

Workload Balancing for Airline Flight Dispatchers

by

Serkan Turhan

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Management Sciences

Waterloo, Ontario, Canada, 2025

© Serkan Turhan 2025

Examining Committee Membership

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

External Examiner: Loubna Benabbou
 Professor
 Dept. of Management Science
 Université du Québec à Rimouski

Supervisors: Samir Elhedhli
 Professor
 Dept. of Management Science and Engineering
 University of Waterloo

 Fatma Gzara
 Professor
 Dept. of Management Science and Engineering
 University of Waterloo

Internal Members: Saeed Ghadimi
 Assistant Professor
 Dept. of Management Science and Engineering
 University of Waterloo

 Houra Mahmoudzadeh
 Associate Professor
 Dept. of Management Science and Engineering
 University of Waterloo

Internal-External Member: Shi Cao
Associate Professor
Dept. of Systems Design Engineering
University of Waterloo

Author's Declaration

This thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Statement of Contributions

Chapter 3 of this thesis has been submitted for publication as a journal paper currently under review (Turhan, Elhedhli, and Gzara, 2025). This chapter was co-authored with my supervisors, Dr. Samir Elhedhli and Dr. Fatma Gzara. My contributions include problem conceptualization, methodology development, computational implementation, analysis of numerical results, and drafting the manuscript. Dr. Samir Elhedhli and Dr. Fatma Gzara provided guidance on conceptual direction, methodological oversight, critically reviewed the manuscript drafts, and contributed to the editing of the final submission.

I am the sole author of the rest of this thesis.

Abstract

We study the flight dispatcher scheduling problem with the objective of balancing workload. Flight dispatchers are responsible for flight planning and monitoring, and their duties are characterized by multi-tasking and uncertainty. We propose novel models, solution methods and a detailed analysis.

In Chapter 2, we introduce and analyze the flight dispatcher workload balancing problem. We model dispatcher workload under two objectives: minimizing peak workload and minimizing absolute workload deviation. To manage large problem sizes, we decompose the problem using Lagrangian relaxation into tractable subproblems and derive a closed-form expression for the Lagrangian lower bound of the peak workload objective. We then propose a focus-search-improve genetic algorithm, which integrates genetic search with refinement heuristics. Numerical tests on real-world flight data show that the heuristic finds optimal or near-optimal solutions and outperforms commercial solvers. It is found that minimizing peak workload reduces workload spikes while minimizing absolute deviation distributes workload uniformly.

In Chapter 3, we incorporate practical and operational constraints such as route-based flight grouping and assignment restrictions based on flight characteristics. We provide a multi-objective model and propose three solution approaches, the weighted-sum method, the ϵ -constraint method, and the non-dominated sorting genetic algorithm II, to capture the trade-off between workload balance and operational efficiency. Computational experiments show that non-dominated sorting genetic algorithm II generates clear Pareto fronts and provides solutions that are comparable to those found by exact methods.

In Chapter 4, we focus on the absolute deviation workload balancing problem which achieves balanced schedules but is computationally challenging. We propose a neural network-assisted column generation algorithm that predicts negative reduced cost solutions of the Lagrangian subproblems. Preliminary results demonstrate the viability of the approach in speeding up the column generation algorithm.

Acknowledgements

I would like to express my sincere gratitude to my supervisors, Dr. Samir Elhedhli and Dr. Fatma Gzara, for their invaluable guidance, support, and encouragement throughout the course of my studies. Their mentorship has been instrumental in shaping both the direction and quality of this work.

I am also thankful to the members of my examining committee, Dr. Saeed Ghadimi, Dr. Houra Mahmoudzadeh, Dr. Shi Cao, and Dr. Loubna Benabbou, for their time, insightful feedback, and constructive suggestions, which contributed significantly to the refinement of this thesis.

I am indebted to Dr. Sibel Alumur Alev for her support and for guiding me toward my supervisors during my search for a doctoral program, as well as to Dr. Safa Erenay for his continued encouragement and kindness throughout my graduate studies. I also wish to acknowledge Zivojin for his consistent assistance with IT-related matters and the Digital Research Alliance of Canada for providing access to computing resources and responsive technical support.

I am grateful to my friends and peers who have accompanied me at various stages of this journey. I would like to thank: Aniket, Atahan, Burak, Emre, Esma, Goksu, Jeremy, Jonny, Omer, Onur, Owen, Shechem, Shri, Shubhi, Sinem, William, Yuce, Zeynep, as well as the members of 9RoundKW and the UW Ballroom Dance Club. Their companionship, encouragement, and humor have been a source of *balance* and joy throughout the challenges of graduate life. I am especially thankful for the steady support and thoughtful presence of close friends who were there when it mattered most.

Finally, I extend my deepest appreciation to my family. To my mother, father, and brother, Kerem, thank you for your unconditional love, patience, and unwavering belief in me.

Dedication

To those affected by airline incidents. Their experiences remind us of the profound responsibility inherent in operational decisions, including those entrusted to flight dispatchers.

Table of Contents

Examining Committee	ii
Author's Declaration	iv
Statement of Contributions	v
Abstract	vi
Acknowledgements	vii
Dedication	viii
List of Figures	xiii
List of Tables	xvi
1 Introduction	1
2 Workload Balancing Problem for Flight Dispatchers	7
2.1 Literature Review	11
2.2 The Dispatcher Workload Balancing Problem	15
2.3 Lagrangian Relaxation and Lower Bounding	21
2.3.1 Properties of the Lagrangian Subproblem	23

2.3.2	The Optimal Lagrangian Bound	27
2.4	A Lagrangian Guided Focus-Search-Improve Heuristic	29
2.4.1	The Search Phase	29
2.4.2	The Improvement Phase	31
2.4.3	The Focus Phase	33
2.5	Numerical Testing	34
2.5.1	Flight Data and Workload Determination	34
2.5.2	Quality of Solutions	38
2.5.3	Quality of the Workload Balance	42
2.6	Conclusions	48
3	A Multi-Objective Framework for Flight Dispatcher Workload Balancing	52
3.1	Literature Review	53
3.1.1	Multi-objective Optimization Methods	54
3.2	Problem Setting	55
3.2.1	Flight Path Similarity	56
3.2.2	Assignment Restrictions	59
3.2.3	Illustrative Example and Operational Objectives	59
3.3	Mathematical Model	61
3.4	Solution Approaches	64
3.4.1	Weighted Sum Method	64
3.4.2	Epsilon-Constraint Method	65
3.4.3	Non-dominated Sorting Genetic Algorithm II (NSGA-II)	66
3.4.4	FSI-GA Adaptations	66
3.5	Numerical Analysis	66
3.5.1	Benchmark Instances, Experimental Setup, and Implementation De- tails	67
3.5.2	Analysis on a Representative Small Instance	68

3.5.3	Problem Complexity and Performance on a Large Instance	72
3.5.4	Multi-objective Analysis on Benchmark Instances	75
3.5.5	Sensitivity Analysis of Operational Assumptions	82
3.6	Conclusions	93
4	A Machine Learning-based Column Generation Approach for Flight Dispatcher Workload Balancing	95
4.1	Literature Review	96
4.1.1	Machine Learning in Combinatorial Optimization	96
4.1.2	Machine Learning for Column Generation	97
4.1.3	Machine Learning for Unconstrained Nonlinear Binary Optimization	98
4.1.4	Objective-Informed Loss Function Design	99
4.2	Methodology	100
4.2.1	Column Prediction with Neural Networks	102
4.2.2	An Alternative Sign Model	104
4.3	Numerical Results	105
4.3.1	Results of the Column Generation	105
4.3.2	Results of the Solution Prediction Approach	106
4.3.3	Preliminary Results of the Sign Model	109
4.3.4	Next Steps on Neural Network Architecture and Fine-Tuning Parameters	111
4.4	Conclusion	113
5	Conclusions and Future Research Directions	115
	References	118
	APPENDICES	128

A Chapter 2 Appendix	129
A.1 Focus-Search-Improve Genetic Algorithm	129
A.2 Focus Procedure of the Algorithm	130
A.3 Search Procedure of the FSI-GA	131
A.4 Improvement Procedure of the FSI-GA	132

List of Figures

2.1	Illustration of an example workstation schedule with 2 dispatcher shifts. . .	9
2.2	Variation of required load for three flights, departing at 6:08 am, 6:52 am, 10:08 am, respectively.	16
2.3	Comparison of workload in two different assignments where workstation assignments of flights are (1, 1, 2) and (1, 2, 2), respectively.	17
2.4	Flight load requirements and distribution of workload in a solution for an instance with 50 flights and 2 workstations.	18
2.5	Overview of FSI-GA.	30
2.6	An example of uniform crossover operation on two solutions for 5 flights and 3 workstations.	31
2.7	An example mutation operation on a solution for 5 flights and 3 workstations.	32
2.8	Distribution of the number of flights in test instances.	35
2.9	Planning and en-route phases over the operational hours in the problem instance of Delta Air Lines 4 Sep 2019.	37
2.10	Distribution of planning and en-route load requirements of flights in the generated instances, categorized by carrier.	37
2.11	Heuristic time and instance size.	42
2.12	Summary of the FSI-GA performance over varying flight to time-point ratio $\frac{ I }{ T }$	43
2.13	Distribution of flight to time-point ratio among instances of each carrier.	43
2.14	Planning and en-route phases over the operational hours in the problem instance of Delta Air Lines 7 Sep 2019.	44

2.15	Distribution of workload in the optimal solution for (PWP) of R100 Delta Air Lines 7 Sep 2019 instance.	45
2.16	Distribution of workload in the heuristic solution for (ADP) of Delta Air Lines 7 Sep 2019 instance.	45
2.17	Distribution of the number of flights per workstation for different carriers in the FSI-GA solutions for (PWP).	48
2.18	Distribution of the workload for different carriers in the FSI-GA solutions for (PWP).	49
2.19	Distribution of the number of flights per workstation for different carriers in the FSI-GA solutions for (ADP).	49
2.20	Distribution of the workload for different carriers in the FSI-GA solutions for (ADP).	50
3.1	Illustration of the flight paths of the flights provided in Table 3.1.	58
3.2	Illustration of flight paths and regions for radius $\rho = 150\text{km}$	60
3.3	Flight paths and airport connections.	69
3.4	Non-dominated ADP z_1 and Similarity Score z_4 objectives from the tested methods in the Allegiant Air instance.	71
3.5	Comparison of non-dominated z_1 and z_4 objectives with overall non-dominated objectives from all methods in the Allegiant Air instance.	71
3.6	Comparison of non-dominated z_1 and z_4 objectives with overall non-dominated objectives from all methods in the United Air Lines instance.	73
3.7	Non-dominated ADP z_1 and Similarity Score z_4 objectives from the tested methods in the Allegiant Air instance.	74
3.8	Distribution of Pareto front sizes obtained by NSGA-II in R50 instances, intervals highlight the carriers of instances.	77
3.9	Distribution of Pareto front sizes obtained by NSGA-II in R100 instances, intervals highlight the carriers of instances.	79
3.10	Algorithm time distributions across tested categories.	87
3.11	Pareto front size distributions across tested categories.	87
3.12	Comparison of Hypervolume ratio measures across categories.	88

3.13	Comparison of (a) ADP and (b) PWP solutions for (2.5h, 5min, 3 airports, 25%, 150km) configuration.	89
3.14	Comparison of (a) ADP and (b) PWP solutions for (2.5h, 5min, 5 airports, 25%, 50km) configuration.	89
3.15	Comparison of (a) ADP and (b) PWP solutions for (2.5h, 5min, 3 airports, 25%, 50 km) configuration.	90
3.16	Comparison of 50-150-300 km solutions on 300 km similarity score objective for (5min, 5 airports, (a) 0% and (b) 50%, 300 km, ADP) configurations. . .	90
3.17	Comparison of 50-150-300 km solutions on (a) 50 km and (b) 150 km similarity score objective for (5min, 5 airports, 25%, ADP) configurations. . . .	91
3.18	Comparison of long-haul separation solutions for (5min, 3 airports, 25%, 300 km, ADP) configuration.	92
3.19	Comparison of departure proximity solutions for (3 airports, (a) 25% and (b) 50%, 300 km, ADP) configuration.	92
4.1	Overview of machine learning applications in column generation.	101
4.2	Visualization of the alternative smooth approximations and the absolute value function.	103
4.3	Solution time distribution for the column generation algorithm on the Alaska Airlines Feb 11th R100 instance, showing Subproblem (SP) and Master Problem (MP) solution times per iteration.	107
4.4	Column generation using the neural network predictor (trained with MSE SP loss) on a 50-flight Alaska Airlines 2019-1-16 instance with two workstations: (a) Upper bound per iteration, (b) Upper bound over time. Fallbacks to the exact solver occur when no negative reduced-cost column is found in 200 attempts.	108
4.5	Subproblem runtime as a function of the portion of s_t^* signs provided to the sign model.	109

List of Tables

2.1	Flight summary statistics of the test instances, categorized by carriers. . .	35
2.2	Computational performance of the FSI-GA solution for (PWP) compared to the Cplex incumbent solution and Lagrangian lower bound for R50 instances.	40
2.3	Computational performance of the FSI-GA solution for (PWP) compared to the Cplex incumbent solution and Lagrangian lower bound for R100 instances.	41
2.4	Comparison of (PWP) and (ADP) solutions for R50.	46
2.5	Comparison of (PWP) and (ADP) solutions for R100.	47
3.1	An example set of flights.	57
3.2	An example set of flights.	60
3.3	Flight summary statistics of the test instances, categorized by carrier. . . .	67
3.4	Pareto front statistics for the Allegiant Air instance	70
3.5	Payoff table obtained by solver for the instance Allegiant Air Feb 9 2019. .	72
3.6	Pareto front statistics for the United Air Lines instance	73
3.7	Payoff table obtained by NSGA-II for the instance United Air Lines Jan 8 2019.	74
3.8	Pareto front statistics of the solutions obtained by NSGA-II for R50 instances.	76
3.9	Pareto front statistics of the solutions obtained by NSGA-II for R100 instances.	78
3.10	Summary of NSGA results for R50 instances for different multi-objective sets.	80
3.11	Summary of NSGA results for R100 instances for different multi-objective sets.	81
3.12	Overview of the configurations tested.	83

3.13	Algorithm run time (s) per configuration category.	84
3.14	Pareto front size statistics per configuration category.	85
3.15	Average Hypervolume ratio measures across categories.	86
4.1	Summary of key machine learning applications in column generation	98
4.2	Runtimes for baseline and sign model under varying portions of s_t^* provided.	110

Chapter 1

Introduction

Air transportation remains a critical component of the global economy, influencing international trade, tourism, and employment significantly. In a typical year, the commercial aviation sector transports billions of passengers and vast quantities of cargo, directly contributing substantial portions of global GDP and supporting tens of millions of jobs (ICAO, 2020; ABBB, 2020; ATAG, 2020). The scale of these global operations is immense, as industry reports indicate approximately 130,000 flights occur daily (IATA Economics, 2022), and within the U.S., the Federal Aviation Administration estimates over 45,000 daily flights (FAA, 2023). This vast system, however, faces persistent challenges including evolving weather patterns, unplanned technical issues, complex regulatory changes, and labor constraints that can further complicate operations (IATA, 2024). Despite this economic importance, the airline industry remains vulnerable to external shocks. For instance, the COVID-19 pandemic resulted in substantial disruptions due to lockdowns, travel restrictions, and stringent health measures, causing severe financial impacts on passenger carriers while temporarily benefiting cargo airlines due to increased freight demand (IATA, 2021; Bouwer, Krishnan, and Saxon, 2020; Bouwer, Krishnan, Saxon, and Tufft, 2022). By 2023, commercial aviation largely recovered to pre-pandemic operational levels. However, enduring challenges persist, including elevated debt levels, workforce management issues, and evolving regulatory environments driven by safety, health, and environmental considerations (Harper, 2023).

The management of airline operations requires the coordination of numerous complex processes. Airlines must design schedules accounting for aircraft availability, crew assignments, passenger demand fluctuations, regulatory compliance, and varied operational conditions. Day-to-day operations include aircraft maintenance inspections, continuous monitoring of weather and air traffic conditions, crew dispatching, and rapid responses to disrup-

tions such as technical failures or severe weather events (Trapote-Barreira, Deutschmann, and Robusté, 2016). Each operational decision directly influences overall safety, reliability, and efficiency.

