable neighborhood search through q-learning for no-idle flowshop scheduling. In *2020 IEEE Congress on Evolutionary Computation (CEC)* (pp. 1-8). IEEE. <https://doi.org/10.1109/CEC48606.2020.9185556>
- Öztop, H., Tasgetiren, M. F., Kandiller, L., & Pan, Q. K. (2022). Metaheuristics with restart and learning mechanisms for the no-idle flowshop scheduling problem with makespan criterion. *Computers & Operations Research*, 138, 105616. <https://doi.org/10.1016/J.COR.2021.105616>
- Pan, Q. K., & Wang, L. (2008a). No-idle permutation flow shop scheduling based on a hybrid discrete particle swarm optimization algorithm. *The International Journal of Advanced Manufacturing Technology*, 39, 796-807. DOI: <https://doi.org/10.1007/s00170-007-1252-0>
- Pan, Q. K., & Wang, L. (2008b). A novel multi-objective particle swarm optimization algorithm for no-wait flow shop scheduling problems. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, 222(4), 519-539. <https://doi.org/10.1243/09544054JEM989>
- Pan, Q. K., Tasgetiren, M. F., & Liang, Y. C. (2008). A discrete particle swarm optimization algorithm for the no-wait flowshop scheduling problem. *Computers & Operations Research*, 35(9), 2807-2839. <https://doi.org/10.1016/j.cor.2006.12.030>
- 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. <https://doi.org/10.1016/j.cor.2008.10.008>
- Pellegrini, P., Pesenti, R., & Rodriguez, J. (2019). Efficient train re-routing and rescheduling: Valid inequalities and reformulation of RECIFE-MILP. *Transportation Research Part B: Methodological*, 120, 33-48. <https://doi.org/10.1016/j.trb.2018.12.008>
- Pinedo, M. L. (2012). *Scheduling: Theory, Algorithms, and Systems, Fifth Edition*. New York: Springer. <https://doi.org/10.1007/978-3-319-26580-3>
- Qian, B., Wang, L., Huang, D. X., & Wang, X. (2009). Multi-objective no-wait flow-shop scheduling with a memetic algorithm based on differential evolution. *Soft Computing*, 13, 847-869. <https://doi.org/10.1007/s00500-008-0350-8>
- Rahimi-Vahed, A. R., Javadi, B., Rabbani, M., & Tavakkoli-Moghaddam, R. (2008). A multi-objective scatter search for a bi-criteria no-wait flow shop scheduling problem. *Engineering Optimization*, 40(4), 331-346.

<https://doi.org/10.1080/03052150701732509>

- Ren, Y., Gao, K., Fu, Y., Sang, H., Li, D., & Luo, Z. (2023). A novel Q-learning based variable neighborhood iterative search algorithm for solving disassembly line scheduling problems. *Swarm and Evolutionary Computation*, 80, 101338. <https://doi.org/10.1016/j.swevo.2023.101338>
- Reeves, C. R. (1995). A genetic algorithm for flowshop sequencing. *Computers & operations research*, 22(1), 5-13. [https://doi.org/10.1016/0305-0548\(93\)E0014-K](https://doi.org/10.1016/0305-0548(93)E0014-K)
- Reza Hejazi, S., & Saghafian, S. (2005). Flowshop-scheduling problems with makespan criterion: a review. *International Journal of Production Research*, 43(14), 2895-2929. <https://doi.org/10.1080/0020754050056417>
- Röck, H. (1984). The three-machine no-wait flow shop is NP-complete. *Journal of the Association for Computing Machinery (JACM)*, 31(2), 336-345.
- Ruiz, R., & Allahverdi, A. (2007). No-wait flowshop with separate setup times to minimize maximum lateness. *The International Journal of Advanced Manufacturing Technology*, 35, 551-565. <https://doi.org/10.1007/s00170-006-0726-9>
- 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. <https://doi.org/10.1016/j.ejor.2005.12.009>
- Saberi-Aliabad, H., Reisi-Nafchi, M., & Moslehi, G. (2020). Energy-efficient scheduling in an unrelated parallel-machine environment under time-of-use electricity tariffs. *Journal of Cleaner Production*, 249, 119393. <https://doi.org/10.1016/j.jclepro.2019.119393>
- Samarghandi, H. (2015). A particle swarm optimisation for the no-wait flow shop problem with due date constraints. *International Journal of Production Research*, 53(9), 2853-2870. : <https://doi.org/10.1080/00207543.2015.1007245>
- Samarghandi, H., & Behroozi, M. (2017). On the exact solution of the no-wait flow shop problem with due date constraints. *Computers & Operations Research*, 81, 141-159. <https://doi.org/10.1016/j.cor.2016.12.013>
- Sapkal, S. U., & Laha, D. (2013). A heuristic for no-wait flow shop scheduling. *The International Journal of Advanced Manufacturing Technology*, 68, 1327-1338. <https://doi.org/10.1007/s00170-013-4924-y>
- Sorouch, H. M. (2013). Scheduling stochastic jobs on a single machine to minimize weighted number of tardy jobs. *Kuwait Journal of Science*, 40(1). 123-147.

- Stecco, G., Cordeau, J. F., & Moretti, E. (2008). A branch-and-cut algorithm for a production scheduling problem with sequence-dependent and time-dependent setup times. *Computers & Operations Research*, 35(8), 2635-2655. <https://doi.org/10.1016/j.cor.2006.12.021>
- Subramanian, A., Battarra, M., & Potts, C. N. (2014). An iterated local search heuristic for the single machine total weighted tardiness scheduling problem with sequence-dependent setup times. *International Journal of Production Research*, 52(9), 2729-2742. <https://doi.org/10.1080/00207543.2014.883472>
- Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.
- Taillard, E. (1993). Benchmarks for basic scheduling problems. *European Journal of Operational Research*, 64(2), 278-285. [https://doi.org/10.1016/0377-2217\(93\)90182-M](https://doi.org/10.1016/0377-2217(93)90182-M)
- Talbi, E. G. (2016). Combining metaheuristics with mathematical programming, constraint programming and machine learning. *Annals of Operations Research*, 240(1), 171-215. <https://doi.org/10.1007/s10479-015-2034-y>
- Tan, K. C., Goh, C. K., Yang, Y. J., & Lee, T. H. (2006). Evolving better population distribution and exploration in evolutionary multi-objective optimization. *European Journal of Operational Research*, 171(2), 463-495. <https://doi.org/10.1016/j.ejor.2004.08.038>
- Tao, Z., & Liu, X. (2019). Study on No-Wait Flexible Flow Shop Scheduling with Multi-objective. In *Intelligent Robotics and Applications: 12th International Conference, ICIRA 2019, Shenyang, China, August 8–11, 2019, Proceedings, Part VI 12* (pp. 457-468). Springer International Publishing. https://doi.org/10.1007/978-3-030-27529-7_39
- Tasgetiren, M. F., Pan, Q. K., Ozturkoglu, Y., & Chen, A. H. (2016a, July). A memetic algorithm with a variable block insertion heuristic for single machine total weighted tardiness problem with sequence dependent setup times. In *2016 IEEE Congress on Evolutionary Computation (CEC)* (pp. 2911-2918). IEEE. <https://doi.org/10.1109/CEC.2016.7744157>
- Tasgetiren, M. F., Pan, Q. K., Kizilay, D., & Gao, K. (2016b). A variable block insertion heuristic for the blocking flowshop scheduling problem with total flowtime criterion. *Algorithms*, 9(4), 71. <https://doi.org/10.3390/a9040071>
- Tasgetiren, M. F., Yüksel, D., Gao, L., Pan, Q. K., & Li, P. (2019). A discrete artificial bee colony algorithm for the energy-efficient no-wait flowshop scheduling problem. *Procedia Manufacturing*, 39, 1223-1231. <https://doi.org/10.1016/j.promfg.2020.01.347>

- Tasgetiren, M., Kizilay, D., & Kandiller, L. (2024). Solving blocking flowshop scheduling problem with makespan criterion using q-learning-based iterated greedy algorithms. *Journal of Project Management*, 9(2), 85-100. <https://doi.org/10.5267/j.jpm.2024.2.002>
- Tavakkoli-Moghaddam, R., Rahimi-Vahed, A. R., & Mirzaei, A. H. (2008). Solving a multi-objective no-wait flow shop scheduling problem with an immune algorithm. *The International Journal of Advanced Manufacturing Technology*, 36, 969-981. <https://doi.org/10.1007/s00170-006-0906-7>
- Utama, D. M. (2021, March). Minimizing number of tardy jobs in flow shop scheduling using a hybrid whale optimization algorithm. In *Journal of Physics: Conference Series* (Vol. 1845, No. 1, p. 012017). IOP Publishing. <https://doi.org/10.1088/1742-6596/1845/1/012017>
- Uzsoy, R., Lee, C. Y., & Martin-Vega, L. A. (1992). Scheduling semiconductor test operations: Minimizing maximum lateness and number of tardy jobs on a single machine. *Naval Research Logistics (NRL)*, 39(3), 369-388. [https://doi.org/10.1002/1520-6750\(199204\)39:3<369::AID-NAV3220390307>3.0.CO;2-F](https://doi.org/10.1002/1520-6750(199204)39:3<369::AID-NAV3220390307>3.0.CO;2-F)
- Vallada, E., Ruiz, R., & Minella, G. (2008). Minimising total tardiness in the m-machine flowshop problem: A review and evaluation of heuristics and metaheuristics. *Computers & Operations Research*, 35(4), 1350-1373. <https://doi.org/10.1016/j.cor.2006.08.016>
- Vallada, E., Ruiz, R., & Framinan, J. M. (2015). New hard benchmark for flowshop scheduling problems minimising makespan. *European Journal of Operational Research*, 240(3), 666-677. <https://doi.org/10.1016/j.ejor.2014.07.033>
- Varmazyar, M., & Salmasi, N. (2012). Sequence-dependent flow shop scheduling problem minimising the number of tardy jobs. *International Journal of Production Research*, 50(20), 5843-5858. <https://doi.org/10.1080/00207543.2011.632385>
- Wang, L., Gao, K., Lin, Z., Huang, W., & Suganthan, P. N. (2023). Problem feature based meta-heuristics with Q-learning for solving urban traffic light scheduling problems. *Applied Soft Computing*, 147, 110714. <https://doi.org/10.1016/j.asoc.2023.110714>
- Watkins, C. J. C. H. (1989). Learning from delayed rewards.
- Weise, T. (2009). Global optimization algorithms-theory and application. *Self-Published Thomas Weise*, 361, 153.
- Wismer, D. A. (1972). Solution of the flowshop-scheduling problem with no intermediate queues. *Operations research*, 20(3), 689-697.

<https://doi.org/10.1287/opre.20.3.689>

- Wu, X., & Che, A. (2020). Energy-efficient no-wait permutation flow shop scheduling by adaptive multi-objective variable neighborhood search. *Omega*, 94, 102117. <https://doi.org/10.1016/J.OMEGA.2019.102117>
- Xu, H., Lü, Z., & Cheng, T. C. E. (2014). Iterated local search for single-machine scheduling with sequence-dependent setup times to minimize total weighted tardiness. *Journal of Scheduling*, 17, 271-287. <https://doi.org/10.1007/s10951-013-0351-z>
- Yamada, T., Nagano, M., & Miyata, H. (2021). Minimization of total tardiness in no-wait flowshop production systems with preventive maintenance. *International Journal of Industrial Engineering Computations*, 12(4), 415-426. <https://doi.org/10.5267/j.ijiec.2021.5.002>
- Ye, H., Li, W., & Nault, B. R. (2020). Trade-off balancing between maximum and total completion times for no-wait flow shop production. *International Journal of Production Research*, 58(11), 3235-3251. <https://doi.org/10.1080/00207543.2019.1630777>
- Ying, K. C., & Lin, S. W. (2018). Minimizing makespan for no-wait flowshop scheduling problems with setup times. *Computers & Industrial Engineering*, 121, 73-81. <https://doi.org/10.1016/j.cie.2018.05.030>
- Ying, K. C., Lu, C. C., & Lin, S. W. (2018). Improved exact methods for solving no-wait flowshop scheduling problems with due date constraints. *IEEE Access*, 6, 30702-30713. <https://doi.org/10.1109/ACCESS.2018.2834954>
- Yu, H., Gao, K. Z., Ma, Z. F., & Pan, Y. X. (2023). Improved meta-heuristics with Q-learning for solving distributed assembly permutation flowshop scheduling problems. *Swarm and Evolutionary Computation*, 80, 101335. <https://doi.org/10.1016/j.swevo.2023.101335>
- Yüksel, D. (2019). Bi-objective no-wait permutation flowshop scheduling problems. Master's Thesis, Fen Bilimleri Enstitüsü, Yaşar University, 2019.
- Yüksel, D., Taşgetiren, M. F., Kandiller, L., & Pan, Q. K. (2020a, July). Metaheuristics for energy-efficient no-wait flowshops: A trade-off between makespan and total energy consumption. In *2020 IEEE Congress on Evolutionary Computation (CEC)* (pp. 1-8). IEEE. DOI: [10.1109/CEC48606.2020.9185554](https://doi.org/10.1109/CEC48606.2020.9185554)
- Yüksel, D., Taşgetiren, M. F., Kandiller, L., & Gao, L. (2020b). An energy-efficient bi-objective no-wait permutation flowshop scheduling problem to minimize total tardiness and total energy consumption. *Computers & Industrial Engineering*, 145, 106431. <https://doi.org/10.1016/j.cie.2020.106431>

- Yüksel, D., Kandiller, L., & Taşgetiren, M. F. (2022a, October). Mathematical Models for No-Wait Permutation Flowshop Scheduling Problems. In *The International Symposium for Production Research* (pp. 554-565). Cham: Springer International Publishing. https://link.springer.com/chapter/10.1007/978-3-031-24457-5_44
- Yüksel, D., Kandiller, L., & Taşgetiren, M. F. (2022b, July). Intelligent valid inequalities for no-wait permutation flowshop scheduling problems. In *International Conference on Intelligent and Fuzzy Systems* (pp. 914-922). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-031-09173-5_105
- Yüksel, D., Kandiller, L., Taşgetiren, M.F. (2023). Bi-Criteria Optimization of Makespan and Total Flow Time in No-Wait Flowshops, In *The 15th International Conference on Multiple Objective Programming MOPGP 2023*, Book of Abstracts (pp. 34). <https://mopgp.org/wp-content/uploads/2023/09/Mopgp-Book-2023.pdf>
- Yüksel, D., Kandiller, L., & Taşgetiren, M. F. (2024). Q-learning Guided Algorithms for Bi-Criteria Minimization of Total Flow Time and Makespan in No-Wait Permutation Flowshops. *Swarm and Evolutionary Computation*, 101617. <https://doi.org/10.1016/j.swevo.2024.101617>
- Zhang, Y., Zhu, H., Tang, D., Zhou, T., & Gui, Y. (2022). Dynamic job shop scheduling based on deep reinforcement learning for multi-agent manufacturing systems. *Robotics and Computer-Integrated Manufacturing*, 78, 102412. <https://doi.org/10.1016/j.rcim.2022.102412>
- Zhao, F., Zhang, L., Cao, J., & Tang, J. (2021). A cooperative water wave optimization algorithm with reinforcement learning for the distributed assembly no-idle flowshop scheduling problem. *Computers & Industrial Engineering*, 153, 107082. <https://doi.org/10.1016/j.cie.2020.107082>
- Zhao, F., Jiang, T., & Wang, L. (2022a). A reinforcement learning driven cooperative meta-heuristic algorithm for energy-efficient distributed no-wait flow-shop scheduling with sequence-dependent setup time. *IEEE Transactions on Industrial Informatics*, 19(7), 8427-8440. <https://doi.org/10.1109/TII.2022.3218645>
- Zhao, F., Di, S., & Wang, L. (2022b). A hyperheuristic with Q-learning for the multiobjective energy-efficient distributed blocking flow shop scheduling problem. *IEEE Transactions on Cybernetics*, 53(5), 3337-3350. <https://doi.org/10.1109/TCYB.2022.3192112>
- Zhao, F., Wang, Z., & Wang, L. (2022c). A reinforcement learning driven artificial bee colony algorithm for distributed heterogeneous no-wait flowshop scheduling problem with sequence-dependent setup times. *IEEE Transactions on Automation Science and Engineering*, 20(4), 2305-2320.

<https://doi.org/10.1109/TASE.2022.3212786>

Zhao, F., Hu, X., Wang, L., Xu, T., Zhu, N., & Jonrinaldi. (2023). A reinforcement learning-driven brain storm optimisation algorithm for multi-objective energy-efficient distributed assembly no-wait flow shop scheduling problem. *International Journal of Production Research*, 61(9), 2854-2872. <https://doi.org/10.1080/00207543.2022.2070786>



APPENDICES

APPENDIX A. Generated Due Dates for VRF Instances for Lateness-Related Performance Measures of NWPFSs

APPENDIX A1. Generated Due Dates for VRF Instances with 10 Jobs

Instances	Due Date of Jobs	Instances	Due Date of Jobs
10-5-01	[384,142,85,301,159,188,411,429,346,393]	10-15-01	[748,684,469,722,304,527,214,375,843,609]
10-5-02	[152,321,294,49,244,234,432,334,454,65]	10-15-02	[820,436,743,705,934,942,393,895,490,587]
10-5-03	[420,276,372,124,57,85,129,458,457,220]	10-15-03	[885,638,747,449,416,910,950,939,632,791]
10-5-04	[126,254,393,423,465,340,296,455,204,227]	10-15-04	[922,652,799,904,834,495,547,567,633,784]
10-5-05	[301,141,379,459,413,231,385,95,206,396]	10-15-05	[757,608,683,342,891,449,638,773,240,541]
10-5-06	[328,403,218,54,434,263,504,462,121,376]	10-15-06	[812,515,873,686,754,256,897,637,485,539]
10-5-07	[447,367,267,394,142,336,87,193,460,341]	10-15-07	[596,452,604,937,892,957,415,540,303,699]
10-5-08	[281,102,37,160,394,445,92,265,449,313]	10-15-08	[394,576,886,987,740,467,811,336,602,842]
10-5-09	[407,194,497,322,398,91,519,299,125,240]	10-15-09	[989,680,425,758,602,644,513,265,336,990]
10-5-10	[283,85,131,187,317,202,265,157,406,410]	10-15-10	[572,859,373,781,708,907,227,635,669,931]
10-10-01	[722,626,228,753,142,546,301,691,324,535]	10-20-01	[966,682,750,1065,499,457,889,920,1066,843]
10-10-02	[613,497,512,273,266,646,702,658,458,573]	10-20-02	[903,583,749,408,365,978,1071,1013,496,771]
10-10-03	[718,665,242,701,596,188,394,174,553,423]	10-20-03	[1055,990,1153,1102,494,842,899,678,528,427]
10-10-04	[507,293,718,436,193,446,723,164,524,582]	10-20-04	[877,777,541,1092,627,521,1088,672,963,312]
10-10-05	[612,430,403,591,222,731,692,259,170,484]	10-20-05	[932,951,804,437,608,977,847,1044,519,666]
10-10-06	[363,238,151,552,624,530,307,680,620,262]	10-20-06	[546,778,680,802,370,1224,947,1290,1246,1067]
10-10-07	[264,722,471,580,606,661,680,525,405,397]	10-20-07	[1067,717,538,1097,474,801,807,948,1073,622]
10-10-08	[351,309,757,751,660,564,408,705,280,634]	10-20-08	[441,883,925,1071,509,1055,804,717,892,733]
10-10-09	[151,629,487,457,679,235,570,335,551,608]	10-20-09	[943,799,465,713,1075,507,647,955,410,1149]
10-10-10	[626,719,537,366,424,494,229,695,675,451]	10-20-10	[692,802,269,765,571,1056,536,953,982,462]