Flight dispatchers play a central role within this operational ecosystem, acting as a critical link between airline management, flight crews, and air traffic control. Operating from Integrated Operations Centers (IOCs), dispatchers are pivotal in managing daily flight operations. Their role involves close collaboration not only with pilots for flight planning and ongoing monitoring but also with other operational teams. During disruptions, such as those caused by severe weather, airspace restrictions, or aircraft substitutions, dispatchers are central to various time-sensitive decisions. Positioned within these IOCs, they must facilitate this critical inter-team coordination, often under pressure, which can lead to significant spikes in their workload. Prior to departure, dispatchers create detailed flight plans that account for routes, altitudes, fuel requirements, and contingency airports. They analyze real-time meteorological conditions, airspace restrictions, and airport availability to ensure each flight plan meets safety and cost-efficiency standards (Munro and Mogford, 2018). Throughout flights, dispatchers track aircraft progress, advise flight crews on route adjustments to avoid adverse conditions, and coordinate necessary responses to operational contingencies. They hold legal authority in many jurisdictions to delay or cancel flights when safety concerns arise, emphasizing the significant responsibility inherent in their role (Gov. of Canada, 2025; Souza, 2024).

Given their extensive responsibilities, dispatchers undergo rigorous training and certification. In jurisdictions such as the United States, dispatch certification involves intensive specialized training covering aviation regulations, aircraft systems, navigation, communication protocols, meteorology, and human performance considerations (U.S. Code of Federal Regulations Title 14, 2022a; IFOD, 2025). Dispatch centers themselves employ advanced infrastructure, including redundant communication and power systems, ensuring operational continuity and situational awareness under all conditions (Sloan, 2013; Level-1, 2022).

A significant challenge for dispatchers involves workload management. Unlike pilots or air traffic controllers who typically manage flights or sectors sequentially, dispatchers often simultaneously oversee multiple flights. Dispatcher workload includes flight planning, real-time monitoring, coordination with flight crews, and responses to operational disruptions. Workload intensity fluctuates significantly, often intensifying during peak operational periods or weather-related disturbances and decreasing during calmer intervals. For instance, adverse weather conditions require dispatchers to continuously monitor evolving meteorological systems, anticipate the potential impacts on numerous flights, and implement adjustments to flight plans (Sheffield Sch. of Aeronautics, 2016). Beyond weather, dis-

patcher workload is also affected by other factors such as maintenance alerts and crew scheduling issues, requiring time-critical decisions that must balance safety, operational performance, and personnel well-being. Excessive dispatcher workload can lead to task saturation, potentially compromising operational safety and reliability. Conversely, balanced workloads enhance situational awareness and operational performance. Consequently, effective workload management for dispatchers is crucial. Excessive stress resulting from high workload can adversely affect their cognitive performance and decision-making capabilities, impacting overall airline safety and efficiency. In such an interconnected operational system, overlooking dispatcher workload can lead to compromised safety and reduced airline reliability, particularly when disruptions escalate. Historically, airlines have managed dispatcher workload using informal strategies and managerial judgment. However, increasing network complexity and rising operational demands necessitate more systematic and analytical decision-support tools for dispatcher workload allocation.

Before formally defining the flight dispatcher workload balancing problem, let us establish the research hypotheses that guide the conceptualization and quantification of workload within this study. The central premise is that for modeling and optimization purposes, dispatcher workload can be represented by the demands stemming from their tasks in flight operations and their characteristics.

This research is guided by several key hypotheses for dispatcher workload. The primary drivers of workload are considered to be the essential tasks of pre-departure flight planning and en-route flight monitoring, with intensity varying by flight phase and characteristics such as flight distance. Furthermore, subjective factors like individual dispatcher experience, while influential in practice, are excluded to maintain a focus on the base problem. While unforeseen operational events, including adverse weather conditions or unexpected flight delays, introduce considerable uncertainty, their direct, dynamic impact is acknowledged but not explicitly modeled as discrete new workloads. Instead, the general effect of such stochastic events is considered implicitly within the baseline load defined for en-route flight monitoring tasks.

Consequently, for the purposes of this thesis, flight dispatcher workload is defined as a quantifiable, unitless, and time-varying measure of the cumulative demands associated with the pre-departure flight planning and en-route monitoring tasks for a given set of flights assigned to a workstation, derived from flight characteristics and operational phases. This definition forms the basis for the mathematical models and solution methodologies developed. Conceptually, the task load associated with managing a flight in this study encompasses the overall demands it places on a dispatcher, which stem from both cognitive and physical effort. While a precise, direct measurement of such effort is complex, this load is primarily approximated using observable flight characteristics, with flight distance serving

as a key proxy due to its correlation with factors influencing these demands. The detailed rationale for utilizing distance and other contributing factors such as flight path, specific timing, aircraft type, and airport characteristics is further elaborated in Section 2.5.1.

It is important to distinguish this operational definition of workload, focused on quantifiable task demands suitable for optimization modeling, from the broader conceptualizations typically found in human factors research. The human factors perspective, as explored in established literature (Hart and Staveland, 1988; Wickens et al., 2021), often treats workload as a multifaceted construct. This view encompasses detailed aspects of cognitive load, including elements like attention, memory, and decision-making demands, alongside perceived mental effort, stress, situational awareness, and other psychological and physiological states critical to performance in complex systems (Parasuraman and Rizzo, 2006; Young and Stanton, 2002). While these human factors elements are vital for a comprehensive understanding of a dispatcher’s overall experience and performance capacity, this thesis adopts a more abstracted, task-based representation of workload. This approach is a necessary simplification to provide a tractable and quantifiable basis for developing and applying the optimization models aimed at balancing the distribution of assigned flight-related tasks, rather than attempting to capture the full spectrum of human workload perception.

The urgency for developing such improved tools is underscored by recent industry trends and events. For instance, a 2024 recommendation from the National Transportation Safety Board advocated for expanding the requirement for certified dispatchers to smaller air carriers, citing anticipated safety benefits and improved flight monitoring (Lynch, 2024). Such a policy, if broadly adopted, would likely increase the overall demand for dispatchers and further underscore the critical importance of effective workload distribution strategies. Moreover, real-world events, such as the early 2024 strike authorization vote by Air Transat flight dispatchers related to concerns over working conditions and compensation (Dove, 2024), illustrate the significant operational and personnel challenges linked to dispatcher workload. These situations can lead to system-wide operational bottlenecks and highlight the pressing necessity for approaches that better balance demanding operational requirements with dispatcher well-being.

Despite their essential role, optimal assignment of flights to dispatchers for workload balancing remains underexplored in academic literature. Previous research predominantly addresses crew scheduling, aircraft maintenance routing, and air traffic flow management, with limited focus on dispatcher workload distribution. When addressed, existing dispatcher scheduling models often rely on oversimplifications, such as assessing workloads only at discrete intervals or neglecting flight-specific variability beyond scheduled times (Rayner, 2021). These simplifications fail to represent the continuous, multitasking nature

of dispatcher responsibilities, leading to impractical or inefficient assignments in practice. Addressing this gap motivates the current research.

This thesis aims to develop and evaluate optimization models and methodologies that effectively balance dispatcher workloads to enhance airline operational safety, efficiency, and resilience. Specifically, the objectives are to formulate realistic models that accurately represent dispatcher workload dynamics, develop both exact and heuristic algorithms capable of solving large-scale problems efficiently, extend initial formulations to include multiple operational criteria and constraints, and explore computational techniques, including machine learning, to address complex problem instances. By achieving these objectives, this thesis seeks to provide valuable insights and practical decision-support tools for improved dispatcher workload management.

The remainder of the thesis is organized as follows. Chapter 2 reviews the literature on workload balancing and airline scheduling with an emphasis on flight dispatchers. It introduces the Flight Dispatcher Workload Balancing Problem and presents a mathematical model designed to minimize the maximum workload across workstations during the day. A Lagrangian relaxation is applied to derive a closed form lower bound, and a custom genetic algorithm, Focus-Search-Improve, is developed to obtain high quality solutions through incremental problem complexity and refinement using insights from the relaxation. A procedure to generate realistic test instances from historical flight data is also described. Computational results indicate that the heuristic achieves solutions within 3–4% of the Lagrangian bound and outperforms a state of the art solver, especially on large instances. This chapter marks the first comprehensive optimization approach for balancing dispatcher workload, demonstrating that minimizing workload deviations significantly improves fairness compared to simpler assignment methods.

Chapter 3 extends the model by incorporating additional operational considerations and multiple objectives. A multi objective framework is proposed that integrates workload fairness, flight grouping similarity, and assignment constraints. This extended model acknowledges that dispatcher efficiency improves when flights sharing similar routes or weather conditions are grouped together, thus reducing redundant monitoring, and when conflicting flights are separated. Three solution methods are examined: a weighted sum approach, an epsilon constraint approach, and an evolutionary algorithm based on NSGA-II. The Focus-Search-Improve heuristic is adapted for the first two methods, while NSGA-II operates independently. A review of related multi objective optimization literature is provided, and extensive computational experiments reveal that NSGA-II rapidly generates a wide Pareto frontier that exposes trade offs between workload balance and flight grouping similarity. In contrast, the weighted sum and epsilon constraint methods, particularly when employing exact solvers, yield fewer efficient solutions and face scalability challenges.

This chapter contributes a novel multi objective model that integrates dispatcher workload with practical operational factors and demonstrates the effectiveness of metaheuristic approaches in balancing multiple assignment objectives.

Chapter 4 tackles the computational challenges associated with the workload deviation objective by incorporating machine learning into the solution process. Exact methods struggle with even moderately sized instances, prompting the reformulation of the Lagrangian relaxation into a prediction task. A neural network is designed and trained to predict near optimal assignments for the Lagrangian subproblem using data generated via column generation. Inputs to the neural network include flight schedule details and dual prices, and its output is a candidate assignment expected to lower the reduced cost. These predictions serve as effective starting solutions or warm starts within the column generation process, thereby accelerating convergence. The chapter describes the neural network architecture, training procedures on both small and large instances, and the integration of these predictions into the overall algorithm. While traditional column generation faces difficulties on large scale problems, the machine learning assisted approach improves both solution times and quality by directing the search toward promising regions. This work introduces an innovative hybrid method that leverages machine learning to address complex NP hard scheduling problems, with potential applications in other resource allocation challenges.

Chapter 5 summarizes the key findings and contributions of this research and outlines potential avenues for future work. The models and algorithms developed herein advance the field of flight dispatcher scheduling. Future research may explore alternative workload metrics, integrate dispatcher assignment with crew scheduling, incorporate stochastic elements to better address operational disruptions, and further apply machine learning to enhance optimization processes. These directions aim to develop decision support systems that improve the safety and efficiency of airline operations.

Chapter 2

Workload Balancing Problem for Flight Dispatchers

The airline industry is a driving force for global economic growth with around 8.5 billion passengers in 2023 and a yearly demand growth of 13.8% in March 2024, measured in revenue passenger kilometers (ACI, 2024; IATA, 2024). Despite facing difficulties in the past, such as fluctuating fuel prices and economic downturns, the COVID-19 pandemic has been the most severe challenge it has encountered. In an attempt to recover to pre-pandemic levels, airline companies are exploring a multitude of cost saving options, including optimizing operational efficiency.

Commercial aviation is a complex business that requires teams of professionals working around the clock to ensure optimal operations planning and safe and efficient execution. While passengers interact with front-end staff such as flight attendants and pilots, there are behind-the-scene roles crucial to flight operations. One such role is that of flight dispatchers who are responsible for flight planning before departure and flight following when en-route. They make critical decisions on flight operations and share legal responsibility on flight operation with pilots. Hence, they require rigorous training and certification. To obtain a flight dispatcher license in the U.S., one must undergo 200 hours of training and is required to pass a knowledge test in meteorology, navigation, communications, air traffic control federal aviation regulations, aircraft operations, among others (U.S. Code of Federal Regulations Title 14, 2022a).

Flight dispatcher tasks may be broadly categorized into two main phases based on the status of flights: pre-departure planning and en-route following. Before departure, dispatcher planning tasks include: fuel, passenger, and cargo load for efficiency; flight

adherence to regulations and standards; and flight path determination and release. In particular, dispatchers plan flight path directions and restrictions and timing constraints. They calculate fuel requirements, decide on loading additional payload, check for weather conditions along the flight path, and determine landing options in case of emergency. The dispatcher compiles this information, along with other technical details, in a dispatch release report, which is then sent to air traffic control at least 30 to 60 minutes before flight departure (U.S. Code of Federal Regulations Title 14, 2022b). En-route responsibilities during the actual flight include continuous communication with pilots, regular checking for weather along the flight path, and monitoring any potential events that could jeopardize safety. In the event of a contingency, such as a medical emergency, the flight crew will turn to the dispatcher for guidance and decision-making.

Under high workload conditions, dispatchers may not have sufficient time to make detailed analysis and may resort to conservative approaches, which can have financial implications for airlines. For instance, when making fuel calculations, dispatchers may tend to use intentional overestimates and add extra fuel for safety, reducing the fuel efficiency of the flight. Even though en-route tasks may require less involvement compared to pre-flight planning, dealing with contingencies such as changing weather adds to the dispatcher workload and allows for less time for efficiency planning (L. Huang, Nneji, and Cummings, 2019). Flight dispatchers are crucial for managing unexpected delays and preventing them from propagating into upcoming flights (A. Abdelghany, K. Abdelghany, and Guzhva, 2024).

As noted by Tan et al. (2024b), workload management is crucial in aviation operations, with direct implications for efficiency, safety, and cost. Investments in advanced simulation-based training, stress management programs, and decision-support systems for aviation operations are crucial in improving the dispatcher workload balance and their ability to keep up with the dynamic environment.

Airline companies have their own flight dispatchers that work out of their operations center. In a given day, flights are assigned to workstations for planning and monitoring. Workstations are up to 24 hours long, and are covered by dispatchers who work in shifts, usually 8 to 12 hours long. In other words, a workstation may be covered by more than one dispatcher in order to ensure smooth handover between dispatchers as shown in Figure 2.1. The workstation is operational from 4:00 am to 10:00 pm and has 10 flights assigned, where a flight is shown by a horizontal line spanning departure to arrival times. The workstation is covered by two dispatchers with shifts of 8.5 and 10 hours, respectively and a 30-minute overlap for takeover. Being responsible for planning and monitoring multiple flights simultaneously, dispatchers are expected to multi-task, which stresses the role they play in maintaining operational efficiency and the level of workload they experience. Dispatchers

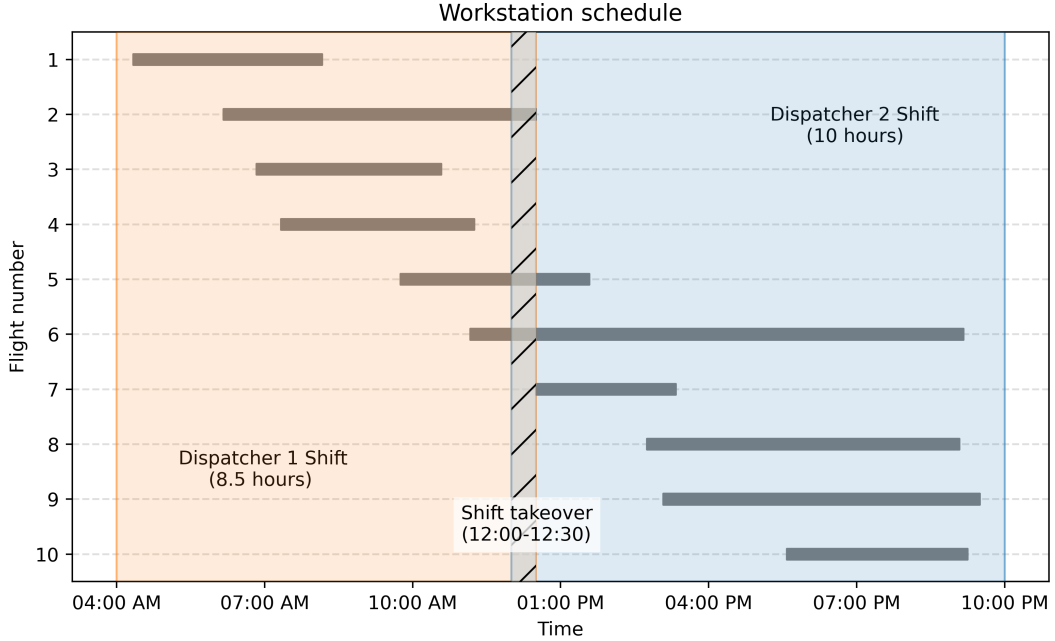


Figure 2.1: Illustration of an example workstation schedule with 2 dispatcher shifts.

operate under fluctuating levels of workload as flights change phases and unforeseen circumstances arise. Their workload depends on the set of flights assigned to their workstation at any given time and can have a significant impact on their performance and the cost and safety of flight operations. This is the motivation for the current work, where we study the allocation of flights to workstations with the purpose of balancing dynamic workload. This chapter specifically addresses the first primary research objective of this thesis: to develop and analyze foundational models for planning and balancing flight dispatcher workload.