APPENDIX A2. Generated Due Dates for VRF Instances with 20 Jobs

Instances	Due Date of Jobs
20-5-01	[189,829,533,883,495,321,70,761,877,405,930,575,554,450,91,619,671,310,223,591]
20-5-02	[382,507,261,940,489,121,874,685,223,424,948,909,752,626,167,817,304,583,355,707]
20-5-03	[596,799,719,136,520,172,226,741,665,436,181,567,840,888,258,418,978,950,486,365]
20-5-04	[636,704,705,453,97,217,585,260,828,372,617,915,362,900,132,269,776,341,492,869]
20-5-05	[447,608,504,853,480,592,975,736,313,213,890,390,157,149,713,993,665,121,808,335]
20-5-06	[575,476,379,888,290,857,160,78,690,868,786,107,316,420,597,201,500,870,254,675]
20-5-07	[427,130,172,941,267,917,622,866,50,806,940,674,529,263,594,720,462,388,318,647]
20-5-08	[396,840,884,372,455,913,499,534,879,663,811,272,686,120,218,747,311,107,226,550]
20-5-09	[473,673,699,122,921,818,131,586,741,291,778,859,254,893,654,520,318,542,401,930]
20-5-10	[775,885,411,535,953,1002,168,363,141,626,273,229,640,752,322,915,427,433,687,790]
20-10-01	[1262,562,1166,358,1337,119,766,842,629,956,1068,1106,680,488,225,1287,451,520,935,704]
20-10-02	[234,353,344,672,540,577,1265,764,1233,1156,301,853,856,1025,588,660,713,1065,1121,1311]
20-10-03	[1246,766,568,329,388,1273,869,1071,1357,371,595,1103,1327,757,951,230,665,932,1140,102]
20-10-04	[1237,1124,615,466,1043,931,932,535,290,366,788,1218,808,1187,1057,281,1290,661,434,503]
20-10-05	[757,969,924,692,859,1164,361,512,1352,328,1402,727,1204,1389,196,1022,995,625,498,247]
20-10-06	[843,707,573,1286,488,1157,357,215,1046,1199,1084,233,571,664,881,426,791,1239,483,927]
20-10-07	[437,968,348,585,891,995,220,270,660,1244,1060,1209,135,741,842,463,495,1272,1352,693]
20-10-08	[1024,516,240,1285,891,323,1117,749,954,972,599,538,426,621,1238,1189,1063,364,695,397]
20-10-09	[566,870,673,1155,666,739,1268,1047,395,373,1201,558,301,218,985,1281,896,151,1134,436]
20-10-10	[435,1287,812,1014,1085,928,665,238,712,1123,1183,899,582,373,641,158,502,763,862,299]
20-15-01	[649,1743,1233,1422,1489,1365,865,346,940,1607,1721,1353,774,565,841,297,708,986,1290,425]
20-15-02	[1588,417,638,1362,1183,855,536,1631,1668,392,1228,929,1276,361,1028,884,1500,775,1690,1473]
20-15-03	[1565,606,1376,1387,300,1535,1276,688,969,1075,383,751,505,1173,883,640,968,523,1614,872]
20-15-04	[841,1061,453,864,1576,1069,648,946,1353,1178,610,326,1480,294,1137,1527,797,1298,691,1267]
20-15-05	[1558,1687,869,1508,702,1604,892,1478,971,927,468,387,1404,606,734,1174,1291,1351,1096,550]
20-15-06	[1335,329,894,577,450,538,989,1252,798,633,1410,1208,1106,1663,913,1614,410,643,1384,1694]
20-15-07	[1384,1281,1491,1046,587,1058,688,346,1489,1378,697,398,1615,948,811,502,1254,548,869,1671]
20-15-08	[627,931,1575,556,673,974,1162,237,1520,578,1448,878,892,500,1579,1373,377,1243,1264,1012]
20-15-09	[710,1077,1128,290,1566,1340,382,870,1255,434,1313,1404,402,1489,935,792,586,837,606,1690]
20-15-10	[1208,1237,1349,997,258,528,1140,578,1389,860,1181,1605,810,1554,487,627,1352,698,1032,1483]
20-20-01	[995,1843,1693,551,908,1978,759,1768,1821,1528,1451,1351,1308,776,2048,1390,1706,1086,583,451]
20-20-02	[1668,1787,463,861,934,499,1060,1122,1480,1991,1158,1394,1893,687,772,1904,655,1719,1256,1525]
20-20-03	[1897,351,1370,1039,1807,558,1544,687,787,485,1234,1094,852,1287,1507,1310,1753,1835,953,1638]
20-20-04	[1628,961,650,1940,1562,809,1047,1221,1262,574,1276,821,1179,1782,451,303,1487,1839,877,1739]
20-20-05	[959,1393,1143,1950,1685,2008,630,1999,413,1696,1737,2085,674,1024,1177,578,1367,837,1791,1649]
20-20-06	[530,1399,1354,863,939,1812,894,437,1965,1576,1280,682,1081,1522,1062,478,1878,708,1707,1999]
20-20-07	[1968,1367,1520,1131,656,915,1572,1288,2017,827,1666,591,688,609,1190,1233,2120,2047,445,1311]
20-20-08	[1723,1854,1445,791,1025,906,683,1806,1331,579,1670,353,1116,1399,1153,755,652,895,1266,1463]
20-20-09	[910,2062,1392,1722,1806,1638,1134,624,1303,1939,2031,1590,999,695,1093,376,937,1344,1502,657]
20-20-10	[1516,879,440,1428,584,1626,1005,1148,664,1588,773,1780,1882,1812,1087,1711,1216,1889,1218,512]

APPENDIX A3. Generated Due Dates for VRF Instances with 30 Jobs

Instances	Due Date of Jobs
30-5-01	[929,224,996,1389,157,594,785,998,891,1231,261,1144,344,144,688,1316,868,781,364,673,1192,537,1093,491,444,614,1337,400,1270,314]
30-5-02	[1117,332,337,465,573,1223,401,1060,1270,568,672,974,1256,790,122,913,70,831,267,220,653,1257,219,1143,753,513,476,156,957,365]
30-5-03	[1322,1095,855,1368,1195,220,593,380,1020,328,462,1394,890,264,1073,1181,65,811,764,509,937,1263,671,727,567,174,974,167,1252,331]
30-5-04	[1039,939,354,212,868,134,287,378,628,744,1340,227,696,1095,1130,402,151,1316,493,436,777,1169,570,748,1208,1275,1038,536,53,880]
30-5-05	[1190,590,246,1110,196,1129,917,820,842,717,705,321,623,139,385,517,875,366,1198,964,789,111,1015,582,472,290,202,1272,1076,436]
30-5-06	[1319,842,123,1273,1006,592,964,1173,696,1149,289,447,1400,244,723,1373,669,537,1052,1101,778,1322,568,755,418,1214,373,430,201,853]
30-5-07	[444,787,828,167,1421,495,1400,360,889,356,986,219,267,518,736,265,987,1195,142,1097,1106,1025,1230,1358,545,78,533,813,1261,690]
30-5-08	[1158,931,1281,396,782,846,568,667,1238,1111,449,890,1318,528,764,627,126,1106,199,1394,153,1403,905,233,641,454,576,249,1005,1055]
30-5-09	[728,1132,427,280,1311,1364,384,755,775,612,844,139,306,994,551,1166,1041,914,65,956,368,1116,475,959,1265,687,800,1356,232,518]
30-5-10	[1254,452,735,1126,1059,635,789,1302,1354,548,440,481,153,411,806,205,910,593,1363,665,368,987,257,1356,879,1227,655,1142,1020,337]
30-10-01	[1753,734,1663,351,1879,977,873,315,605,1467,447,1620,1314,152,1589,1798,1091,1326,704,559,192,1903,721,1057,1149,1237,1017,1395,1432,517]
30-10-02	[1883,1059,481,1758,307,1857,1508,1366,1398,1223,1179,553,1106,217,693,872,1420,688,1950,1549,1318,211,1562,934,817,521,428,1971,1714,778]
30-10-03	[1576,1745,975,601,1678,1807,226,343,306,1943,1041,553,334,675,1866,1094,1449,852,1363,488,455,1243,528,911,1648,1837,806,1169,1439,1618]
30-10-04	[828,322,1686,1487,975,1231,1737,537,454,1007,1876,1006,1593,1192,1327,235,1936,647,1945,929,1666,695,219,1457,1429,434,263,1070,1810,585]
30-10-05	[910,1014,723,415,559,1658,405,1454,1311,654,1755,792,1344,1404,1808,1581,1210,435,1199,1884,307,842,235,1490,1890,1595,964,598,1132,1037]
30-10-06	[379,1893,1667,1346,1712,242,565,1283,825,629,1547,1890,363,1411,986,787,1590,909,759,1781,1310,1843,515,1499,1051,409,1033,258,469,1609]
30-10-07	[1747,939,465,1571,351,1666,1318,1176,1242,1058,1025,530,977,306,567,808,1299,564,1752,1430,1126,160,1492,838,780,476,396,1792,1533,722]
30-10-08	[707,1720,306,1804,1035,612,146,497,929,1145,1269,1584,355,801,1675,544,743,1333,1012,730,824,247,1318,1176,1701,1519,442,395,1217,1471]
30-10-09	[1025,1122,1515,254,870,1741,923,465,1461,771,428,1178,710,315,471,1838,978,1414,171,557,791,1159,1836,1034,1542,647,1254,1269,1670,1644]
30-10-10	[1570,1327,1663,522,1079,1133,696,961,1616,1541,546,1214,1710,615,981,848,241,1391,345,1780,279,1827,1249,372,911,597,778,447,1355,1364]

APPENDIX A3. (Cont'd.) Generated Due Dates for VRF Instances with 30 Jobs

Instances	Due Date of Jobs
30-15-01	[1033,1641,1796,1161,1485,2012,1347,470,2237,698,1701,421,893,1990,1563,363,2180,1897,1436,2301,1526,549,1080,1120,2292,743,942,647,2215,300]
30-15-02	[1987,981,464,2498,1217,1289,1423,1598,856,1021,1727,656,695,2127,1171,1840,583,2421,818,1528,2508,392,1612,295,565,1923,1379,1457,2088,1957]
30-15-03	[533,1024,1301,2437,872,2111,1086,584,2035,2132,1742,1198,1651,1867,1381,838,1907,2295,676,1980,2397,1499,1080,1188,1936,445,1535,1780,1593,956]
30-15-04	[912,1887,530,1737,306,2207,720,2323,2226,799,715,376,1575,2424,1699,1467,1790,939,585,496,1021,1211,1350,2094,1272,1892,2373,1388,1109,1622]
30-15-05	[2047,776,503,856,196,1476,1644,543,1307,465,1449,2305,2442,794,1906,386,1075,2023,2106,1290,2363,1497,917,2214,1913,1856,1177,592,1069,1604]
30-15-06	[1303,1697,358,1500,1962,1847,853,545,1517,2333,1233,2203,798,1914,1063,1072,870,2390,2470,2290,671,2086,729,505,2421,1432,1518,2013,477,1346]
30-15-07	[2072,2315,1323,2021,615,2266,874,1129,1455,346,408,1304,1973,611,1392,1156,1441,772,815,1813,2130,448,2180,519,1869,2307,1755,1667,1678,959]
30-15-08	[2134,1198,623,2025,492,2060,1666,1495,1609,1428,1371,744,1335,380,849,1020,1628,778,2225,1827,1444,323,1881,1072,1001,710,579,2303,1908,981]
30-15-09	[449,1941,2348,2430,1430,1867,1331,594,788,710,1994,624,1612,1708,249,2146,2290,1072,2416,1088,528,913,509,1527,1230,1700,1558,2010,777,1477]
30-15-10	[1645,1757,715,1134,2343,2036,1581,1714,1960,403,995,2075,668,2288,1226,1414,1043,1595,1891,2354,1266,1478,2263,954,900,557,1293,281,330,813]
30-20-01	[1880,2153,950,1268,1521,1778,2075,1125,1057,2696,1728,2744,1799,900,471,776,1299,2433,1259,594,2458,305,2788,2602,577,691,1386,2263,2652,2293]
30-20-02	[2527,835,607,932,396,1760,2041,682,1465,580,1710,2816,2873,896,2157,465,1241,2314,2678,1385,2857,1789,1010,2731,2276,2085,1280,777,1094,1963]
30-20-03	[2830,2041,742,1872,2588,2204,1735,1359,946,1262,2839,1819,1217,2726,966,2195,1145,828,622,2531,2415,393,1086,1574,1946,2320,902,2449,1524,566]
30-20-04	[2649,1926,773,1861,2506,2151,1702,1522,1060,1467,2674,1743,1351,2583,1159,2077,1283,844,699,2453,2291,538,1247,1666,1884,2188,1039,2396,1604,567]
30-20-05	[2694,1502,733,2600,1803,1026,823,1329,2525,2456,1694,364,2622,2242,1841,1649,1764,2177,1063,1281,2733,2050,1856,1993,2372,929,797,2359,1338,411];
30-20-06	[518,1129,2216,1048,1632,2553,741,1448,2113,792,2727,1503,370,849,2409,1842,1748,2478,2290,2693,497,865,2093,1547,2673,1971,1862,1382,1172,1765]
30-20-07	[1526,668,2436,2475,2695,349,1695,1018,2082,2826,2227,1319,1893,1760,1968,2807,2625,2258,1365,978,1604,740,574,467,2770,2541,1469,1135,639,1833]
30-20-08	[1858,2052,1114,1412,1644,1748,1916,1235,1180,2516,1674,2789,1826,1026,604,985,1492,2307,1379,678,2411,580,2845,2448,658,810,1583,2123,2472,2260]
30-20-09	[2364,1301,1050,2591,1622,628,1712,1442,2437,1854,2762,2038,1501,909,880,344,1153,1307,817,2173,1811,2249,696,1689,2738,2679,714,1409,2510,2127]
30-20-10	[1732,2857,1156,2850,2352,2299,1444,1299,2663,2654,996,1505,756,1687,1433,1111,649,2068,2458,1826,2239,777,1353,460,603,2738,2502,818,2083,2211]

APPENDIX A4. Generated Due Dates for VRF Instances with 40 Jobs

Instances	Due Date of Jobs
40-5-01	[1046,794,1282,389,707,1103,1490,721,187,1831,380,1252,1686,1395,466,848,941,1315,36,1896,594,1205,207,88,745,1556,1593,1666,541,976,490,1768,314,290,1447,1353,661,905,1896,1516]
40-5-02	[107,581,727,832,526,297,1384,635,705,441,1510,1127,952,586,1449,189,1881,1192,1962,480,341,978,855,1918,1094,1259,1543,1286,1702,1638,1030,253,160,382,1818,1156,1582,755,1753,1377]
40-5-03	[1299,973,1581,444,598,549,706,472,822,1482,1799,1406,782,1539,431,329,1798,173,1836,1270,1380,1207,1040,207,1041,1703,1680,613,852,1049,83,744,284,1320,1761,1140,398,354,1520,957]
40-5-04	[673,1148,720,518,570,421,1780,1786,1705,1019,379,1199,770,1401,124,474,1628,1351,1299,779,1117,210,911,1555,292,979,1219,870,609,633,192,1447,854,1300,1535,1075,1657,1509,313,1753]
40-5-05	[1733,858,1165,536,1539,702,917,302,131,583,768,1144,1226,657,1672,1798,986,1348,360,168,499,1878,1548,1587,1241,1511,1439,758,636,405,717,234,1067,1369,1709,1031,1843,1365,268,399]
40-5-06	[513,1235,1728,1767,965,1029,71,816,786,710,1247,1137,363,1569,1363,297,1337,1188,323,1031,572,265,1487,596,81,434,938,1604,155,1511,1266,1418,206,713,943,652,438,1096,1413,1654]
40-5-07	[1705,734,855,308,1600,289,193,54,1271,622,1626,493,795,1378,1766,826,1135,650,959,425,269,1484,541,454,142,1824,1449,359,1325,692,1077,1022,989,1658,996,758,1501,1140,1202,1364]
40-5-08	[1932,1730,806,1180,725,1281,482,1983,1419,341,135,847,1358,875,704,1706,1262,998,1550,902,1309,577,222,91,212,185,1864,1098,1422,536,669,983,385,2003,1556,1146,615,464,1658,312]
40-5-09	[1537,734,1617,139,244,1900,207,479,1573,366,305,1776,776,626,713,1284,1697,264,1829,1891,825,838,1432,1444,975,1205,604,1727,1110,1735,539,1074,1355,986,1257,880,911,1134,180,432]
40-5-10	[2027,1734,959,1408,606,772,1232,441,188,1118,731,992,1934,1559,322,554,1783,981,1333,1060,1157,1636,720,1718,1435,1719,1602,379,813,102,1368,472,166,745,1841,1941,853,1478,1195,272]
40-10-01	[621,2422,1833,1091,1913,2388,1148,552,1297,940,2049,842,208,1229,1506,1978,2488,1007,2653,2049,664,717,1896,2316,1247,2244,362,252,1738,1682,320,747,511,1547,2625,2695,2259,2461,1054,880]
40-10-02	[593,1460,1261,2597,1723,2488,847,2377,1914,2344,2499,684,421,2037,1687,1154,722,849,232,812,466,1841,2247,1918,264,2362,592,2303,641,1781,1606,1120,1750,344,1254,2116,1380,1192,1453,2003]
40-10-03	[2267,238,816,2203,2111,1462,966,729,2171,1257,616,534,501,1391,1837,1735,1999,287,2394,1240,687,1946,1759,2347,1185,1032,355,898,416,1343,1281,750,1112,1625,642,455,1899,2230,1502,1662]
40-10-04	[1597,813,791,2306,2236,642,1049,346,1447,1636,2156,2065,1262,1787,498,885,1925,517,2157,2029,578,676,738,1085,1587,1150,366,2597,1741,2512,1003,2473,2124,1461,1847,1159,271,1358,1411,204]
40-10-05	[2282,535,586,313,960,1757,2049,1270,1515,1378,1482,2519,1861,652,2259,614,414,2317,2420,1810,993,1566,783,2475,1212,1400,477,2178,2015,2039,1615,248,2378,1782,1460,1142,994,2099,889,849]
40-10-06	[2063,1215,1009,438,593,1439,1125,788,1973,2096,1073,2448,1512,1876,1379,1737,2251,1671,2439,960,823,284,328,1282,520,879,516,1900,638,1593,2154,2332,352,2318,1748,1811,2201,369,1250,1164]
40-10-07	[1160,1965,2258,2148,858,390,1488,2082,483,284,766,1074,2383,1787,1274,653,1318,2186,1704,424,1583,312,1283,2309,1511,1148,1648,582,1229,773,818,455,1004,1762,1839,855,1996,2517,1441,2463]
40-10-08	[348,1907,2307,1727,2371,612,534,1623,1293,592,1956,2504,842,1855,1019,1359,2029,435,1603,2460,1840,1408,356,2237,1319,2177,1149,1512,2068,722,993,1627,1069,2540,686,295,899,788,1234,2257]
40-10-09	[256,2069,2415,1793,2496,522,490,1612,1195,512,2110,2565,813,1960,986,1391,2150,455,1584,2505,1815,1413,347,2349,1282,2278,1103,1520,2201,742,924,1664,1076,2567,664,229,917,777,1191,2360]
40-10-10	[896,1593,954,710,795,643,2470,2530,2413,1430,466,1655,1002,2010,196,660,2257,1911,1825,1102,1560,341,1314,2227,432,1368,1708,1203,812,869,234,2039,1197,1830,2146,1480,2333,2080,457,2460]