Airline operations planning like flight scheduling and crew scheduling have substantial bodies of literature devoted to them and share mixed integer optimization models and path-based column generation solution frameworks (Barnhart and Cohn, 2004; Yildiz, Gzara, and Elhedhli, 2017; Zeighami and Soumis, 2019; D. Lu and Gzara, 2015). More comprehensive models involving integrated decisions, practical features or uncertainty/robustness may lead to larger models that are tackled with metaheuristics (see for example Y. Xu, Wandelt, and Sun (2021) and the review paper Y. Xu (2024)). In contrast, there is scarce dispatcher related work and the flight assignment problem does not have a path structure that is amenable to column generation like other airline problems due to the fact that flights are assigned to and processed at a workstation simultaneously. Another unique feature is the measure for workload both for individual tasks and for cumulative workload

assigned to a single dispatcher. There is very little research on dispatcher workload measurement. For the models developed and analyzed in this chapter, we introduce load as a unit-less quantity that captures the relative task load of flight planning and following as a function of flight distance. This measure allows the use of an additive function of task loads to calculate dispatcher workload. While it falls outside the scope of this work, the authors recognize that quantifying dispatcher workload is a research question by itself and is crucial to workload balancing. We note that the problem under study may seem to have similarity with other known problems which we explore in the literature review and show that the problem is unique and novel.

The primary contributions of the work presented in this chapter include the definition of a new class of problems, the solution methodology that can handle real-life problems, a test set and detailed testing. We define the flight dispatcher workload balancing problem that assigns flights to workstations so that the workload imbalance amongst the workstations over the operational time is minimized. We examine the problem under two measures for workload balance: peak workload and absolute deviation from the average workload. This problem fills a gap in the literature on a critical airline operations planning problem and establishes a new class of problems on task assignment to multiple agents with multi-tasking. We provide a mathematical programming formulation and use Lagrangian relaxation to decompose the problem into easier-to-solve subproblems and provide a lower bound. We prove that the Lagrangian lower bound has a closed-form expression, removing the need to iteratively search for it through subgradient and/or column generation type of methods. To find feasible solutions and guided by the Lagrangian lower bound, we develop a Focus-Search-and-Improve metaheuristic with a Genetic Algorithm core (FSI-GA) where restrictions of the full problem are explored and searched by a genetic algorithm, and solutions are further fine-tuned by an improvement heuristic. We performed extensive testing on 2019 U.S. historical domestic flight data from the Bureau of Transportation Statistics (Bureau of Transportation Statistics, 2023b) and analyzed solution quality, computational time, and workload balance metrics. For the peak workload objective, we found that the commercial solvers fail for problem sizes as small as 750 flights and 15 workstations. The Lagrangian lower bound is optimal in all instances with proven solution, which corresponds to 40% of instances, and the (FSI-GA) heuristic is on average 3.1% away from the Lagrangian bound. The peak workload objective achieves the goal of balancing the workload at times where peaks occur but does not directly balance the workload during other times. On the other hand, the absolute deviation objective achieves better workload balance, but it is more difficult to solve.

This chapter is organized as follows: In Section 2.1, the relevant literature on flight dispatching and workload balancing is reviewed. In Section 2.2, the problem setting and

mathematical models for two objectives are introduced. In Section 2.3, a Lagrangian relaxation is proposed and optimal Lagrange bound is derived for the peak workload objective. In Section 2.4, a genetic algorithm-based search heuristic is proposed. In Section 2.5, problem instances are constructed based on real world data, and numerical results are analyzed. Finally, in Section 2.6, conclusions and future research directions are provided.

2.1 Literature Review

Despite the rich literature on airline operations planning, we only found three studies that focus on flight dispatcher scheduling and assignment. This does not speak to the importance of the problem, but rather raises a question to whether the problem is entirely new or is part of a bigger problem class. We conducted the literature review with this in mind. We first focus on the three studies mentioned earlier, then we look at problem classes that have some similarity with dispatcher operations, and finally we review the literature on workload balancing measures. In our review, we found that flight dispatcher scheduling presents a new problem and its uniqueness stems from the multi-tasking and dynamic nature of the workload. This review will therefore examine existing approaches to identify gaps that this thesis aims to address.

The assignment of flight tasks to dispatchers is studied in Agnetis and Smriglio (2000), where each flight has only one task associated to it and a dispatcher is assigned one task at a time. Each task requires a known amount of time to process and a due date. Under these assumptions, the problem resembles a parallel machine scheduling problem with due dates where the goal is to find a feasible assignment. The paper further assumes that some flights are already assigned to dispatchers, which corresponds to unavailable machine times in the machine scheduling context, and that the task processing times take a given set of choices. The authors use a constraint satisfaction approach and explore solution properties as a function of processing times. C. Santos et al. (2011) study the assignment of flights to workstations under several goals and requirements. They attempt to capture the load requirement of a flight by dividing the workday into hourly intervals and defining the task requirement per hour as the time that needs to be allocated to the flight during that hour, e.g., a flight from 13:30 to 14:15 may have a 10-minute task in hour 13, and a 5-minute task in hour 14. The authors make an additional simplifying assumption that a flight may be assigned to different desks during different hours. The goals to be optimized include balancing workload which is measured as the hourly processing time assigned to a dispatcher, minimizing dispatcher over-utilization (i.e., being assigned more than 60 minutes of processing time), and minimizing the assignment of same flight tasks to different

workstations. Rayner (2021) considers a similar problem to the one studied in this thesis but focuses on the objective of minimizing the sum of peak workloads over all workstations. A Lagrangian relaxation and column generation were devised for the problem, along with construction and improvement heuristics. The construction heuristic assigns flights, one by one, to workstations with minimum workload at departure times. The improvement heuristic swaps flight assignments between workstations. Numerical testing is carried for flights of a European Airline and American Airlines, containing 50-434 flights and for 15, 20, and 25 flights per workstation.

These three studies provide valuable insights into dispatcher scheduling within the airline industry but fall short of modelling the realistic problem. Agnetis and Smriglio (2000) assume that a dispatcher deals with only one flight at a time, which is not true in practice. A dispatcher usually manages multiple flights, some in planning and others en-route, at any given time. Ignoring the multi-tasking feature of a dispatcher’s work leads to inaccurate assessment of their load. Furthermore, Agnetis and Smriglio (2000) do not take workload minimization into account as they aim to find a feasible solution. C. Santos et al. (2011) models dispatcher workload but resort to representing the task load requirement as time per hour, which fails to account for due dates and consequently, does not represent an accurate model. For example, a 10-minute task and two 5-minute tasks represent an equivalent workload, but if both 5-minute tasks must be finished earlier than the 10-minute task, say at the 7th and 10th minutes of the hour, not only they would represent a higher workload for their duration but also the dispatcher may not have enough time to finish them by the set deadline. Moreover, they split flight tasks between workstations, leading to flights being assigned to multiple dispatchers, which is not the case in practice.

Rayner (2021) focuses on workload balancing but considers the workload only at flight departure times. Moreover, minimizing the sum of peak workloads of workstations might not necessarily achieve a balance, as assigning all flights to the same workstation would also minimize this objective. Alongside flight dispatchers, air traffic control specialists (ATC) provide ground support for the flights. Since both ATC and flight dispatchers plan commercial air traffic for different flights, they share similar workload challenges related to multi-tasking and unforeseen changes.

ATC are government employees responsible for ensuring safety and order in air traffic. They manage flight operations at the airport and ensure that flight paths are followed and that there is sufficient separation. If necessary, they can make changes to the aircraft speed and altitude to manage the traffic (Ibáñez-Gijón et al., 2023). A study on the air traffic controllers’ mental load by Alharasees and Kale (2024) finds that the communication and workload are the most crucial factors affecting the operator’s efficiency. The study relates the operator workload to four characteristics: human behavior, skills, tacit knowledge, and

practice. There is a vast literature on ATC operations, studies include rostering (Pavlović, Jovanović, and Stanojević, 2023), capacity planning (Starita et al., 2020), modeling and predicting the workload (Ibáñez-Gijón et al., 2023; Alharasees and Kale, 2024), and traffic scheduling (Ikli et al., 2021; Mannino, Nakkerud, and Sartor, 2021). The nature of the work makes the critical questions for dispatchers and ATCs inherently distinct. While workload balance is a common goal, the mechanisms to achieve it are different. ATC workload is determined by the preset airspace sector and consequently the flights that use it. On the other hand, the workload of the flight dispatcher may be managed by optimizing the assignment of flights to workstations.

Unlike the flight dispatcher workload, the ATC workload is impacted by the airport and airspace they manage, which are not done for specific flights as in flight dispatchers, but are assigned ahead of time based on airspace. There is no literature, however, that tackles dynamic and time-varying load optimization based on airspace assigned, that would apply to flight dispatchers.

Dispatcher scheduling may resemble known problems already studied in the literature that consider workload balance such as parallel machine scheduling (Schwerdfeger and Walter, 2018). Parallel machine scheduling aims to assign jobs with known processing times to parallel machines. The load, as measured by the processing times, does not vary over time and the completion time is the objective often minimized (Ying, Pourhejazy, and X.-Y. Huang, 2024).

Another closely related problem setting is that of service personnel scheduling where multiple tasks are simultaneously handled by single personnel within a work shift. Nurses and doctors, for example, may encounter a similar workload structure as flight dispatchers (Legrain, Bouarab, and Lahrichi, 2015; Schoenfelder et al., 2020). Nurses are responsible for providing care to patients and monitoring their status, they have specialized training that designates the unit they can work in. Considering that they monitor multiple patients simultaneously, face unpredictable spikes in workload as emergencies arise, and are assigned to specific wards, their workload resembles that of flight dispatchers. The literature, however, does not model the dynamic nature of the workload over time. Instead, the cumulative workload is used. Patient-to-nurse ratio is a measure used for modeling the nurse workload, and schedules need to have this ratio to be as close as possible to a target value, which may be set by the government or hospital depending on the location. Anh (2022), for example, study the fair allocation of service tasks to healthcare workers by constraining the disparity in cumulative time it takes workers to finish their assigned tasks. The workload measure is reduced to the cumulative duration of all assigned tasks and does not explicitly model complexity nor load variations over time. Nurse scheduling may have additional complexities, for example, the study by Schoenfelder et al. (2020)

addresses the scheduling problem while also considering patient transfers between units. Other studies that considered static workload include Maenhout and Vanhoucke (2010), and Burke, J. Li, and Qu (2010).

These static models consider balancing the total amount of work assigned to nurses, as opposed to modeling a time-dependent workload and balancing it over time, which is the case in flight dispatchers. Moreover, nurse scheduling consists of various restrictions and features, such as patient transfers, unit and specialty restrictions, and assignment quotas, whereas the assignments of flight dispatchers do not have such constraints. As a result, these methods are unsuitable in the context of flight dispatchers.

The workload balancing problem of nurses and personnel scheduling is different than that of flight dispatchers in terms of how the workload is defined. The workload in the former is one dimensional and represents the total load over time, which aims to balance the cumulative load as opposed to over time. Flight dispatchers encounter varying levels of workload over time. As flights depart and land, they require different tasks and loads from the dispatcher based on their status.

To measure workload balance, the literature has suggested multiple metrics depending on the problem studied. The work of Stecke (1983) on flexible manufacturing systems forms the groundwork of workload balancing measures. They focus on balancing various aspects of the manufacturing process, such as the processing times of machines, the number of machines used to process each part, and tool magazine utilization rates. To measure the workload balance, they utilize functions such as (i) maximum workload, (ii) maximum pairwise difference in workload, and (iii) sum of squared pairwise differences in workload. They formulate these problems as mixed integer programming problems and linearize the measure (iii) using cross-product terms. Kumar and Shanker (2001) provide a detailed comparison of the effectiveness of different workload balancing objectives. Their results indicate that maximizing machine utilization balances the workload and makes most objectives ineffective. They also find that minimizing the maximum workload and the pairwise difference are the least and most effective measures for balancing workload, respectively.

However, it should be noted that these results are for the flexible manufacturing systems where the characteristics of a job are different from those of a flight in flight dispatcher scheduling. The difference is in the availability of jobs. In flexible manufacturing systems, a job may be processed at any time once the prerequisite processes are completed, which are also determined by the schedule. In flight dispatching, a flight has specific intervals of time during which the dispatcher must complete the related tasks. Vanheusden et al. (2022) address the workload balancing problem in warehouse order-picking operations and extend such metrics to multiple sets as: the sum of the peak workload in each zone, the

sum of variance in the workload of each zone over time, the sum of absolute deviations from the median workload in each zone over time, and the lexicographic peak workload in zones.

It is important to note that the workload balancing problem considered in this thesis differs fundamentally from classical scheduling problems. In traditional nurse scheduling, for instance, the objective is to balance the cumulative workload over an entire shift while accommodating constraints such as patient transfers, unit and specialty restrictions, and assignment quotas. In contrast, our problem addresses the dynamic nature of flight dispatcher operations where workload varies continuously over time. Moreover, although Rayner (2021) also studies workload balancing, their formulation considers workload only at flight departure times. In their objective function, assigning all flights to a single workstation would minimize the sum of peak workloads, a solution that is clearly not useful. By continuously monitoring workload throughout the operational period, our model avoids such trivial minimization and more realistically captures the complex, time-dependent nature of dispatcher workload.

The literature review revealed that current work on flight assignment and dispatcher scheduling makes limiting assumptions on load measures, on flight-to-dispatcher assignment, and on workload balancing. In this work, we model a more realistic setting that accounts for multi-tasking and introduces a load definition that is not time-based. This turns out to be, to the best of our knowledge, a new problem that has not been studied in literature. The identified limitations in current approaches and the unique characteristics of the dispatcher problem motivate the work presented in this chapter. In the following section, we formally define the problem under study, we detail the load definition that we adopt, and we present a nonlinear mathematical model along with a linearization.

2.2 The Dispatcher Workload Balancing Problem

The flight dispatcher workload balancing problems attempts to find an assignment of flights to workstations such that the workload is balanced among workstations. Each flight is assigned to exactly one workstation, and it is managed by the dispatcher working at the workstation at the time. Flights have scheduled departure and arrival times that dictate the time-varying load required by the assigned dispatcher. The load represents the cognitive and physical load that a flight requires from the dispatcher, which may depend on the flight path, its time, aircraft characteristics, and airports. The load varies over time as the flight changes status, and as a result, the workload of a dispatcher changes as the load requirements of their assigned flights changes. It should be noted that the terms

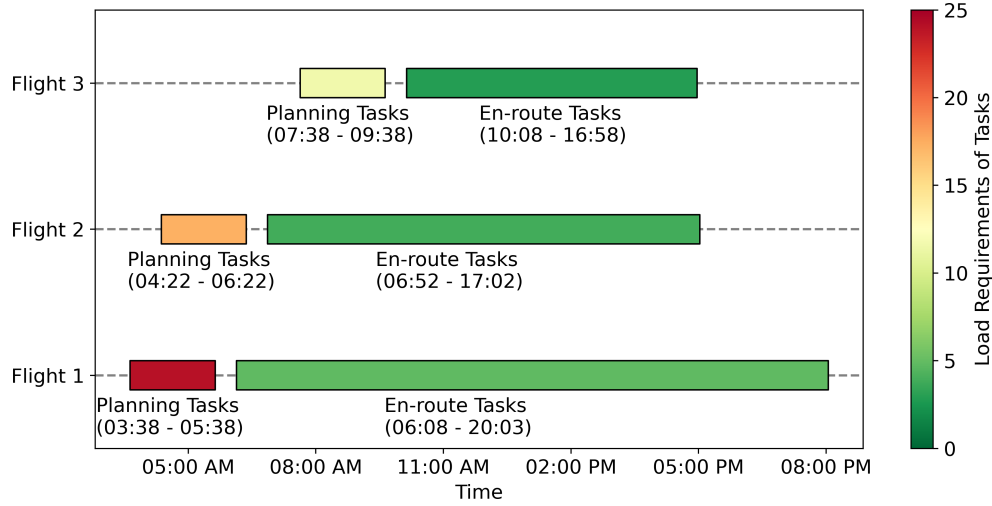


Figure 2.2: Variation of required load for three flights, departing at 6:08 am, 6:52 am, 10:08 am, respectively.

dispatcher and workstation are used interchangeably in this context, as a workstation has one dispatcher always working at it.