APPENDIX A4. (Cont'd.) Generated Due Dates for VRF Instances with 40 Jobs

Instances	Due Date of Jobs
40-15-01	[344,2342,2885,2171,2915,809,691,2006,1620,753,2448,3030,1125,2329,1237,1765,2492,629,1969,2972,2201,1811,558,2698,1665,2640,1467,1862,2580,979,1227,2142,1256,3136,855,299,1149,1025,1571,2770]
40-15-02	[800,2660,2069,1378,2138,2632,1532,727,1714,1172,2380,1076,319,1559,1816,2269,2953,1258,3074,2350,894,943,2076,2545,1674,2501,518,384,1906,1881,449,987,613,1842,2954,3085,2533,2815,1336,1104]
40-15-03	[2511,1680,386,1750,2406,1361,1501,1801,1294,589,1817,1567,1239,1079,2069,2094,902,975,1135,2667,3007,2230,2446,3160,2870,338,1428,2487,2298,2341,816,3076,2675,864,447,1896,639,2978,2812,768]
40-15-04	[1757,1873,1543,1167,598,2668,928,2444,2107,1067,2806,896,2149,2262,254,2649,1993,371,1235,2900,733,2354,1672,978,2957,1448,2554,1962,1573,2394,2783,583,1723,769,1280,780,1914,1398,492,471]
40-15-05	[799,2990,2796,2114,2861,1041,1755,2821,1230,2475,1445,1063,394,481,1414,1985,1211,897,1647,1578,853,2434,688,2752,503,3038,3158,3111,458,2148,2435,2356,315,2568,1941,1846,796,1699,1111,2247]
40-15-06	[2885,1367,1710,649,2756,608,542,361,2307,1134,2806,951,1451,2463,3010,1558,2137,1167,1754,873,554,2639,1121,925,421,3055,2574,738,2378,1243,2018,1950,1845,2840,1915,1434,2714,2186,2215,2434]
40-15-07	[2499,1609,484,1702,2280,1394,1551,1756,1389,594,1861,1563,1299,1135,1966,2008,1023,1036,1214,2551,2950,2069,2429,3059,2800,338,1478,2441,2145,2251,831,2999,2693,927,515,1901,623,2857,2767,647]
40-15-08	[1785,2019,2303,2417,664,1908,1099,3053,1478,1229,2660,1985,693,1421,3098,585,506,1594,2834,411,2982,2173,2748,2572,1711,1221,598,959,3108,2074,2308,1433,2643,1793,1458,2551,2353,923,3127,1048]
40-15-09	[778,1732,1579,3097,2024,2980,1178,2921,2290,2825,3037,985,623,2546,1989,1355,1071,1259,377,1137,639,2244,2649,2412,467,2828,766,2745,780,2148,1957,1288,2109,530,1541,2580,1581,1431,1676,2439]
40-15-10	[2587,1653,442,1808,2435,1420,1556,1836,1346,569,1938,1600,1192,973,2060,2099,895,962,1109,2662,3039,2218,2463,3090,2909,329,1484,2559,2316,2357,768,3076,2800,804,506,2010,577,2982,2876,661]
40-20-01	[1708,2984,2551,2405,3369,1733,3621,3095,1106,2752,2592,961,1242,1891,2953,3654,623,2785,1628,572,3208,1158,3528,2230,2639,1309,849,1027,2097,2063,2027,3411,868,1544,228,741,1459,3488,2331,1346]
40-20-02	[792,2341,1496,2901,2937,3523,2809,2177,2067,3581,2718,697,1767,2250,1237,494,3472,436,1154,1271,3020,3459,896,907,1894,2530,2449,2853,3178,1874,3362,1329,3728,3216,535,2127,1601,3757,2571,1579]
40-20-03	[670,2807,930,2261,2660,3640,2911,1748,2491,2148,1174,1381,3671,1608,733,2623,817,1237,3513,1022,1881,542,1683,798,2230,445,3134,2883,1423,1579,2108,2798,3375,1248,2540,1978,3156,1833,3628,3448]
40-20-04	[537,3206,344,3305,2891,841,868,3142,3346,2363,3531,2464,1234,715,2841,1766,3504,1000,3656,756,2561,2875,3697,2667,1409,1914,1369,1559,2205,2327,1938,1806,2707,3042,2137,1503,1035,1011,1957,479]
40-20-05	[2788,2708,405,529,1654,2943,2434,2592,3152,3106,1812,3022,3410,2547,1697,2912,2775,1311,2283,1048,2225,662,858,2016,2056,2366,1095,776,1617,821,1345,525,1272,1573,2199,1850,1395,3466,2667,892]
40-20-06	[553,2744,3367,1722,1410,796,1765,1291,2524,1354,2221,1906,946,745,3269,1055,613,3135,3654,1587,2672,683,2355,1976,2857,2159,2789,1190,3323,3617,2273,2967,535,2137,3240,1400,3439,1091,3533,2729]
40-20-07	[1506,2374,1722,1127,1282,874,3911,3975,3531,2184,868,2476,1728,2906,359,912,3355,2827,2603,1798,2330,606,1995,3347,622,2104,2566,1910,1379,1440,510,3036,1898,2809,3251,2193,3459,3198,712,3823]
40-20-08	[1089,2952,1948,3331,1232,414,1259,869,509,2036,3406,2594,1747,2364,1132,2447,1491,2842,3049,1535,3255,588,781,3468,3087,1872,2181,1052,1837,2448,1676,814,2780,377,2208,935,1963,1209,2772,2852]
40-20-09	[539,2251,3194,2426,1333,2944,3077,2204,1883,3065,1437,979,1923,2350,1201,2056,713,2694,2531,529,3479,1721,1520,2115,1386,762,3456,1012,2836,1139,2800,1088,895,3343,1067,3015,1860,3282,2775,1638]
40-20-10	[3248,1245,1016,1201,1977,695,3080,2730,3352,1179,2103,2433,1662,2321,844,1369,1911,3384,1767,3439,2048,389,1349,2691,592,2358,3200,1828,2278,1118,966,2850,2994,449,2668,1061,1848,767,3047,2470]

APPENDIX A5. Generated Due Dates for VRF Instances with 50 Jobs

Instances	Due Date of Jobs
50-5-01	[1684,1997,1610,1449,1477,1336,2496,2534,2406,1800,950,1030,517,259,1570,1126,2289,1557,769,1075,1381,390,1146,2271,753,610,2001,697,1747,889,327,387,209,732,1314,1114,870,596,1834,321,124,2069,163,1950,2139,1846,2371,2074,1204,2449]
50-5-02	[1462,2191,1177,133,2148,2191,737,688,1224,1081,1623,699,1302,1519,1590,614,1874,979,1814,1236,309,435,818,332,943,168,865,567,1395,1541,1830,2070,1940,828,153,382,270,1125,1353,536,519,2276,1766,1017,1681,2056,2269,1421,1710,1984]
50-5-03	[1025,1599,230,1250,1762,583,1241,1975,402,749,1556,1179,2035,2108,500,1020,2219,2129,1911,445,1504,936,198,1100,1850,2229,311,2185,1817,1370,694,1264,1786,2007,1717,624,171,89,362,811,1378,1118,566,803,1636,144,1446,1436,277,813]
50-5-04	[472,1935,205,2207,2022,286,1506,172,1847,243,1239,2322,1745,348,1853,1985,628,1533,580,480,540,2109,1174,1798,1650,851,328,1104,1296,1197,2288,2113,2160,1328,1684,2052,792,415,1752,1395,721,1063,1438,1565,786,988,937,109,968,825]
50-5-05	[1664,1890,1589,1377,1413,1266,2356,2400,2285,1738,908,969,498,188,1556,1096,2210,1427,804,1010,1305,480,1119,2160,740,593,1963,621,1688,875,377,445,187,706,1219,1064,859,512,1771,288,80,2008,111,1854,2135,1806,2240,2067,1165,2317]
50-5-06	[1135,1774,957,1606,1953,1218,2167,1821,783,1453,1703,742,884,1280,1328,2209,122,1735,569,257,1869,191,2155,376,1094,1410,299,1531,847,512,1189,1973,1469,1520,1672,629,675,1352,543,1909,1008,375,214,1051,73,2115,2061,2247,439,1550]
50-5-07	[1633,1182,837,601,179,1397,704,862,1607,1672,1074,2057,1417,1534,1314,1010,1876,1513,1946,602,256,451,1087,938,274,750,997,1571,658,337,1707,1926,118,1920,457,225,571,417,1841,155,2054,1471,1765,2200,1250,2179,1999,1170,777,1289]
50-5-08	[821,1534,430,1908,1761,2154,1126,2012,332,1793,1402,1744,369,1024,684,887,120,1676,1455,2042,1379,790,982,2230,2124,1976,375,624,1238,473,897,333,1294,769,1513,260,454,1849,1003,1249,585,210,1661,2168,288,631,1473,1146,1187,91]
50-5-09	[278,1412,503,873,129,2035,1513,1261,1154,2130,1536,898,802,1335,734,225,2030,349,938,429,1736,2012,1081,542,1210,1440,702,1616,1748,397,1966,416,2182,1069,1034,1655,1684,2219,1588,607,1320,1900,1849,1817,680,1251,979,166,153,1889]
50-5-10	[717,2361,164,1014,2096,377,2458,465,626,478,264,814,1436,2243,297,1848,1372,2009,1588,2327,1043,1918,855,1792,1963,1991,2235,1760,1119,1431,1892,1197,2194,391,2055,1271,1514,1671,2355,915,1556,720,1324,215,325,1169,545,837,2431,656]