To illustrate these aspects on an example, let us consider 3 flights and 2 workstations. The flights depart at 6:08 am, 6:52 am, and 10:08 am, and land at 4:58 pm, 5:02 pm, and 8:03 pm, respectively. Planning for a flight starts 150 minutes before departure and lasts until 30 minutes before departure. Following the planning tasks, there is no task or load required from the dispatcher until the flight departs, after which the en-route tasks begin and last until the flight arrives at the destination airport. Figure 2.2 illustrates the distribution of load requirements of the three flights over time. Let us consider two possible flight-workstation assignments where flights are assigned to workstations (1, 1, 2) and (1, 2, 2), respectively. Under a workload model where workload of a workstation at any given time is equal to the total load requirement of its assigned flights at the given time, the workload distribution of the two solutions would vary over time as depicted in Figure 2.3. Although the question of whether solution 1 or 2 has a better-balanced workload depends on the balance measure, solution 2 is more likely to be preferred by most measures because of its smaller difference in workstation workloads over time.

In order to calculate workstation workload, a set of time-points is required. While defining this set in equal time intervals such as minutes or seconds may be accurate to capture workload variation over time, it is neither necessary nor efficient. It is not necessary since the variation in load over time is due to flights changing status, so it is sufficient to only

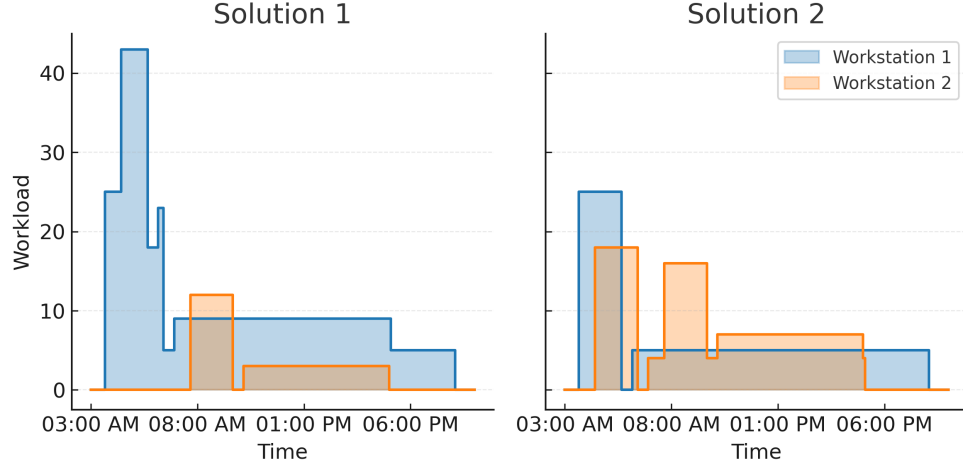


Figure 2.3: Comparison of workload in two different assignments where workstation assignments of flights are $(1, 1, 2)$ and $(1, 2, 2)$, respectively.

consider the set of time-points where at least one flight has a change in load requirement. Similarly, it is not efficient because if two time-points are identical in terms of flight load requirements, they will also have an identical workload, and the number of time-points can be reduced. Let us denote by T the set of time-points at which there is a change in the load requirement of any flight. For the example instance provided in Figure 2.2, T is $\{3:38, 4:22, 5:38, 6:08, 6:22, 6:52, 7:38, 9:38, 10:08 \text{ am}; 4:58, 5:02, 8:03 \text{ pm}\}$, i.e., whenever planning or en-route tasks of a flight start or end. Note that the cardinality of T is not necessarily a multiple of the number of flights since some time-points may be common to multiple flights and the elements of T are all distinct.

The distribution of flight load and workstation workload gets more complex as the number of flights increases. Figure 2.4 illustrates an example instance that consists of 50 flights and 2 workstations. Ideally, the load would be distributed equally among workstations resulting in equal workload for all. This is unlikely to happen in practice given the discrete and the time-varying nature of load and the fact that a flight load cannot be split between workstations. Hence, we aim to balance workload under two measures: minimizing the peak workload and minimizing total absolute deviation from the average workload over time. Formally, given a set of flights I , workstations K , discrete time-points T , and flight load requirement h_{it} of flight $i \in I$ at time $t \in T$, the Flight Dispatcher Peak Workload Balancing Problem (PWP) aims to find an assignment of flights to workstations that minimizes the maximum workload observed over $t \in T$ and $k \in K$. On the other

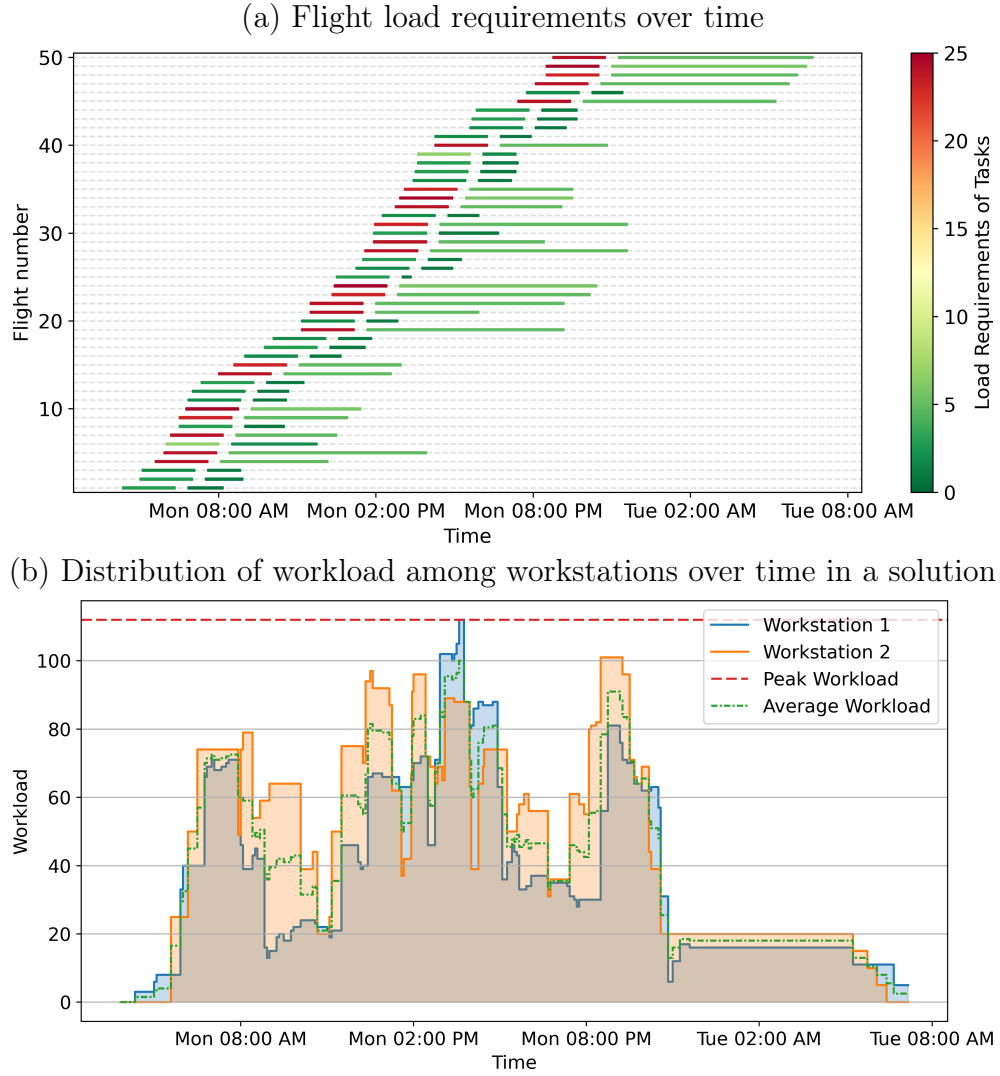


Figure 2.4: Flight load requirements and distribution of workload in a solution for an instance with 50 flights and 2 workstations.

hand, the Flight Dispatcher Absolute Workload Deviation Balancing Problem (ADP) aims to find an assignment that minimizes the total absolute deviation of workstation workloads from the average workload observed over $t \in T$.

The load of flight $i \in I$ at time $t \in T$, h_{it} , is unitless and reflects the relative load level among flights and tasks, e.g., if the planning task requires twice as much as the en-route task, the load associated with planning and en-route must reflect that same ratio. It is flight dependent as load may vary between flights, e.g., regional, international, as well as during different phases of the flight, such as planning and en-route. It is commonly said that a dispatcher has about 5 flights en-route for every flight they are planning at a given time (Collier and Bartels, 2021). The load may also depend on the dispatcher given their experience, which we do not consider in this research as we focus on the assignment of flights to workstations and not on dispatcher scheduling. The latter is in and of itself an important research question that needs to consider personnel scheduling aspects like seniority, preference, and availability. In calculating workstation workload using as the sum of assigned flight loads, we assume that flights do not share common characteristics that might affect their combined workload, which might not be the case in the real world. For example, when planning for two flights that have the same origin airport or that fly similar routes, the dispatcher may face a reduced load due to the similarity between the tasks. Incorporating workload functions that account for learning effects, dispatcher experience, and other dependencies requires rigorous research that is lacking in the current literature.

We denote the assignment of flight i to workstation k by a binary decision variable x_{ik} , which takes value 1 if the flight is assigned to the workstation and 0 otherwise. We use $f(\mathbf{x}, \mathbf{h})$ to represent a general function that measures the workload imbalance in a given assignment $\mathbf{x}$ based on the flight load requirements $\mathbf{h}$. The workload imbalance measure f produces a scalar value, which is minimized by the assignment with optimal workload balance. Given this notation, general mathematical model for the Flight Dispatcher Workload Balancing Problem is:

$$\min \quad f(\mathbf{x}, \mathbf{h}) \tag{2.1}$$

$$\text{s.t.} \quad \sum_{k \in K} x_{ik} = 1 \quad \forall i \in I \tag{2.2}$$

$$x_{ik} \in \{0, 1\} \quad \forall i \in I, k \in K. \tag{2.3}$$

The objective function (2.1) minimizes the workload balance measure function f , con-

straints (2.2) assign each flight to exactly one workstation, and constraints (2.3) set the domains of assignment variables to binary. The measure function f is the key factor affecting the difficulty of the problem, as it may comprise nonlinear terms.

When minimizing the peak workload,

$$f(\mathbf{x}, \mathbf{h}) = \max_{k \in K, t \in T} \sum_{i \in I} h_{it} x_{ik} \quad (2.4)$$

which is a nonlinear function and may be linearized by introducing decision variable $F^{\max}$ to represent the maximum workload observed across all workstations and over all time-points. The Flight Dispatcher Peak Workload Balancing Problem (PWP) is:

$$(PWP) : \quad \min \quad F^{\max} \quad (2.5)$$

$$\text{s.t.} \quad \sum_{k \in K} x_{ik} = 1 \quad \forall i \in I \quad (2.6)$$

$$\sum_{i \in I} h_{it} x_{ik} \leq F^{\max} \quad \forall t \in T, k \in K \quad (2.7)$$

$$x_{ik} \in \{0, 1\}, F^{\max} \geq 0 \quad \forall i \in I, k \in K. \quad (2.8)$$

The objective function (2.5) minimizes the peak workload $F^{\max}$, and constraints (2.7) set the peak workload $F^{\max}$ as the maximum over $t \in T$ and $k \in K$.

For the absolute deviation workload, let $\eta_t = \frac{\sum_{i \in I} h_{it}}{|K|}$ denote the average workload per workstation at time $t \in T$. The balance measure function takes the form:

$$f(\mathbf{x}, \mathbf{h}) = \sum_{k \in K} \sum_{t \in T} \left| \sum_{i \in I} h_{it} x_{ik} - \eta_t \right|, \quad (2.9)$$

which minimizes the total absolute deviation of workstation workloads from the average workload η_t over $t \in T$. The measure is nonlinear due to the absolute value function, which may be linearized introducing a decision variable v_{tk} that represents the absolute difference between the workload of workstation $k \in K$ and the average workload η_t at time point

$t \in T$. The Flight Dispatcher Absolute Workload Deviation Balancing Problem (ADP) is:

$$(ADP) : \quad \min \quad \sum_{k \in K} \sum_{t \in T} v_{tk} \quad (2.10)$$

$$\text{s.t.} \quad \sum_{k \in K} x_{ik} = 1 \quad \forall i \in I \quad (2.11)$$

$$\sum_{i \in I} h_{it} x_{ik} - \eta_t \leq v_{tk} \quad \forall k \in K, t \in T \quad (2.12)$$

$$- \sum_{i \in I} h_{it} x_{ik} + \eta_t \leq v_{tk} \quad \forall k \in K, t \in T \quad (2.13)$$

$$x_{ik} \in \{0, 1\}, v_{tk} \geq 0 \quad \forall i \in I, k \in K, t \in T. \quad (2.14)$$

The objective function (2.10) minimizes the total workload deviations over workstations and time-points. Constraints (2.12) and (2.13) set v_{tk} as absolute value of the difference between workstation $k \in K$ workload at time $t \in T$ and the average workload at time $t \in T$. Constraints (2.14) set the domains of the variables.

Even though we can linearize the nonlinear objectives for both (PWP) and (ADP), the resulting mixed integer programming problems are challenging to solve. As reported in Section 2.5.2, existing solvers fail to solve real world size instances within hours of computational time. The main challenges are the inherent nonlinear objectives and large scale. For example, there may be thousands of flights in each day, which could amount to hundreds of thousands of decision variables. We thus focus on developing solution methods that can solve real world instances quickly. In the following section, we propose a Lagrangian relaxation of (PWP) and derive closed form expressions of the Lagrangian bound.

2.3 Lagrangian Relaxation and Lower Bounding

In this section, we explore a Lagrangian relaxation (Geoffrion, 1974) on the Flight Dispatcher Peak Workload Balancing Problem, examine characteristics of the resulting subproblem and prove that the Lagrangian bound has a closed-form expression.

In an effort to solve the Flight Dispatcher Peak Workload Balancing Problem (PWP) efficiently, we propose to relax assignment constraints (2.6) with unrestricted Lagrangian

multiplier μ_i , $i \in I$. The resulting Lagrangian subproblem is:

$$(SP) : \quad z_{SP} = \min \quad F^{\max} - \sum_{i \in I} \sum_{k \in K} \mu_i x_{ik} \quad (2.15)$$

$$\text{s.t.} \quad \sum_{i \in I} h_{it} x_{ik} \leq F^{\max} \quad \forall t \in T, k \in K \quad (2.16)$$

$$F^{\max} \geq 0, x_{ik} \in \{0, 1\} \quad \forall i \in I, k \in K. \quad (2.17)$$

The Lagrangian lower bound $\mathcal{LR}_{MW}$ is the solution of the following Lagrangian dual problem:

$$(DP) \max_{\mu} z_{SP} + \sum_{i \in I} \mu_i, \quad (2.18)$$

which may be reformulated as a linear program, leading to the Lagrangian master problem:

$$(MP) : \quad z_{MP} = \max \quad \theta + \sum_{i \in I} \mu_i \quad (2.19)$$

$$\text{s.t.} \quad \theta + \sum_{i \in I, k \in K} x_{ik}^h \mu_i \leq F_h^{\max} \quad \forall h \in H \quad (2.20)$$

$$\theta, \mu_i \in \mathbb{R} \quad \forall i \in I. \quad (2.21)$$

where $h \in H$ represents the indices of solutions x_{ik}^h and $F_h^{\max}$ feasible to SP , and θ is the auxiliary variable for linearizing the master problem objective, representing the subproblem objective z_{SP} . The dual of (MP) is the Dantzig-Wolfe master problem:

$$(DW) : \quad \min \quad \sum_{h \in H} \alpha_h F_h^{\max} \quad (2.22)$$

$$\text{s.t.} \quad \sum_{h \in H} \alpha_h = 1 \quad (\text{dual variables: } \theta) \quad (2.23)$$

$$\sum_{h \in H} \alpha_h x_{ik}^h = 1 \quad \forall i \in I, k \in K \quad (\text{dual variables: } \mu_i) \quad (2.24)$$

$$\alpha_h \geq 0 \quad \forall h \in H. \quad (2.25)$$

Different approaches exist to solve the Lagrangian dual problem. Two of the most common algorithms are subgradient optimization and cutting plane/column generation methods. However, in this particular case, the Lagrangian dual problem DP may be

solved explicitly. In the remainder of this section, we explore the mathematical properties of the subproblem SP and the Lagrangian dual problem DP and provide a closed-form expression of its optimal solution.

2.3.1 Properties of the Lagrangian Subproblem

To study the properties of the subproblem for its effective solution, we first investigate the relationship between SP and its linear relaxation, which we call LSP . In particular, we show that SP does not have the integrality property, and the maximum workload is not necessarily the same in the optimal solutions of LSP and SP . Second, we prove an important property of the subproblem that greatly helps with its solution.

For the integrality property, let us consider an example with 4 flights $I = \{1, 2, 3, 4\}$, 2 workstations $K = \{1, 2\}$, three-time-points $T = [1, 2, 3]$, Lagrangian multipliers $\mu_i = [12, 9, 12, 9]$, and load matrix:

$$h_{it} = \begin{bmatrix} 0 & 0 & 16 \\ 0 & 24 & 24 \\ 16 & 0 & 0 \\ 24 & 24 & 0 \end{bmatrix} \quad (2.26)$$

The optimal solution of SP is $x_{ik} = 1$ for all $i \in I, k \in K$ with objective -36 and $F^{max} = 48$, whereas LSP has solution

$$x_{ik}^{LP*} = \begin{bmatrix} 1 & 1 \\ 0.67 & 0.67 \\ 1 & 1 \\ 0.67 & 0.67 \end{bmatrix} \quad (2.27)$$

with objective -40 and $F^{max} = 32$, different from that of SP . Therefore, the SP does not have the integrality property. Next, we show that the Lagrangian bound has a closed-form solution by first showing that the subproblem SP has an equivalent reduced problem in Proposition 1, we then use this result to derive the closed-form Lagrangian bound in Proposition 2.

Proposition 1 *The Lagrangian subproblem SP :*

1. *has an optimal solution x^* in which all workstations are assigned the same set of flights. i.e., $x_{ik_1}^* = x_{ik_2}^* \forall k_1, k_2 \in K$.*

2. reduces to an identical subproblem of this form:

$$(RSP) : \quad \min \quad F^{\max} - |K| \sum_{i \in I} \mu_i x_i \quad (2.28)$$

$$\text{s.t.} \quad \sum_{i \in I} h_{it} x_i \leq F^{\max} \quad \forall t \in T \quad (2.29)$$

$$F^{\max} \geq 0, x_i \in \{0, 1\} \quad \forall i \in I, \quad (2.30)$$

where $x_i, i \in I$, is a binary decision variable that takes value of 1 if flight is assigned and zero otherwise.

Proof: Let us write SP in the following equivalent form where $F_k^{\max}$ is the peak workload of workstation k :

$$\min \quad F^{\max} - \sum_{i \in I, k \in K} \mu_i x_{ik} \quad (2.31)$$

$$\text{s.t.} \quad \sum_{i \in I} h_{it} x_{ik} \leq F_k^{\max} \quad \forall t \in T, k \in K \quad (2.32)$$

$$F_k^{\max} \leq F^{\max} \quad k \in K \quad (2.33)$$

$$F^{\max}, F_k^{\max} \geq 0, x_{ik} \in \{0, 1\} \quad \forall i \in I, k \in K. \quad (2.34)$$

The proof seeks to find a lower bound and an upper bound, corresponding to a feasible solution, that coincide. For the lower bound, let us resort to the following Lagrangian relaxation where the second set of constraints are relaxed with Lagrange multipliers $\beta_k \geq 0, k \in K$, leading to

$$z_{SP}(\beta) = \min \quad \left(1 - \sum_{k \in K} \beta_k \right) F^{\max} + \sum_{k \in K} \beta_k F_k^{\max} - \sum_{i \in I, k \in K} \mu_i x_{ik} \quad (2.35)$$

$$\text{s.t.} \quad \sum_{i \in I} h_{it} x_{ik} \leq F_k^{\max} \quad \forall t \in T, k \in K \quad (2.36)$$

$$F^{\max}, F_k^{\max} \geq 0, x_{ik} \in \{0, 1\} \quad \forall i \in I, k \in K, \quad (2.37)$$

where β is the vector with items $\beta_k, k \in K$. For any $\beta_k \geq 0$, a valid lower bound is $z_{SP}(\beta)$.

We are particularly interested in $\beta_k^* = \frac{1}{|K|}$ and its corresponding subproblem:

$$z_{SP}(\beta)^* = \min \sum_{k \in K} \frac{1}{|K|} F_k^{\max} - \sum_{i \in I, k \in K} \mu_i x_{ik} \quad (2.38)$$

$$\text{s.t.} \quad \sum_{i \in I} h_{it} x_{ik} \leq F_k^{\max} \quad \forall t \in T, k \in K \quad (2.39)$$

$$F_k^{\max} \geq 0, x_{ik} \in \{0, 1\} \quad \forall i \in I, k \in K. \quad (2.40)$$

The latter is decomposable into $|K|$ identical subproblems:

$$(RSP) : \quad z_{RSP} = \min \quad \frac{1}{|K|} F^{\max} - \sum_{i \in I} \mu_i x_i \quad (2.41)$$

$$\text{s.t.} \quad \sum_{i \in I} h_{it} x_i \leq F^{\max} \quad \forall t \in T \quad (2.42)$$

$$F^{\max} \geq 0, x_i \in \{0, 1\} \quad \forall i \in I. \quad (2.43)$$

Let us denote the optimal solution of (RSP) as $F_*^{\max}$ and x_i^* , $i \in I$. The lower bound is then

$$|K| \left(\frac{1}{|K|} F_*^{\max} - \sum_{i \in I} \mu_i x_i^* \right) = F_*^{\max} - |K| \sum_{i \in I} \mu_i x_i^*. \quad (2.44)$$

Now let us build a feasible solution to SP , which we recall as:

$$\min \quad F^{\max} - \sum_{i \in I} \sum_{k \in K} \mu_i x_{ik} \quad (2.45)$$

$$\text{s.t.} \quad \sum_{i \in I} h_{it} x_{ik} \leq F^{\max} \quad \forall t \in T, k \in K \quad (2.46)$$

$$F^{\max} \geq 0, x_{ik} \in \{0, 1\} \quad \forall i \in I, k \in K. \quad (2.47)$$

Based on x_i^* , let $\bar{x}_{ik} = x_i^*$ $k \in K$ and $\bar{F}^{\max} = F_*^{\max}$, which is obviously feasible to SP . Its corresponding objective gives the following an upper bound:

$$\bar{F}^{\max} - \sum_{i \in I} \sum_{k \in K} \mu_i \bar{x}_{ik} = F_*^{\max} - |K| \sum_{i \in I} \mu_i x_i^* \quad (2.48)$$

As the lower bound 2.44 is the same as the upper bound 2.48, $(\bar{x}_{ik}, \bar{F}^{\max})$ is optimal to

SP . □

In fact, for any optimal solution of subproblem SP , every pair of workstations $k_1, k_2 \in K$ has identical flight assignments with respect to maximum workload $\max_{t \in T} \sum_{i \in I} h_{it} x_{ik}$ and total Lagrangian multiplier weight $\sum_{i \in I} \mu_i x_{ik}$. In other words, $\max_{t \in T} \sum_{i \in I} h_{it} x_{ik_1} = \max_{t \in T} \sum_{i \in I} h_{it} x_{ik_2}$ and $\sum_{i \in I} \mu_i x_{ik_1} = \sum_{i \in I} \mu_i x_{ik_2}$ for $k_1, k_2 \in K$.

To show this, let $\mathbf{x}^* = \{x_{ik} \in \{0, 1\} \forall i \in I, k \in K\}$ denote an optimal solution to the subproblem. The total penalty for workstation k , denoted by ν_k , is given by $\sum_{i \in I} \mu_i x_{ik}^*$. Furthermore, we define $\nu^{\max}$ as the maximum penalty across all workstations: $\nu^{\max} = \max_{k \in K} \nu_k$. Consider two subsets of workstations: A and B such that $A, B \subseteq K$. We define A as the set of workstations for which $F_k^{\max}$ is maximized. Similarly, B consists of workstations where ν_k is maximized. Hence,

$$F^{\max} = F_k^{\max}, \forall k \in A \text{ and } \nu^{\max} = \nu_k, \forall k \in B. \quad (2.49)$$

Let us establish that $\mathbf{x}^*$ is optimal if and only if sets A and B coincide. First, suppose $A \not\subseteq B$, meaning $A \setminus B \neq \emptyset$. For a workstation l in A but not in B , we have:

$$F_l^{\max} = F^{\max}, \quad \nu_l < \nu^{\max}. \quad (2.50)$$

For any workstation l' in B ,

$$\nu_l < \nu_{l'}, \quad F_l^{\max} \geq F_{l'}^{\max}. \quad (2.51)$$

Considering a modified solution where workstation l follows the assignments of workstation l' in $\mathbf{x}^*$ (with all other assignments remaining unchanged), the objective becomes:

$$\max_{k \in K \setminus \{l\}} F_k^{\max} - \left(\sum_{k \in K} \nu_k - \nu_l + \nu_{l'} \right). \quad (2.52)$$

This is strictly lower than the objective value of $\mathbf{x}^*$, indicating that workstation l does not appear in $\mathbf{x}^*$. This leads us to conclude $A \subseteq B$ for optimality. Second, for the reverse scenario, assume $B \not\subseteq A$ or $B \setminus A \neq \emptyset$. A workstation $l \in B \setminus A$ satisfies:

$$F_l^{\max} < F^{\max}, \quad \nu_l = \nu^{\max}. \quad (2.53)$$

If all workstations follow the assignments of l , the objective becomes:

$$F_l^{\max} - |K| \nu_l \quad (2.54)$$

which is strictly lower than $\mathbf{x}^*$. Thus, for optimality, $B \subseteq A$. Therefore, sets A and B are identical in an optimal solution.

Next, we show that A and B collectively include all workstations K in optimal solutions. Suppose l' belongs to both A and B . For a workstation l not in A or B , we have:

$$F_l^{\max} < F^{\max}, \quad \nu_l < \nu^{\max}. \quad (2.55)$$

A solution where l adopts the assignments of l' yields a reduced objective value:

$$F^{\max} - \sum_{k \in K} \nu_k > F^{\max} - \left(\sum_{k \in K} \nu_k - \nu_l + \nu_{l'} \right). \quad (2.56)$$

Therefore, all workstations must be in both A and B for optimality, implying that all workstations have equal peak workloads and incur identical penalties in optimal solutions. Proposition 1 implies that instead of solving SP which involves $|K||I|$ binary variables, we could just solve RSP with just $|I|$ binary variables. In that case, the Lagrangian master problem MP is reduced to:

$$(MP)z_{MP} = \max \quad \theta + \sum_{i \in I} \mu_i \quad (2.57)$$

$$\text{s.t.} \quad \theta + |K| \sum_{i \in I} x_i^h \mu_i \leq F_h^{\max} \quad \forall h \in H \quad (2.58)$$

$$\theta \in \mathbb{R}, \mu_i \in \mathbb{R} \quad \forall i \in I. \quad (2.59)$$

In the following section, we provide a closed form expression for the optimal Lagrangian bound.

2.3.2 The Optimal Lagrangian Bound

The reduced subproblem RSP provides the basis for deriving a closed form expression of the optimal Lagrangian lower bound. Proposition 2 shows that the master problem MP has a known optimal solution.

Proposition 2 *The optimal objective value of the Lagrangian master problem MP is given by:*

$$\max_{t \in T} \sum_{i \in I} \frac{h_{it}}{|K|} \quad (2.60)$$

Proof: First, we show that $\max_{t \in T} \sum_{i \in I} \frac{h_{it}}{|K|}$ is a lower bound for MP . Consider the dual solution given by $(\theta = 0, \mu_i = \frac{h_{is}}{|K|} \forall i \in I)$. Here, s corresponds to a time-point that maximizes the total load requirement of flights. Explicitly, s is:

$$s \in \operatorname{argmax}_{t \in T} \sum_{i \in I} h_{it}. \quad (2.61)$$

This solution $(\theta = 0, \mu_i = \frac{h_{is}}{|K|} \forall i \in I)$ satisfies constraints (2.58) since the left-hand side becomes equivalent to the workload at time s , which is always less than or equal to the peak workload $F_h^{\max}$:

$$(\theta = 0) + |K| \sum_{i \in I} x_i^h \left(\mu_i = \frac{h_{is}}{|K|} \right) = \sum_{i \in I} x_i^h h_{is} \leq F_h^{\max} = \max_{t \in T} \sum_{i \in I} h_{it} x_i^h. \quad (2.62)$$

Therefore, its objective, $\max_{t \in T} \sum_{i \in I} \frac{h_{it}}{|K|}$, is a lower bound. Next, we show that this value is also an upper bound for MP . Let us consider a relaxation of MP obtained by keeping constraints (2.58) for two feasible solutions of RSP . That is, we remove all constraints that correspond to feasible solutions other than these two. The first solution does not select any flights, $x_i = 0, i \in I$, whereas the second solution selects all flights: $x_i = 1, i \in I$. The corresponding relaxed master problem is:

$$\max \quad \theta + \sum_{i \in I} \mu_i \quad (2.63)$$

$$\text{s.t.} \quad \theta \leq 0 \quad (2.64)$$

$$\theta + |K| \sum_{i \in I} \mu_i \leq \max_{t \in T} \sum_{i \in I} h_{it} \quad (2.65)$$

$$\theta \in \mathbb{R}, \mu_i \in \mathbb{R} \quad \forall i \in I. \quad (2.66)$$

Let us assume that its solution is $\bar{\theta}, \bar{\mu}_i \ i \in I$. Being a relaxation of MP , the corresponding objective $\bar{\theta} + \sum_{i \in I} \bar{\mu}_i$ is an upper bound of MP . Multiplying the constraint (2.64) by $(|K| - 1)$ and adding the resulting inequality to the constraint (2.65) leads to the following condition on the objective function:

$$|K|\bar{\theta} + |K| \sum_{i \in I} \bar{\mu}_i \leq \max_{t \in T} \sum_{i \in I} h_{it} \Rightarrow \bar{\theta} + \sum_{i \in I} \bar{\mu}_i \leq \max_{t \in T} \sum_{i \in I} \frac{h_{it}}{|K|}, \quad (2.67)$$

which implies that $\max_{t \in T} \sum_{i \in I} \frac{h_{it}}{|K|}$ is an upper bound. Therefore, as this value is also

previously shown to be a lower bound, the Lagrangian dual problem MP has an optimal objective value of $\max_{t \in T} \sum_{i \in I} \frac{h_{it}}{|K|}$. For the problem instances where h_{it} is integer valued and x_{ik} are binary, the optimal solution is integer valued, and the Lagrangian bound can be rounded up to $\left\lceil \max_{t \in T} \sum_{i \in I} \frac{h_{it}}{|K|} \right\rceil$. $\square$

As a result, the optimal Lagrangian lower bound is calculated directly without the need to solve the Lagrangian dual iteratively. In the following section, we devise a heuristic that exploits the Lagrangian bound.