APPENDIX A5. (Cont'd.) Generated Due Dates for VRF Instances with 50 Jobs

Instances	Due Date of Jobs
50-10-01	[2485,2420,483,520,1664,2625,2233,2296,2724,2716,1850,2682,2839,1300,1713,2554,2436,1476,2101,1008,2137,909,1224,1629,1531,2180,1198,238,1908,610,1779,776,819,1672,2036,309,986,2973,462,879,339,1379,1059,2911,1931,1487,652,2350,566,1150]
50-10-02	[954,550,2255,2401,266,1411,1434,2105,524,3252,399,3308,2590,924,1484,1967,2287,1460,665,468,3000,1082,2939,1655,2674,1882,3087,1173,1327,2014,1053,1591,2738,2825,353,2171,1834,2578,2800,2266,772,1754,2052,3163,817,1259,1567,729,2925,2460]
50-10-03	[2804,820,239,1395,190,387,2752,2484,2834,916,1769,1194,1870,1166,1545,285,2124,2891,998,2972,2067,1262,322,1250,494,2422,2776,1373,2024,940,2220,1813,2578,1132,2683,666,2321,466,1509,648,785,1731,1950,2458,559,2197,773,2645,3031,1640]
50-10-04	[580,1733,1375,2248,2145,1539,2180,2718,3157,427,2100,2906,1133,538,1446,3292,1290,1072,2932,1978,2266,2502,3104,920,972,1380,2302,1205,354,1832,1524,2550,1678,1230,2680,1852,107,2025,2450,801,3293,1954,2527,3101,696,3192,740,325,1049,2735]
50-10-05	[371,2534,2941,1775,3012,1090,516,2314,1964,790,2572,3094,1486,2485,1565,2088,843,277,1929,3054,1383,435,577,2863,907,2756,972,1886,2696,224,2594,1352,687,3119,1116,2469,2200,2279,2039,1245,468,1231,1040,1511,2361,713,1325,1685,1862,2931]
50-10-06	[2986,2490,2054,1783,952,1084,1256,855,2816,1472,1902,1799,452,2343,3158,2372,2460,1330,885,92,2915,2948,3131,2629,3009,777,2234,1869,2102,1597,410,1987,2680,2577,1377,2142,1583,675,448,2859,365,702,1097,612,2532,2180,1163,260,1405,1737]
50-10-07	[2764,1427,1155,2125,864,1705,1601,1724,2496,2993,1383,2631,2264,1862,1044,2967,1553,649,924,2065,1193,2176,393,757,1780,719,282,2364,1647,480,2421,396,944,2854,1146,792,1474,1285,1802,224,2723,606,2547,506,2922,2310,2002,2767,1073,1978]
50-10-08	[1497,2497,1270,2242,2747,1603,3090,2580,1079,1889,2332,1044,1208,1630,1699,3120,244,2342,820,426,2635,340,3030,612,1427,1830,468,2073,1135,737,1591,2750,1913,2033,2309,904,1003,1718,781,2713,1329,501,382,1365,199,2978,2862,3150,705,2104]
50-10-09	[911,1480,2711,1993,2461,1135,423,1746,3037,520,2511,815,2077,2738,193,2314,1239,1044,955,3081,264,2142,1903,542,2873,2770,1950,534,641,2000,1142,3002,2197,2604,1854,1337,2237,2954,1643,731,1091,1529,1659,390,1356,831,355,2397,1553,2357]
50-10-10	[2292,1653,2493,1117,1522,2361,2671,1578,742,3046,1020,2475,2901,524,301,1926,2128,448,311,3203,699,2454,1150,1430,1385,2723,2750,2892,842,959,1256,2963,476,583,2171,1309,791,2210,3167,2754,3122,1763,1558,2574,1032,1906,2050,3040,1227,1731]

APPENDIX A5. (Cont'd.) Generated Due Dates for VRF Instances with 50 Jobs

Instances	Due Date of Jobs
50-15-01	[701,3037,2506,1256,2624,2123,789,667,2567,3636,3649,2956,2430,2820,887,3322,3111,1616,1884,1178,3462,1750,1972,619,1051,3071,2766,2238,1378,1786,1509,921,2195,2733,859,3502,2621,2067,536,1820,467,1320,3419,1459,2039,3145,2393,1514,1007,291]
50-15-02	[3517,1447,2451,1215,3273,942,658,567,2935,1680,3416,828,2023,3090,3693,2110,1875,1957,2546,882,2190,3145,1010,2598,2336,3731,1437,2254,1118,2809,405,3009,2411,3444,2675,329,1393,525,676,1643,1606,1041,3773,1335,3194,3628,2873,1733,1261,2714]
50-15-03	[2102,1395,703,372,3622,2625,1882,2960,3073,2766,3178,2635,2550,3361,2246,3567,2959,2427,2357,1824,886,3430,1114,1610,3591,1709,1201,3699,1998,826,1276,1752,1946,3491,2465,782,552,3752,961,2541,665,415,323,3279,3160,2111,2809,1149,1522,1443]
50-15-04	[1745,3375,2541,2571,1394,3366,1999,263,1153,2224,3215,949,1216,2381,2322,3091,3616,2155,3726,3178,863,1935,528,3242,940,2103,1519,1422,1242,599,2259,1634,1289,1359,3663,3877,3016,1900,2422,2807,2806,562,3382,787,2753,2856,1030,3572,692,1622]
50-15-05	[3464,1464,2418,1125,3264,896,591,472,2960,1779,3377,730,2033,3065,3671,2111,1871,1909,2553,749,2115,3169,925,2641,2283,3728,1383,2192,1029,2758,374,3025,2313,3430,2673,289,1296,444,624,1582,1553,986,3764,1270,3203,3494,2835,1805,1149,2705]
50-15-06	[2749,3692,2823,1331,3344,3277,1218,2363,2611,3468,1151,567,2619,458,1954,2236,302,1765,3386,1465,2494,862,650,3166,1007,2443,2213,2395,608,2911,1577,1323,429,1835,1581,3650,787,2150,758,1110,3564,3017,2107,1879,895,2901,1626,3399,3111,3201]
50-15-07	[1539,3308,2675,3658,2183,1026,2309,829,694,2777,3738,3085,2388,2975,1065,2944,1872,3209,3458,785,3561,588,1204,3795,3538,1633,1328,2013,2845,1745,906,1924,1865,1451,1386,2580,2321,2631,2440,417,3401,3128,612,1190,436,1673,977,2144,654,3229]
50-15-08	[2441,1087,1920,1998,2570,2188,1176,218,615,3408,498,1649,3137,2833,2634,752,2252,2896,964,1589,3844,1017,1888,2284,2103,481,1386,3274,3627,2684,2972,806,3822,664,694,3492,2049,2266,1555,1421,3487,3171,1761,1716,3697,2470,1306,1495,426,1251]
50-15-09	[2531,3209,3532,3338,2283,1574,2757,3243,1907,1480,793,2465,3609,2602,2630,2393,2782,3373,3034,1129,435,1263,1229,3547,1714,1079,1551,2692,1810,859,2214,1948,604,911,1121,3093,2169,3828,1696,2088,1329,391,2038,642,2353,3761,2892,3673,2963,3809]
50-15-10	[1230,2394,794,972,1130,1396,543,2921,3277,3299,1938,2463,2799,2574,3708,2765,3393,720,2094,574,2291,1017,2212,1271,1858,2070,1618,3060,594,489,3020,1034,1143,1726,1490,3756,3137,1551,3574,3090,2591,1623,380,810,1947,2972,2629,3699,1450,2349]

APPENDIX A5. (Cont'd.) Generated Due Dates for VRF Instances with 50 Jobs

Instances	Due Date of Jobs
50-20-01	[2977,3465,2907,2523,2552,2412,4446,4470,4212,3102,1796,1882,1081,749,2789,2056,4159,2781,1657,1932,2459,1024,2158,3955,1598,1208,3685,1318,3044,1731,982,997,650,1409,2279,2013,1705,1149,3232,756,538,3701,601,3393,3827,3350,4187,3781,2247,4249]
50-20-02	[751,2900,1208,1768,363,4128,3117,2624,2410,4277,3146,1900,1709,2775,1674,684,4100,828,1971,1134,3520,4059,2270,1319,2449,3010,1561,3236,3620,1012,3948,1032,4308,2256,2119,3254,3414,4334,3207,1357,2740,3829,3730,3696,1515,2535,2032,583,538,3793]
50-20-03	[3385,2724,4246,4186,2801,3903,4295,564,1165,1753,3404,4471,3970,3815,1686,1564,2218,1242,3072,1113,2201,784,3911,2663,2331,3748,1747,4349,448,754,2745,1575,2508,1892,3666,2592,733,1359,1254,3037,3164,605,2475,3229,2853,4001,501,2000,842,3532]
50-20-04	[1234,1288,4227,3109,2484,4231,2315,3578,2551,477,747,3196,2202,1522,2028,1006,1563,3436,1848,4324,3601,2770,4612,4018,842,668,2124,926,280,2861,2269,2665,3720,3847,2408,3247,4089,2810,1023,2957,624,4399,3515,1856,2580,3873,1978,3074,1968,1611]
50-20-05	[3584,2755,1212,4468,2945,1328,4193,1555,651,2274,2149,4235,3143,2170,3933,4080,828,2975,4411,1120,3863,873,2076,1468,2424,370,1745,3596,4007,3285,1598,2547,3658,3899,3498,3027,1433,1030,3444,2050,2210,2516,4517,2795,3174,1819,544,1937,474,2393]
50-20-06	[3290,2606,4273,4187,2791,3841,4336,692,1258,1808,3316,4471,4063,3771,1642,1513,2122,1412,3059,1217,2100,1124,3985,2567,2275,3670,1791,4428,581,925,2747,1596,2312,1946,3574,2461,801,1493,1459,3008,3102,696,2302,3138,2951,4098,609,2025,1175,3529]
50-20-07	[650,1191,492,2653,3996,4219,1005,4281,2551,2061,2732,937,2305,3291,3328,717,3030,1306,3969,1546,3879,1010,1691,2517,2867,1717,4615,2182,4249,3842,610,2613,1798,1516,1988,4501,2047,1527,3490,958,3155,4367,729,3572,2150,2892,1434,3793,3389,2408]
50-20-08	[3822,1214,428,2063,409,683,3734,3409,4074,1274,2516,1797,2636,1750,2196,486,2891,4101,1481,4131,2871,1994,613,1905,742,3303,3745,2034,2692,1394,3117,2576,3441,1566,3630,1042,3182,726,2185,930,1172,2460,2685,3360,870,3049,1128,3540,4233,2399]
50-20-09	[2008,1628,3969,2211,2822,3678,1792,3173,2191,3018,4076,2919,1691,1353,1613,3241,3234,2601,936,4163,915,2806,3935,583,450,2501,896,2066,2319,4397,796,4048,2903,3615,4296,648,1535,1242,2650,2775,4250,3402,2412,1131,3433,2011,4333,3718,1256,1908]
50-20-10	[2015,3490,1333,3381,4569,3520,1490,633,3897,2614,1942,1460,2452,2394,823,4232,2835,762,2367,1828,3133,2132,2895,4201,3344,3721,4277,1430,3869,1868,2506,3615,1723,945,925,3104,396,3241,2961,1136,1197,728,2070,2057,3990,1053,586,2661,1183,3770]