2.4 A Lagrangian Guided Focus-Search-Improve Heuristic

In practice, flight dispatcher operations often involve managing over thousands of flights per day. This leads to large-scale instances of the load balancing problem that need to be solved repeatedly and in short computational times. As a result, we develop a multi-layered genetic algorithm, which we refer as Focus-Search-Improve Genetic Algorithm (FSI-GA), that produces high quality solutions in very short times as shown in the numerical testing. The heuristic searches for solutions in three main phases: focus, search, and improve, which form a hierarchy in terms of how they affect the search for feasible solutions and the frequency in which they are applied. First, FSI-GA focuses the search by restricting the search space by considering only a subset of time-points $T' \in T$, thus reducing the computational cost of building and evaluating solutions. Second, a genetic algorithm first generates local solutions on the restricted space, then updates the pool of overall solutions by evaluating the workload balance over all time-points T . Third, solutions are further improved using a two-swap improvement heuristic. All three search phases are applied repeatedly by selectively augmenting T' and consequently expanding the search space. Genetic algorithm was chosen as the core solution exploration method mainly because almost all of its procedures can be parallelized to take advantage of multi-threading, which allows utilizing more of the available computational resources. Figure 2.5 gives an overview of the algorithm, and the three phases are explained next.

2.4.1 The Search Phase

The search phase consists of a Genetic Algorithm, which uses conventional settings and employs crossover-mutation and elitism in generating new solutions and terminates when

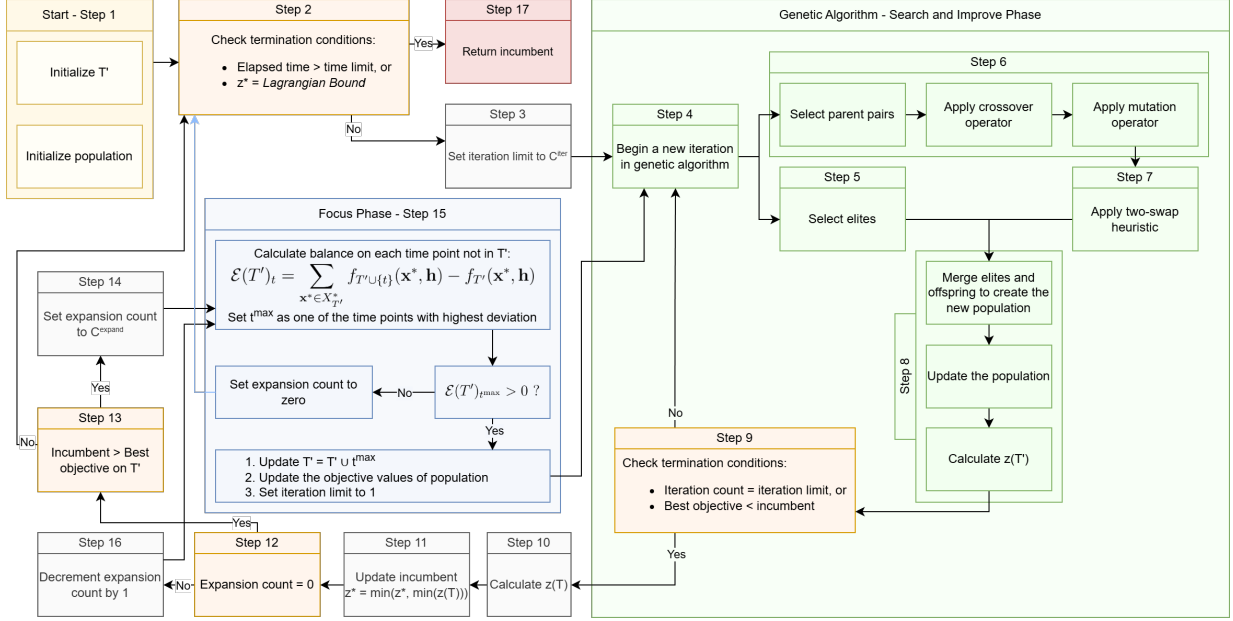


Figure 2.5: Overview of FSI-GA.

the incumbent equals the Lagrangian lower bound, or by time limit. Solutions are represented as integer vectors of size $|I|$ with values $\{1, 2, \dots, |K|\}$, indicating the workstation assignments of flights, e.g., if index i takes value k , flight i is assigned to workstation k in the generated solution. Example representations can be seen in Figures 2.6 and 2.7.

To initialize the genetic algorithm, a population of solutions is initialized randomly by generating vectors of size $|I|$ with each element taking a random value from $\{1, 2, \dots, |K|\}$, i.e., each solution is a random permutation of $\{1, 2, \dots, |K|\}$.

To generate a new population, the C^{elite} solutions with best objective values are selected as elites and are added to the new population without modification. In addition, a select set of parent pairs are recombined and mutated. Selection is based on an exponential rank selector that assigns a selection weight $w_r = \frac{c^r(c-1)}{c^P-1}$ to a solution with rank r , where the best solution has rank $r = 1$, c is the rate parameter, and P is the population size. Each pair of solutions undergoes crossover and mutation operations to generate two new solutions that are added to the population. The crossover operator uses a uniform crossover where each bit of an offspring is treated separately and chosen from either parent. Each of the two offspring has a designated parent from which they are more likely to inherit genes, this is controlled by the crossover probability p_c where the offspring inherits the gene from

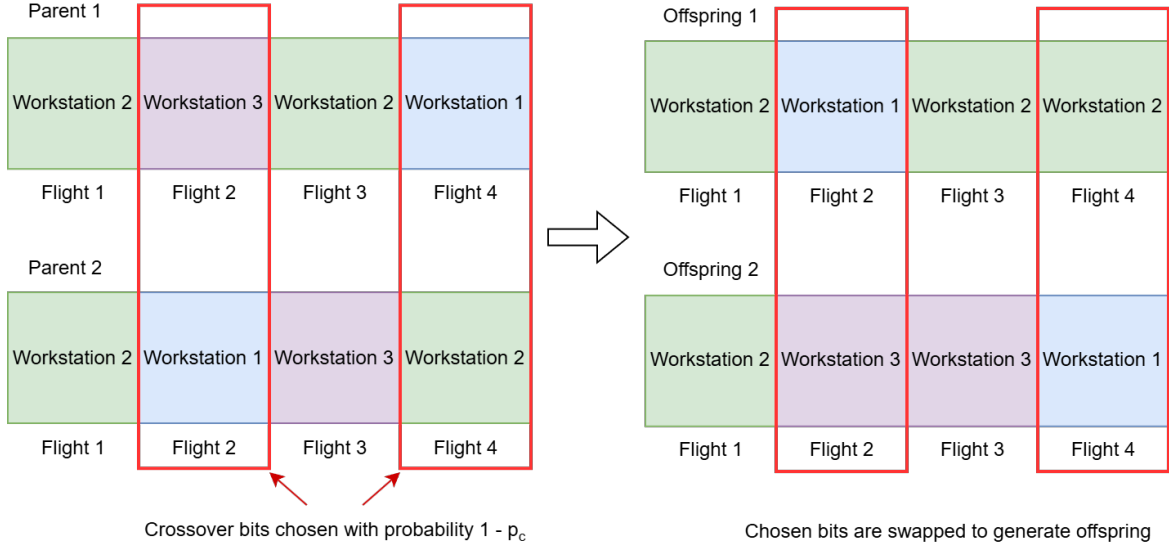


Figure 2.6: An example of uniform crossover operation on two solutions for 5 flights and 3 workstations.

their designated parent with probability $1 - p_c$ or from the other parent with probability p_c . An example crossover operation is illustrated in Figure 2.6. The mutation operator changes the bit to a different and randomly chosen value, which corresponds to changing the workstation assignment of a flight in the solution, Figure 2.7 illustrates an example mutation.

2.4.2 The Improvement Phase

The Two-Swap Improvement Heuristic modifies a given solution by considering swapping workstation assignments of two flights. The first flight is selected so that, if assigned to a different workstation, the balance objective may improve. For ADP, this could be any flight, whereas for PWP, it corresponds to flights that are assigned to workstations with highest peak workload and active at a time-point where the workstation workload is at the peak, i.e., from set $I^1 = \{i_1 \in I | \exists t \in T, k \in K : x_{i_1 k} = 1, h_{i_1 t} > 0, k \in \operatorname{argmax}_{l \in K} \sum_{j \in I} h_{j t} x_{j l}\}$. The second flight is selected such that, when swapped with the first flight, there is an improvement in the balance objective for their workstations over the times where either flight is active. For ADP, it means a decrease in the objective function value, while for PWP it means that the peak workload of the related workstations at times where either

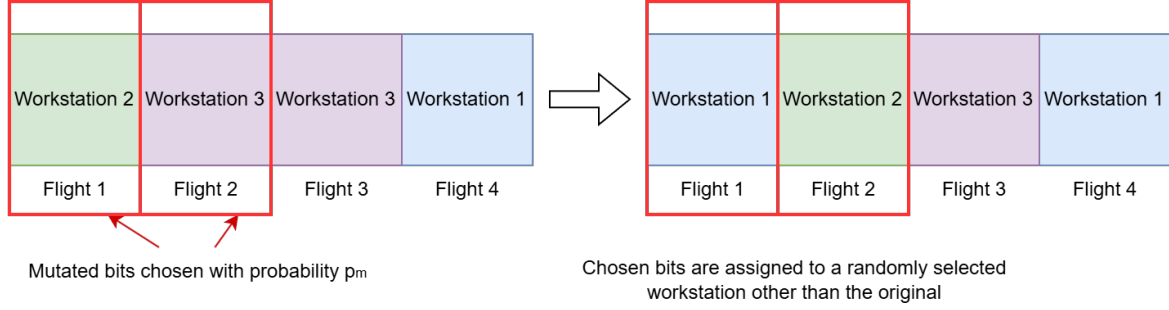


Figure 2.7: An example mutation operation on a solution for 5 flights and 3 workstations.

flight is active is less than the previous overall peak workload. Let $F_k = \max_{t \in T} \sum_{i \in I} h_{it} x_{ik}$ denote the peak workload of workstation $k \in K$, the set of such flights for a given flight $i_1 \in I^1$ can be expressed for PWP and ADP as:

$$(PWP) I^2(i_1) = \left\{ i_2 \in I \left| \begin{array}{l} x_{i_1 k_1} = 1, \quad x_{i_2 k_2} = 1, \\ \sum_{j \in I, j \neq i_1} h_{jt} x_{jk_1} + h_{i_2 t} < F_{k_1}, \\ \sum_{j \in I, j \neq i_2} h_{jt} x_{jk_2} + h_{i_1 t} < F_{k_2}, \\ \forall t \in T : h_{i_1 t} + h_{i_2 t} > 0, \quad \exists k_1, k_2 \in K \end{array} \right. \right\}, \quad (2.68)$$

$$(ADP) I^2(i_1) = \left\{ i_2 \in I \left| \begin{array}{l} x_{i_1 k_1} = 1, \quad x_{i_2 k_2} = 1, \\ \sum_{t \in T} \left| \sum_{j \in I, j \neq i_1} h_{jt} x_{jk_2} + h_{i_2 t} - \eta_t \right| < \sum_{t \in T} v_{k_1 t}, \\ \sum_{t \in T} \left| \sum_{j \in I, j \neq i_2} h_{jt} x_{jk_2} + h_{i_1 t} - \eta_t \right| < \sum_{t \in T} v_{k_2 t}, \\ \exists k_1, k_2 \in K \end{array} \right. \right\}. \quad (2.69)$$

The improvement heuristic searches the set $I^2(i_1)$ for $i_1 \in I^1$, and a swap is performed when an eligible flight $i_2 \in I^2(i_1)$ is found for $i_1 \in I^1$. The operation is applied recursively until no eligible swap can be found. For PWP, it should be noted that a swap does not

necessarily decrease the global peak workload, but it does guarantee a decrease in peak workload of the assigned workstations of flights i_1 and i_2 at times where either flight is active.

2.4.3 The Focus Phase

The focus procedure initialises the subset T' by ranking time-points in non-increasing order of total flight load requirements, calculated as $\sum_{i \in I} h_{it}$ for $t \in T$, and selecting the first τ time-points. This may not yield entirely accurate solutions since the objective does not account for the workload balance outside T' . To address this, the algorithm regularly evaluates the solution objectives for two different time sets within its solution pool, incrementally expanding T' with the time-points that have the largest impact on the objective. Meaning that the added time-point is the one that has the largest peak workload exceeding that of T' for PWP, or the one that has the largest total absolute deviation from the average workload for ADP.

The incremental extension of T' serves two purposes: it refines the solution accuracy as the algorithm progresses and avoids sets of time-points that have similar workloads. Workload similarity in this context refers to time-points in proximity (e.g., 15:01 and 15:02) where flights are at similar operational phases, requiring similar levels of dispatcher workload. Including time-points with similar workloads in T' may result in redundant computations.

An error metric, denoted as $\mathcal{E}(T')$, is introduced to update the set T' .

$$\mathcal{E}(T')_t = \sum_{\mathbf{x}^* \in X_{T'}^*} f_{T' \cup \{t\}}(\mathbf{x}^*, \mathbf{h}) - f_{T'}(\mathbf{x}^*, \mathbf{h}) \quad \forall t \in T, T' \subseteq T. \quad (2.70)$$

In the metric, $X_{T'}^*$ denotes the set of solutions on T' , and $f_T(\mathbf{x}, \mathbf{h})$ is the balance objective value for solution $\mathbf{x}$ and flight loads $\mathbf{h}$ calculated over set T . For a given time-point t that is not present in T' , metric calculates the error in using T' as an approximation of the workload balance compared to using $T' \cup \{t\}$.

In particular, for a given set of best solutions in the population X , and the best objective value in the population z^* , procedure selects one of the time-point that maximizes the

expressions (2.71) and (2.72) for PWP and ADP, respectively:

$$\operatorname{argmax}_{t \in T \setminus T'} \sum_{x \in X} \max\{0, \max_{k \in K} \sum_{i \in I} h_{it} x_{ik} - z^*\} \quad (2.71)$$

$$\operatorname{argmax}_{t \in T \setminus T'} \sum_{x \in X} \sum_{k \in K} \left| \sum_{i \in I} h_{it} x_{ik} - \eta_t \right| \quad (2.72)$$

A detailed procedure to update this subset of time-points is presented in Algorithm A.2.

2.5 Numerical Testing

In this section, we report on extensive numerical testing performed on (PWP) and (ADP) to assess the quality of the Lagrangian lower bound calculated using the closed form expression, and the performance of the FSI-GA heuristic relative to the Lagrangian bound and to the commercial solver Cplex. We also analyze the effectiveness of the minimum peak workload objective function in achieving the main goal of balancing workload among workstations and over time. All testing is performed on real data instances that we describe in Section 2.5.1.

All computations are made on a computer that has an Intel i7-4790 processor with a speed of 3.6 GHz and 8 GB of RAM. Mathematical formulations are solved using Cplex 20.1 implemented in Java 18.0.1. Node files, produced when solving the mixed integer program as the branch-and-bound tree is explored, are stored in the memory with a 3 GB limit. Additional files that exceed the limit are stored on a hard-disk drive. The heuristic methodology is implemented in the same Java version utilizes a multi-threading structure limited to 8 threads, same as the Cplex solver.

2.5.1 Flight Data and Workload Determination

We generate test instances from U.S. historical domestic flight data of the Bureau of Transportation Statistics (Bureau of Transportation Statistics, 2023b) for the year 2019. Since the workload balancing problem is a daily problem and is airline specific, we form instances by considering flights departing on the same day and operated by the same carrier on a single day, excluding cancelled flights. A flight is defined by the departing and arrival airports and times. As flights may span multiple time zones, we standardize local flight times

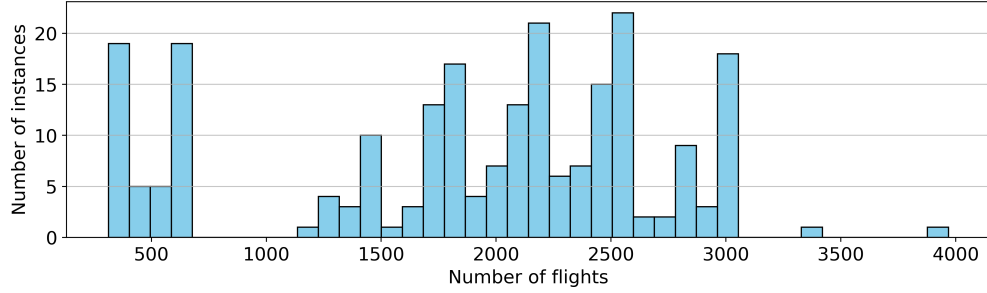


Figure 2.8: Distribution of the number of flights in test instances.

to Central Standard Time (CST, UTC -6). This ensures consistency when accounting for workload as a dispatcher center may oversee flights from several time zones. For example, a flight departing from Atlanta at 10am and another departing from Denver at 8am, in local times, would need their planning tasks done in parallel. We generate 231 instances that span multiple airlines and varying sizes in terms of number of flights. Table 2.1 and Figure 2.8 provides summary statistics of the instances.

Table 2.1: Flight summary statistics of the test instances, categorized by carriers.

Carrier	Instance count	Flights per instance			time-points per instance		
		Min.	Avg.	Max.	Min.	Avg.	Max.
Alaska Airlines	24	527	611	640	325	355	390
American Airlines	24	1,889	2,101	2,208	1,331	1,389	1,448
Delta Air Lines	83	1,706	2,276	2,627	1,151	1,262	1,330
Endeavor Air	24	313	378	431	591	651	731
SkyWest Airlines	22	1,415	1,774	2,363	1,091	1,166	1,249
Southwest Airlines	35	2,656	2,965	3,968	278	299	335
United Air Lines	19	1,161	1,416	1,870	1,107	1,176	1,334
Overall	231	313	1,873	3,968	278	955	1,448

There are relatively more instances from Delta Air Lines and Southwest Airlines as they are both among the top three carriers in the U.S. domestic passenger market in terms of market share (T4, 2022; Bureau of Transportation Statistics, 2023a), and have the largest problem instances, with the largest instance having 3,968 flights. The time-points refer to the set T defined in Section 2.2. The number of time-points per instance is on average about half the number of flights, which implies that usually multiple flights have a load change at some time-point. Both the number of flights and number of time-points, along with the number of workstations, determine the size of the optimization models solved in

terms of the number of decision variables and constraints. For an instance with $|I| = 1826$ flights, $|T| = 943$ time-points, and $|K| = 100$ workstations, problem (PWP) has 182,600 binary variables, and 96,126 constraints.

Determining flight load requirements is critical to the load balancing problem. As discussed earlier, it is commonly agreed that the load of a flight has two phases: planning and en-route. The planning phase includes tasks to get the flight plans ready, such as fuel planning and preparing the dispatch release. It starts 150 minutes before departure and lasts 120 minutes. There is no load in the 30-minute interval before departure as the flight prepares for takeoff. The en-route phase starts at takeoff and ends at landing and includes monitoring the flight and the weather and dealing with any alerts. To put numbers to the load associated with the tasks in each phase, we base the load on flight distance. The idea is that longer flights require more preparation before takeoff and has larger meteorological regions to check while in the air, which serves as an indication of the load required from the dispatcher. Moreover, weather information at the arrival airport is not as reliable on longer flights due to the larger time span, which presents a higher risk of contingencies that may require revising the flight plan. In rare case that may entail diversion to an alternate airport which would cause substantial workload on the dispatcher (Bartels, 2019; Collier and Bartels, 2021; Collier and Bartels, 2023). We acknowledge that an accurate numerical representation requires data from actual flight dispatcher operations. However, no such study has been done in the literature, and the subjective definition of what constitutes workload presents another research problem by itself.

We note that unforeseen events such as abrupt changes in weather conditions are not captured explicitly in the workload. This is due to the difficulty in determining the timing of such random events as we account for workload at time-points when workload changes. It is implicitly aggregated in the workload corresponding to en-route flight monitoring.

The planning load and the en-route load, respectively, are assigned $\left\lceil \frac{\text{Distance (mi)}}{250} \right\rceil$ and $\left\lceil \frac{\text{Distance (mi)}}{1250} \right\rceil$ units of load per minute, adding one planning and en-route load level for every 250 and 1250 miles flown. The reasoning for using these intervals is related to the average distance covered by flight between weather forecast updates, which roughly equates to 1250 miles considering landing and take-off. Figure 2.9 presents the distribution of load requirements over the 24-hour operational cycle. Figure 2.10 illustrates the distribution of flight planning and en-route loads per carrier over all testing instances.

The number of workstations in the operations centers of carriers is typically kept confidential and not shared with the public. According to statistics reported by Hope (2017),

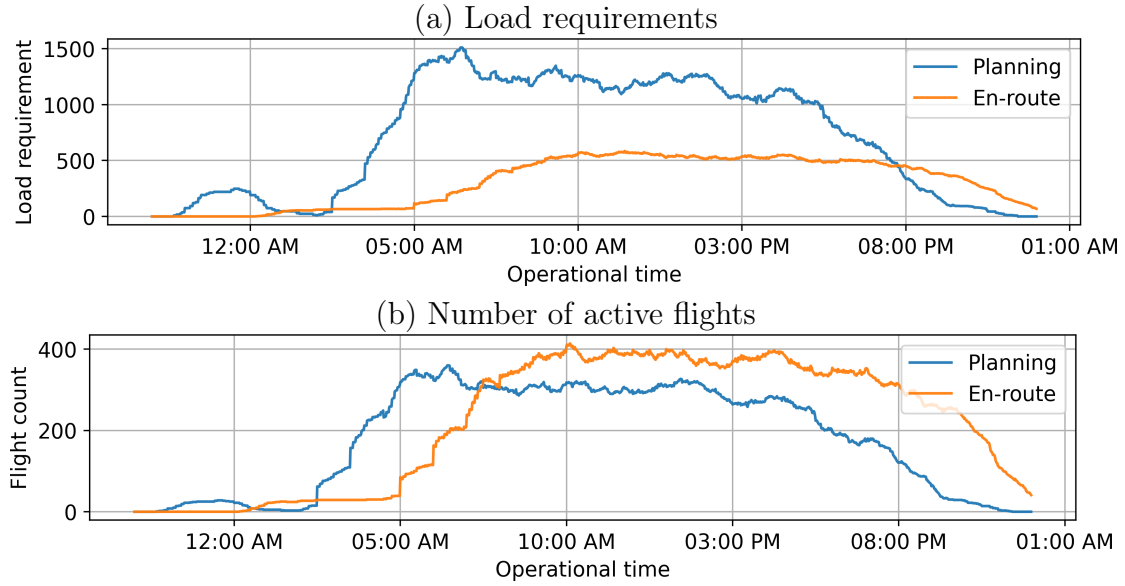


Figure 2.9: Planning and en-route phases over the operational hours in the problem instance of Delta Air Lines 4 Sep 2019.

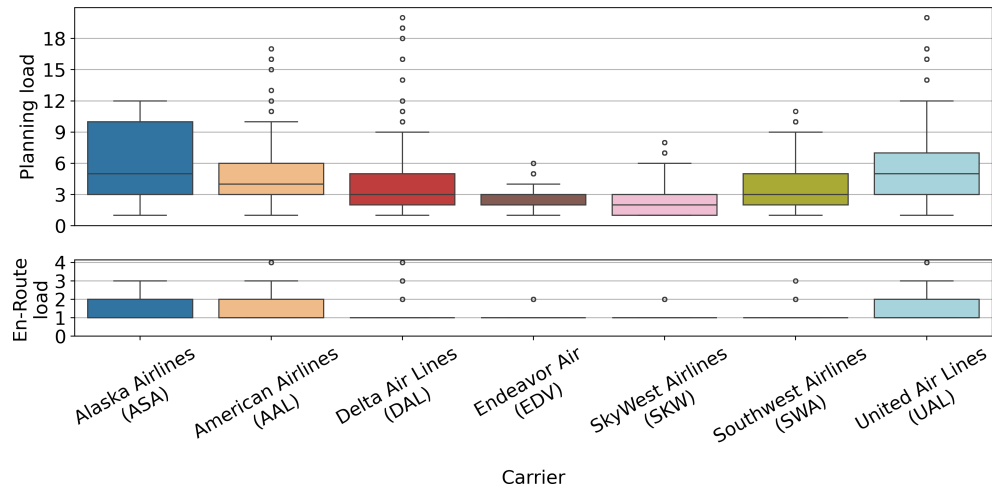


Figure 2.10: Distribution of planning and en-route load requirements of flights in the generated instances, categorized by carrier.

Air Canada utilizes 22 flight dispatchers working in 8-hour shifts, managing an average of 750 flights daily. With all workstations operating 24 hours and three dispatcher shifts distributed throughout the day, each workstation handles an average of 102 flights. For the sake of simplicity, we round this number to 100 flights per workstation and use it to determine the number of workstations as $\left\lceil \frac{|I|}{100} \right\rceil$. Additionally, we test a rate of 50 flights per workstation, and each instance is tested at ratios resulting in two classes of instances, R100 and R50 instances refer to an average of 100 and 50 flights per workstation, respectively. Note that we do not enforce that 100 (or 50) flights are assigned to each workstation, but rather we use it to set the number of workstations only.

The method of determining the number of workstations $|K|$ based on a flight-to-workstation ratio was informed by the Air Canada operational data and led to the R50 and R100 scenarios detailed previously. This approach was adopted for the research to ensure a practical and scalable means of defining this key parameter, allowing for consistent experimentation with the workload balancing models across diverse datasets and operational scales. Using a preset ratio is, however, a simplification of real-world practices.

2.5.2 Quality of Solutions

In this section, we assess the quality of the solutions generated by FSI-GA relative to the Lagrangian lower bound and to the performance of the commercial solver Cplex. The genetic algorithm is applied with a population size of 128, elite count of 16, mutation rate of $\frac{1.5}{|I|}$, crossover rate of $\frac{3.5}{|I|}$, initial time-point set size $\tau = 20 + \max \left\{ 4, \left\lceil \frac{|T|}{10} \right\rceil \right\}$, and a time limit of 90 seconds for the full heuristic. Cplex is applied with a time limit of 1 hour clock time (TiLim=3600).

Tables 2.2 and 2.3 report on the gaps between Cplex and FSI-GA and Lagrangian bound, the percentage of instances where either method outperforms the other, and on computational times for R100 and R50 instances. We report the results by averaging instances with the same number of workstations K . The Cplex MIP Gap is calculated as $C_Gap = 100 \times \frac{\text{Cplex Incumbent} - \text{Cplex lower bound}}{\text{Cplex Incumbent}}$. The gap between Cplex and FSI-GA is calculated as $HC_Gap = 100 \times \frac{\text{FSI-GA solution} - \text{Cplex Incumbent}}{\text{Cplex Incumbent}}$. Similarly, the gap between FSI-GA and the Lagrangian bound is calculated as $HL_Gap = 100 \times \frac{\text{FSI-GA solution} - z_{MP}}{z_{MP}}$. Columns C+ and H+, respectively, contain the proportion of instances where Cplex incumbent outperforms the FSI-GA solution and vice versa. Column HL_Gap contains the difference between FSI-GA objective and the Lagrangian bound, relative to the Lagrangian bound. Finally, column Solved under FSI-GA vs Lag contains the proportion of instances

where FSI-GA has reached the Lagrangian bound.

The Lagrangian bound is very tight as it is confirmed to be equal to the optimal objective value in all problem instances with known optimal solutions, which corresponds to 24.2% and 48.5% of the R50 and R100 instances. In the remaining instances the gaps are 6.7% and 3.1% for R50 and R100 respectively. It is only able to solve smaller instances with fewer flights and number of workstations. The average gaps are 44.3% and 5.6% and the average CPU times are 2,861s and 2,442s for R50 and R100. Extending the time limit to 3 hours was not sufficient for solver to obtain an optimal solution, it failed to improve the MIP gap for both R50 and R100 in two instances tested. When compared to the Lagrangian gap and Cplex, FSI-GA demonstrates better performance across different settings. It outperforms Cplex in solution quality and finds solutions that are very close to the Lagrangian bound in less than 100s. It surpasses the Cplex solver in 76.6% of R50 instances and 42.4% of R100 and obtains a better workload balance, outperforming the solver by an average of 41.9% and 4.1% in R50 and R100 instances, respectively.

For (ADP), the Cplex solver and FSI-GA are applied under the same settings used in the testing of (PWP). Due to time constraints and consistency of the results, (ADP) is tested on 10 of the problem instances with fewest flights. In tested instances, the Cplex solver failed to obtain a solution of comparable quality within the time limit and finished with a MIP gap of over 90%. On average, the heuristic objective outperforms the Cplex incumbent by 96% in R100 and 94% in R50.

Figure 2.11 illustrates the relation between the problem size and the algorithm time. As expected, increasing number of flights make the problem intractable more quickly in R50 instances than R100. This is evident as the algorithm has more instances terminating earlier than the set time limit. It is also seen that the heuristic still manages to obtain optimal solutions for instances with more than 1,000 flights within a short time.

Comparing R100 and R50, it is clear that Cplex performs much worse on R50 instances and on higher number of workstations and flights, the effects are not as pronounced on FSI-GA. time-points T is another factor that may impact the difficulty of the instances that is not covered in Tables 2.2 and 2.3. It should be noted that, in these tables, the $|K|$ column indicates the number of workstations, which is determined by the number of flights $|I|$ in the problem instance as $\lfloor \frac{|I|}{50} \rfloor$ for R50, and $\lfloor \frac{|I|}{100} \rfloor$ for R100. Therefore, as the number of flights in each instance differ, so does the number of instances for each $|K|$ value. In order to analyze the effects of time-points we further plot the gaps relative to the ratio of number of flights and time-points $\frac{|I|}{|T|}$ in Figure 2.12.

There seems to be smaller variability in the heuristic solutions (when compared to the Lagrangian bound) for higher flight to time-point ratios. A similar trend is apparent for

Table 2.2: Computational performance of the FSI-GA solution for (PWP) compared to the Cplex incumbent solution and Lagrangian lower bound for R50 instances.

Instance		FSI-GA vs Lag		FSI-GA vs Cplex		Cplex		Time (s)	
$ K $	Count	HL_Gap	Solved	HC_Gap	$C+,H+$	C_Gap	Solved	CTime ¹	HTime ²
6	6	0.0	100	0.0	-, -	0.0	100	1.3	0.2
7	13	0.5	84.6	0.5	15.4, -	0.0	100	10.8	15.1
8	5	0.0	100	0.0	-, -	0.0	100	4.0	1.6
10	1	0.0	100	0.0	-, -	0.0	100	5.0	1.0
11	6	0.0	100	0.0	-, -	0.0	100	9.0	1.8
12	17	0.0	100	0.0	-, -	0.0	100	15.5	2.8
23	1	3.3	-	-1.6	-, 100	4.8	-	3,601.0	91.0
25	4	2.1	-	-2.0	-, 100	4.0	-	3,600.8	91.0
26	1	0.0	100	-1.6	-, 100	1.6	-	3,601.0	67.0
27	2	2.7	-	-18.1	-, 100	20.1	-	3,601.5	91.0
28	7	3.4	14.3	-26.2	-, 100	28.3	-	3,601.3	88.6
29	3	4.3	-	-50.9	-, 100	52.7	-	3,602.0	91.3
31	1	7.4	-	0.0	-, -	6.9	-	3,601.0	91.0
33	6	4.2	-	-15.7	-, 83.3	19.2	-	3,605.2	91.7
34	7	4.6	-	-35.1	-, 100	38.0	-	3,610.4	91.0
35	8	5.0	12.5	-27.4	-, 62.5	30.7	-	3,608.4	87.1
36	11	4.6	-	-47.2	-, 90.9	49.7	-	3,607.1	91.1
37	3	4.7	33.3	-34.8	-, 100	37.1	-	3,604.3	86.0
38	2	7.9	-	-50.6	-, 100	53.8	-	3,606.5	91.0
39	1	2.0	-	-33.3	-, 100	35.7	-	3,607.0	91.0
40	6	6.5	-	-50.5	-, 100	53.1	-	3,607.5	91.0
41	9	9.7	-	-64.0	-, 100	66.9	-	3,609.0	91.0
42	5	10.0	-	-72.6	-, 100	75.2	-	3,605.8	91.4
43	11	7.0	-	-75.3	-, 100	76.4	-	3,610.4	91.0
44	10	6.8	-	-67.1	-, 100	69.5	-	3,610.3	91.0
45	5	8.5	-	-51.9	-, 100	56.1	-	3,612.4	91.2
47	7	11.1	-	-97.6	-, 100	97.9	-	3,619.1	91.0
48	4	8.0	-	-37.0	-, 100	42.4	-	3,616.8	91.0
49	9	9.7	-	-59.7	-, 100	63.7	-	3,616.0	91.1
50	13	9.1	-	-84.5	-, 100	85.9	-	3,617.1	91.0
51	11	9.4	-	-87.7	-, 100	89.0	-	3,614.8	91.0
52	1	17.9	-	-97.7	-, 100	98.1	-	3,604.0	92.0
53	1	2.6	-	-35.5	-, 100	37.1	-	3,602.0	91.0
54	1	2.6	-	-41.2	-, 100	43.2	-	3,603.0	91.0
55	3	2.5	33.3	-34.1	-, 100	35.6	-	3,601.3	87.7
56	5	1.6	40.0	-40.2	-, 100	45.1	-	3,603.2	82.2
57	3	1.7	33.3	-44.9	-, 100	48.3	-	3,604.3	79.7
58	2	2.6	-	-36.7	-, 100	40.8	-	3,603.5	91.0
59	4	4.1	-	-40.5	-, 100	49.1	-	3,605.0	91.5
60	14	3.7	-	-44.0	-, 100	49.5	-	3,605.3	91.6
67	1	9.4	-	-47.0	-, 100	52.8	-	3,602.0	91.0
79	1	13.3	-	-48.5	-, 100	55.6	-	3,604.0	91.0
Overall	231	5.1	23.4	-41.9	0.9, 76.6	44.3	20.8	2,861.1	72.6

1: Heuristic runtime in seconds

2: Solver runtime in seconds (clock time)

Table 2.3: Computational performance of the FSI-GA solution for (PWP) compared to the Cplex incumbent solution and Lagrangian lower bound for R100 instances.