APPENDIX A6. Generated Due Dates for VRF Instances with 60 Jobs

Instances	Due Date of Jobs
60-5-01	[790,283,2043,1072,2815,612,2006,852,2280,162,2357,1128,1784,199,1613,1910,2186,380,1348,973,1949,2467,635,177,548,1689,484,1887,1468,2301,2112,2375,1767,795,2213,1189,2093,1420,2814,387,1538,589,1500,128,1,444,238,940,1178,2438,2635,1743,2556,2588,1016,2677,2738,1312,1236,2698,684]
60-5-02	[1874,1910,786,1169,309,584,260,1107,1638,2501,1790,2418,1409,1533,1609,447,878,2535,2630,1757,1051,16,88,418,684,2564,1260,1338,1042,535,2086,2228,36,511,2118,354,622,2272,2134,140,903,1953,275,1537,1492,866,2180,717,1282,1140,840,1745,2456,1367,2369,224,204,2029,2335,125,931]
60-5-03	[325,1497,2200,979,165,945,530,1001,1311,2003,2603,1929,1291,2075,419,743,2558,898,2625,1800,1088,1752,1645,508,1427,2371,2251,582,1392,643,462,237,1834,141,2452,2293,1559,2131,2535,1728,869,1154,1657,435,2328,823,2197,716,2624,1264,1206,677,2386,765,1347,1085,293,2044,1870,1447]
60-5-04	[1628,1969,529,76,1290,1308,461,572,2066,1937,2699,1980,2243,784,1435,2360,913,245,1140,2186,2409,1658,1033,2464,2542,484,1823,2136,2543,125,1220,843,334,712,1105,699,2743,607,204,2754,1009,1843,1706,1907,2655,1335,2173,1190,2606,295,2783,1515,814,1577,1385,2213,1587,428,993,1748]
60-5-05	[1366,2466,1931,2694,723,908,1548,431,249,2004,2745,2265,1600,2133,705,2085,977,2373,2550,1417,2659,61,9,1294,2793,2642,24,1360,846,2038,1723,589,1823,290,177,1113,770,1677,1624,813,1401,2519,1240,2273,184,8,1785,1886,1188,342,2216,531,122,2348,493,1620,643,1060,1071,1509,310,2434]
60-5-06	[1844,2522,1624,1892,108,2565,437,503,2397,2291,1278,591,287,1273,623,916,745,759,986,1317,173,1790,81,9,2440,252,520,698,1106,320,945,2198,1145,428,880,1350,395,2061,685,1965,2266,2458,1018,1899,462,2007,2249,2668,1513,1462,1733,2363,1662,1502,1451,2118,2613,197,1185,2696,1392]
60-5-07	[2513,1248,485,345,514,615,2477,2326,2545,898,1719,1389,677,1369,1652,1238,1892,2598,250,2644,1862,558,206,673,1059,2174,2491,157,1749,968,1968,1191,2372,607,2445,1618,2027,733,2098,1319,1807,292,2092,220,8,300,1484,1302,1481,2686,1568,437,911,2281,1110,1940,1108,2719,1698,817,2395]
60-5-08	[1729,2070,599,130,1338,1376,478,640,2194,2044,2824,2124,2389,800,1492,2429,980,310,1195,2312,2522,176,9,1080,2562,2636,530,1901,2222,2641,186,1278,980,414,767,1168,683,2838,647,205,2931,1055,1917,1826,198,0,2718,1436,2241,1218,2681,346,2997,1510,844,1594,1485,2327,1652,476,1007,1847]
60-5-09	[2212,1950,911,1432,973,2700,2362,1106,720,2425,1214,522,752,1733,1804,460,781,939,1023,1748,287,1634,2006,598,1438,1493,2512,2182,359,1399,2615,2148,520,2539,2690,1278,2340,2411,1537,1612,2594,408,1169,1881,2478,2061,1980,260,1254,1847,2076,1065,1359,843,589,144,1563,1663,1348,130]
60-5-10	[971,123,1665,2759,1797,2690,1644,1776,2548,2168,2306,1384,375,2211,1919,2122,686,1439,2036,2513,2748,2348,1208,627,436,1074,2636,2259,338,2094,619,731,840,1259,2038,684,1582,397,1479,257,2494,2705,1324,1,842,2248,2409,201,998,2421,2723,842,866,556,1088,1743,1514,1874,1127,1195,492]

APPENDIX A6. (Cont'd.) Generated Due Dates for VRF Instances with 60 Jobs

Instances	Due Date of Jobs
60-10-01	[1776,2560,693,2120,2729,852,2031,3243,1114,590,2478,2010,3353,3387,1326,1752,3724,3553,3200,387,2388,200,1001,1614,3137,3730,1218,3634,3053,1852,1539,1542,2977,3314,2691,1662,2154,2457,107,1488,461,1365,348,314,2815,3098,671,762,2916,2229,1964,1279,918,2646,833,3495,2374,2351,1072,1150]
60-10-02	[824,3532,406,3536,1418,586,1117,3392,3691,2305,3790,2330,2227,250,1588,2340,3741,1709,3895,1634,557,2612,3960,766,1022,267,906,2269,2179,2006,1402,1921,310,3294,652,513,3188,1189,3457,1514,735,2871,3444,2883,1764,1200,3832,2726,1807,3066,2619,1981,2572,2755,1291,1056,1830,3158,357,986]
60-10-03	[2540,3146,2181,2035,2962,3779,2972,861,3460,604,1334,3329,2323,2282,2752,3875,1550,3528,2558,240,749,2650,3861,1895,3319,2782,2728,1814,529,563,264,2276,2112,1047,3658,207,1065,3520,1230,1707,3554,2003,1421,1511,2045,1765,3063,350,900,353,1299,1138,3281,720,3140,818,1435,2883,3736,3724]
60-10-04	[661,2210,3350,2815,1219,1331,3201,880,2262,3122,2061,635,2337,1144,2137,2577,264,2849,1488,757,3745,2413,325,1884,2541,288,3727,943,671,1528,1621,1374,1631,3459,1832,490,1770,2996,3641,2725,2032,414,1185,2676,3637,545,2580,3151,1035,1005,1943,1559,3570,2762,3362,3045,1322,882,2887,2437]
60-10-05	[1951,3485,1919,2339,1260,3336,1060,1014,1281,1737,3075,503,595,834,128,2966,3750,2076,3844,2996,1569,1557,1884,3092,690,433,1083,944,476,1212,1778,312,1039,659,3796,3889,2833,869,2318,571,1993,1432,3529,1663,3204,3275,2587,3695,2238,1371,2683,2579,355,2533,2734,3229,1191,3552,1766,2102]
60-10-06	[2648,1264,2456,2278,1093,2879,1006,480,1312,573,611,2984,1869,1816,2782,2445,1474,381,1505,3267,2833,3754,1770,2659,3388,1722,904,2939,1836,862,3139,2311,3543,3248,3645,514,2522,1320,304,1569,3582,3483,1609,238,3663,2158,595,3115,1166,3418,708,3741,1120,2002,2230,768,2073,2049,2102,3342]
60-10-07	[2445,3410,1514,2334,2549,1982,3586,3640,2686,3149,2963,2528,1711,954,1163,3257,1944,751,1057,2716,582,3712,2019,1199,2765,250,3103,158,1829,3621,2271,2128,389,3292,1253,460,1388,3329,3539,1032,3778,2826,1643,828,2986,2891,3917,2300,370,1362,984,2025,1778,3839,1610,1559,540,3351,2906,739]
60-10-08	[888,3215,1140,3642,3329,205,2124,983,3114,1215,2021,3731,2906,1669,3194,3281,1373,2644,738,608,1342,3436,2295,3074,2784,692,224,1850,1654,1624,3667,3511,3591,1326,2836,3402,423,829,2615,308,2170,2072,557,791,2727,1887,2960,937,2949,2478,498,2199,2548,1036,1252,2317,1697,263,2529,1992]
60-10-09	[556,2440,3668,3720,2040,2577,1341,1880,2289,2208,763,1654,1244,3429,3054,282,3122,808,1210,2086,227,908,3308,435,2247,2589,1015,3461,2736,3360,2371,3301,1972,2911,3250,579,2532,1831,2000,1394,695,490,2980,2702,1409,2150,3157,1576,1672,2480,1073,1453,1850,344,2890,1119,459,2638,1156,3545]
60-10-10	[3784,3474,1977,2941,1878,1329,2871,1188,770,314,372,2492,3667,2188,667,626,3548,2460,1546,2229,2808,3249,2756,3300,2623,3333,3186,1456,2689,2143,1897,985,1281,1029,3644,3744,2321,2236,858,1682,2467,2714,782,906,3015,3041,1541,1100,553,1982,300,3471,547,1134,1834,738,2364,1815,2917,453]

APPENDIX A6. (Cont'd.) Generated Due Dates for VRF Instances with 60 Jobs

Instances	Due Date of Jobs
60-15-01	[1819,1590,2909,3870,3639,4539,1169,790,1592,1915,1053,952,3747,2743,3902,906,1893,2265,646,1326,3961,2364,4531,4441,492,3166,2335,1418,2796,4100,2921,615,3363,693,2428,2172,2620,2681,3510,196,4390,4286,1250,3690,1197,2127,3194,3260,1812,3281,1630,3612,3993,413,4141,2573,1459,3055,3552,2525]
60-15-02	[1361,4144,1649,2784,582,2324,4398,4327,3296,3496,3783,1889,4081,2388,3090,3192,928,1106,2264,3919,1048,487,2917,1625,1815,340,628,4370,3020,2529,3234,3836,3449,1272,2217,2045,648,4200,2390,1937,2068,1466,968,1067,641,3354,2709,2634,2995,386,4167,3707,2654,1994,1346,3963,1688,4000,3569,813]
60-15-03	[2638,4131,1639,2582,2828,2237,4208,4358,2884,3749,3465,2652,1934,986,1298,3858,2121,850,1203,2893,753,4457,2325,1384,3118,457,3532,385,2035,4291,2454,2439,559,3909,1458,650,1611,3949,4173,1118,4466,3129,1890,888,3489,3365,4588,2567,533,1506,1021,2384,1982,4560,1835,1733,722,4055,3409,808]
60-15-04	[4320,2824,2497,1421,3368,4504,1940,4240,4741,1874,3556,4174,4659,3114,675,3408,374,3898,2329,3194,758,4105,581,3678,660,1613,2556,4273,4308,3935,1037,3766,3608,1538,2886,2044,836,3317,1219,1344,3042,323,2266,1713,2817,3779,1487,4064,3098,2674,2396,1271,1762,4693,2720,4451,1025,1467,979,2097]
60-15-05	[1237,624,999,3713,4394,1002,3562,918,944,3038,451,3612,2669,4448,1288,4069,1835,1349,2330,4163,1532,2640,2898,463,3907,3068,2177,3647,3210,3845,2819,1570,4530,4010,522,1636,2531,4086,742,3329,1341,1856,4581,2428,2027,3716,1895,3142,1164,1741,2374,4365,4228,2944,867,2199,2521,4292,3365,683]
60-15-06	[1423,700,3191,1848,4494,1177,3110,1603,3557,389,3757,1936,2889,560,2620,3002,3459,765,2322,1720,3055,3912,1340,442,1147,2674,1014,2936,2400,3573,3426,3777,2819,1541,3483,2066,3296,2387,4421,787,2566,1175,2450,2174,950,576,1693,1978,3883,4244,2785,4069,4111,1773,4281,4339,2231,2117,4312,1410]
60-15-07	[2687,1660,1161,637,819,846,552,3271,4492,2120,3001,1390,3493,4662,2335,456,2393,2503,2608,588,2051,1875,659,1601,3290,1746,4640,4044,220,4311,3650,3202,372,3937,956,3862,1545,4262,2179,4525,1947,1492,1399,2443,2093,2249,3798,3585,3895,2665,1862,3093,2731,2768,4226,2893,3018,1118,1039,3634]
60-15-08	[3247,2520,3553,1695,2659,3320,3831,2162,573,4574,1915,3530,4203,1532,1313,2947,1025,1191,631,4699,2387,3343,2194,1399,484,3908,3929,4073,614,264,2431,4276,2291,2726,2105,996,2697,801,4651,4032,4615,2982,2382,2020,3023,4234,1386,2049,2913,1665,525,3589,3652,756,4386,1057,705,4366,1871,2779]
60-15-09	[2567,3910,2495,2976,1801,3900,1435,1244,1835,2257,3673,753,856,1034,374,3473,4347,2704,4414,3586,2104,2079,2434,3691,996,569,1538,1163,718,1636,2402,443,1317,920,4386,4430,3430,1108,2929,778,2637,2014,4055,2170,3700,3850,3232,4276,2840,1876,3288,3114,447,3063,3307,3776,1578,4141,2301,2720]
60-15-10	[4477,1761,2519,4339,1400,313,2655,1626,3806,587,3235,2727,904,2591,3541,2826,3959,1625,4565,4002,1258,2201,4166,2408,3164,932,2728,1397,827,4157,2262,1945,3224,3423,1502,514,2341,4458,3972,3779,3360,3554,2141,451,2552,1082,1291,2974,1127,3667,4318,421,727,1654,1724,2962,998,2058,684,4634]

APPENDIX A6. (Cont'd.) Generated Due Dates for VRF Instances with 60 Jobs

Instances	Due Date of Jobs
60-20-01	[2273,1607,1373,2776,407,1021,3255,4592,1538,4908,2616,4638,881,3148,2328,5190,3849,578,1420,3945,475 4,4224,4594,499,1103,2819,5206,2166,5076,1827,2114,5116,4482,2929,4522,5316,1190,3501,3812,721,4359,3 612,2424,3165,4140,1273,3724,2537,3354,3473,1393,3031,1913,936,4159,1888,2578,695,4915,1937]
60-20-02	[1475,4511,1740,5004,4623,490,2981,1592,4292,1755,2731,5183,4010,2368,4325,4590,2232,3736,1284,1136,2 097,4724,3176,4241,3842,1209,716,2593,2337,2305,5102,4846,4931,2082,3938,4697,886,1366,3696,818,3046, 2910,1025,1356,3774,2666,4114,1496,4063,3362,989,3107,3409,1605,1850,3341,2564,749,3408,2677]
60-20-03	[5259,2651,1061,543,2451,3392,1743,1580,4643,5582,2149,3983,2680,3491,3509,5393,4148,817,2388,3114,14 38,755,2800,1092,3892,4971,1257,714,5129,4854,2723,533,3003,3850,3265,1309,5476,1522,615,1381,4337,36 56,4366,4074,2230,1744,923,2122,4778,5154,2272,3659,4735,3025,2969,4687,4458,2086,4297,4490]
60-20-04	[1801,3624,1891,2004,5210,1548,3155,2678,1064,5462,4346,4838,916,2092,611,2999,1608,2068,1313,2544,13 55,3801,2357,2889,752,3187,2313,5495,5118,407,5528,4717,2816,651,3193,4602,1714,4896,618,3993,4503,12 40,4998,3525,2630,1396,4008,4553,2656,4205,2914,4040,3829,1114,4676,1094,5129,5251,2166,4303]
60-20-05	[2243,5311,2965,4155,456,2882,5205,2736,4951,2611,5177,2010,813,5320,2051,5111,2570,4824,2106,723,258 2,3321,3747,4626,1567,1607,3973,3090,987,3238,3483,1758,4071,4641,4740,4955,3837,1228,3187,1405,4536, 2339,4572,3954,4413,1945,1011,4205,3787,657,1653,4240,514,2800,1439,3522,3019,1212,766,1089]
60-20-06	[3552,1764,3494,3077,1522,3899,1493,686,1781,904,1113,4015,2680,2522,3776,3316,2115,676,2128,4384,384 0,5159,2408,3694,4646,2338,1429,3965,2593,1363,4241,3174,4842,4298,4993,860,3507,1969,635,2281,4927,4 709,2292,558,5042,3000,1040,4111,1615,4660,1232,5109,1605,2694,3032,1287,2884,2734,2929,4424]
60-20-07	[5094,1833,3030,1619,5283,3705,873,2767,4646,2254,2319,2135,800,2095,4222,1884,1330,3185,4316,3127,46 97,2043,1522,2541,4127,1606,1928,3617,2584,4446,493,4574,3060,1746,3865,2660,5048,605,4977,3989,1020, 3490,1243,4814,5216,4218,2366,3237,4035,1673,882,3817,3603,5312,2810,4477,1390,693,1210,3406]
60-20-08	[5123,4884,2673,4194,2419,1989,4141,1870,1192,532,555,3604,5092,2888,1005,999,4913,3348,2250,2935,401 6,4555,3915,4640,3644,4727,4490,2063,3713,2815,2525,1513,1931,1575,5068,5098,3193,3017,1376,2300,3543 ,3800,1258,1418,4336,4445,2201,1754,826,2726,373,4807,807,1776,2404,1129,3321,2365,4186,605]
60-20-09	[3813,2247,3265,3308,3987,2461,2150,1909,1170,4865,1309,2862,4695,4246,4024,1002,3595,4320,1684,2785, 5296,1856,3081,3685,3562,1588,2617,4785,5171,4042,4474,864,5284,2352,3044,5050,626,489,742,570,5010,4 745,3884,1529,5191,4471,2536,2708,445,796,2636,4199,1345,3023,2186,900,1976,4153,2105,1605]
60-20-10	[3129,4670,2023,3126,3462,2769,4894,5033,3549,4253,3935,3423,2357,1318,1755,4353,2742,1090,1653,3606, 942,5156,2784,1814,3709,523,4159,510,2524,4927,2994,2973,793,4476,1892,857,1966,4519,4744,1476,5274,3 747,2244,1296,3984,3808,5417,3060,699,1931,1338,2869,2480,5327,2210,2133,895,4646,3873,1044]