Instance		FSI-GA vs Lag		FSI-GA vs Cplex		Cplex		Time (s)	
$ K $	Count	HL_Gap	Solved	HC_Gap	C+,H+	C_Gap	Solved	CTime ¹	HTime ²
3	19	0.0	100	0.0	−,−	0.0	100	0.4	0.0
4	5	0.0	100	0.0	−,−	0.0	100	1.0	0.0
5	7	0.0	100	0.0	−,−	0.0	100	1.0	0.6
6	17	0.0	100	0.0	−,−	0.0	100	1.2	0.9
11	1	0.8	−	0.8	100,−	0.0	100	1,261.0	91.0
12	4	0.6	25.0	0.2	25.0,−	0.4	50.0	2,347.0	78.2
13	3	0.0	100	0.0	−,−	0.0	100	376.0	49.3
14	10	0.4	70.0	-0.4	−,30.0	0.7	60.0	2,005.4	78.0
15	1	0.0	100	-1.8	−,100	1.8	−	3,601.0	76.0
16	6	0.9	50.0	0.6	33.3,−	0.3	83.3	1,894.7	78.7
17	15	0.6	66.7	-0.9	6.7,20.0	1.5	60.0	2,112.3	74.5
18	14	0.9	42.9	-1.0	7.1,64.3	1.9	21.4	3,143.4	81.9
19	3	2.1	−	-0.3	33.3,66.7	2.4	33.3	2,575.7	91.3
20	15	2.8	20.0	-0.0	33.3,26.7	2.7	20.0	3,108.3	87.1
21	16	3.4	6.2	-0.1	25.0,37.5	3.4	12.5	3,272.6	91.2
22	15	1.9	13.3	-3.2	6.7,86.7	5.0	−	3,608.5	90.5
23	7	4.7	−	-0.9	−,57.1	5.4	−	3,604.1	91.4
24	13	3.4	−	-0.3	15.4,38.5	3.6	7.7	3,386.5	91.1
25	24	4.6	−	-21.4	16.7,54.2	24.6	4.2	3,528.0	91.0
26	2	3.9	50.0	-47.9	−,50.0	48.1	50.0	2,170.0	68.0
27	4	0.3	75.0	-3.3	−,100	3.6	−	3,600.8	66.2
28	8	0.2	87.5	-5.0	−,100	5.2	−	3,602.8	75.4
29	6	0.7	66.7	-3.5	−,100	4.2	−	3,601.2	77.5
30	14	2.1	−	-6.7	−,100	8.6	−	3,601.7	91.3
33	1	6.2	−	-34.0	−,100	37.9	−	3,602.0	91.0
39	1	6.7	−	-36.0	−,100	40.7	−	3,602.0	92.0
Overall	231	1.8	43.3	-4.1	10.0,42.4	5.6	37.2	2,442.0	67.1

1: Heuristic runtime in seconds

2: Solver runtime in seconds (clock time)

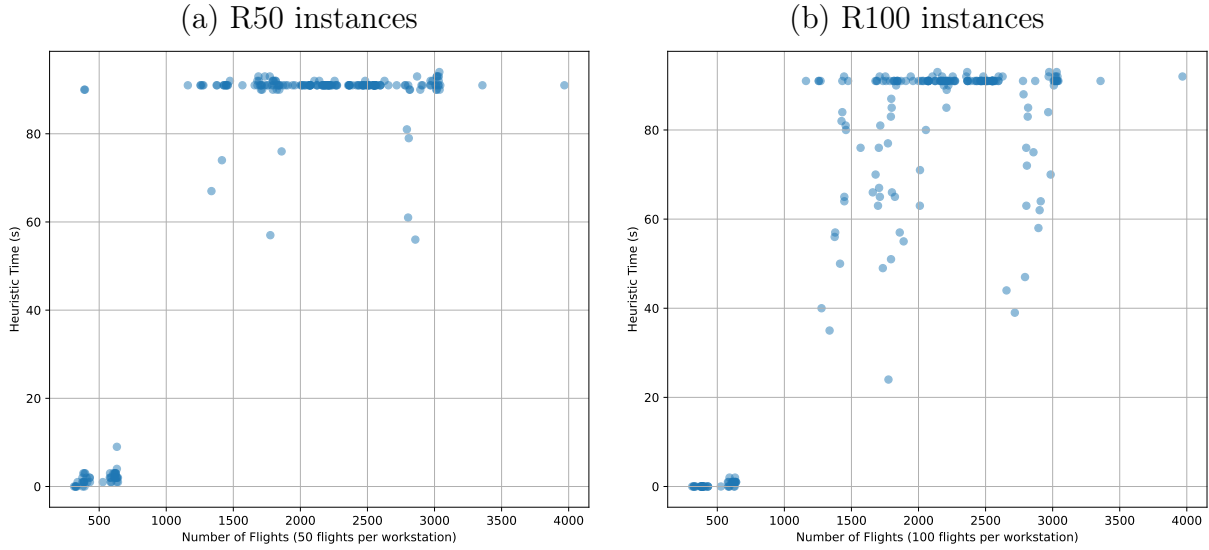


Figure 2.11: Heuristic time and instance size.

the Cplex gap. It is worth noting the jump in the ratio from 3 to 9. It turns out that SouthWest Airlines instances have a much higher flight to time-point ratio in the range of 9 to 11. had this ratio ranges between 0.5 to 2.5 for the other carriers in the testing. This is probably due to the flight schedules operated by Southwest Airlines, where a high number of flights have the same departure/arrival times. In fact, the flight to time-point ratio seems to be a characteristic of the carrier, as illustrated in Figure 2.13 which plots the distribution of flight to time-point ratios for each carrier. Endeavor airlines has lower ratios implying that few flights share the same time-points.

2.5.3 Quality of the Workload Balance

The quality of workload balance achieved in the solution is influenced by the workload balance measure adopted. When minimizing the peak workload in (PWP), the objective value is set at time-points with high load requirements. This focus may lead to imbalance at other time-points, the absolute deviation measure in (ADP) addresses this imbalance as it minimizes the deviation from the average. For instance, Figure 2.14 displays the distribution of total load requirements in one instance corresponding to Delta Air Lines, September 7, 2019. Figures 2.15 and 2.16 show, respectively, the distribution of workload

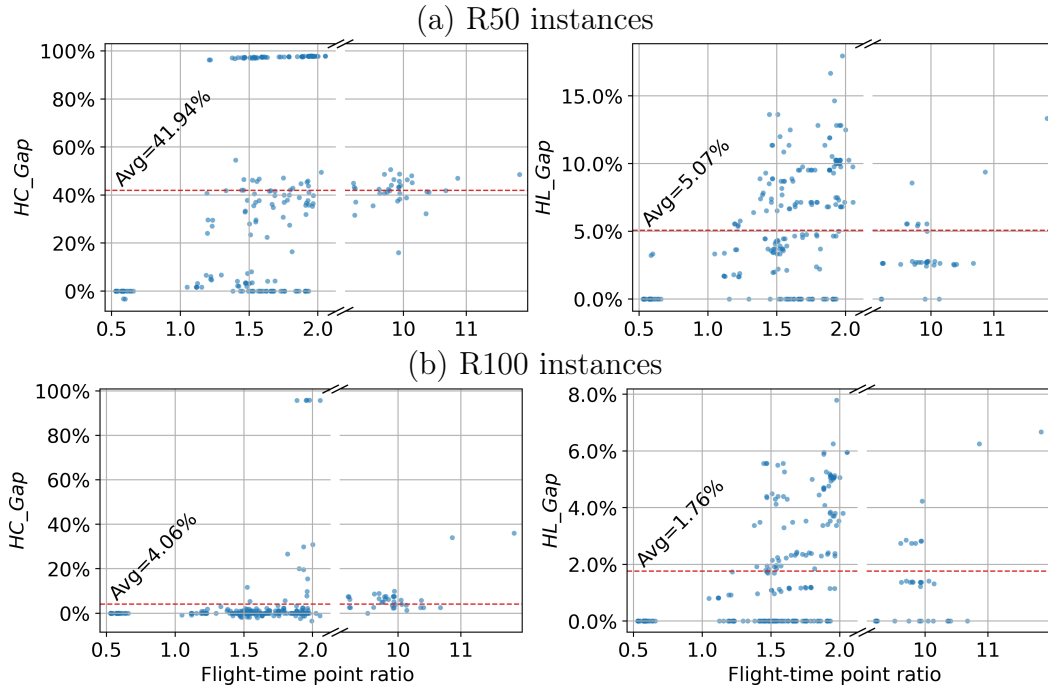


Figure 2.12: Summary of the FSI-GA performance over varying flight to time-point ratio $\frac{|I|}{|T|}$.

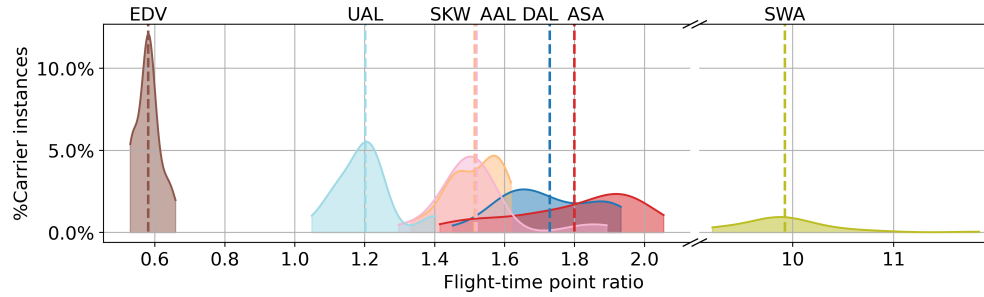


Figure 2.13: Distribution of flight to time-point ratio among instances of each carrier.

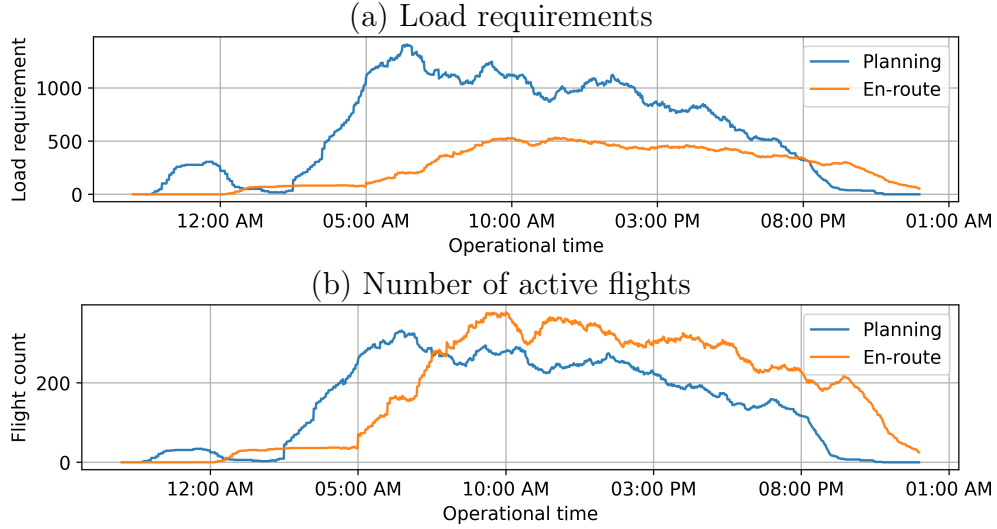


Figure 2.14: Planning and en-route phases over the operational hours in the problem instance of Delta Air Lines 7 Sep 2019.

among workstations in a (PWP) and an (ADP) solution with R100. The blue dashed line shows the Lagrangian lower bound, which is equal to the peak workload in this case. The red, cyan and green lines show the maximum, minimum and average workloads over all workstations throughout the day. These do not correspond to any single workstation but rather show the range over all workstations. The orange lines show individual workstation workloads. In a balanced workload, it is desirable to have all these lines in proximity. This observation underscores the limitations of the maximum workload balance measure and suggests that while it is effective in balancing workloads at peak times, it may not provide a uniformly balanced workload throughout the day. The absolute deviation measure can obtain a more balanced workload distribution, potentially with a higher overall peak workload. This observation is supported by the results reported in Tables 2.4 and 2.5, which provide the comparative statistics of the peak workload and absolute deviation solutions over all instances. All the stats are calculated as the ratio of the value for the (PWP) solution, and that of the (ADP) solution. The (PWP) always results in smaller peak workload with a ratio of 0.83 on average. This comes at the expense of doubling the range on average compared to the (ADP) solution. Similarly, the (ADP) solutions lead to better workload balance range not only in the objective value but also in the maximum, average, and standard deviation.

Workload balance throughout the day is the key factor in assessing the quality of so-

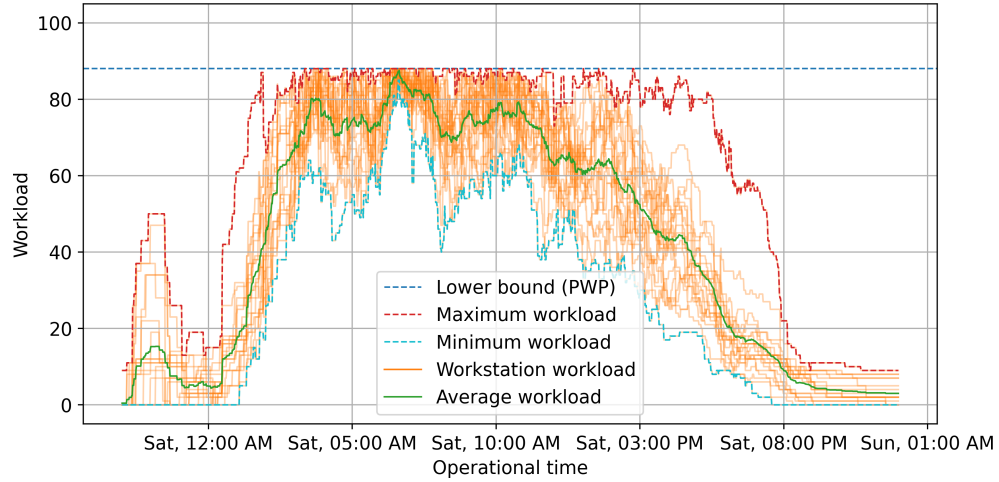


Figure 2.15: Distribution of workload in the optimal solution for (PWP) of R100 Delta Air Lines 7 Sep 2019 instance.

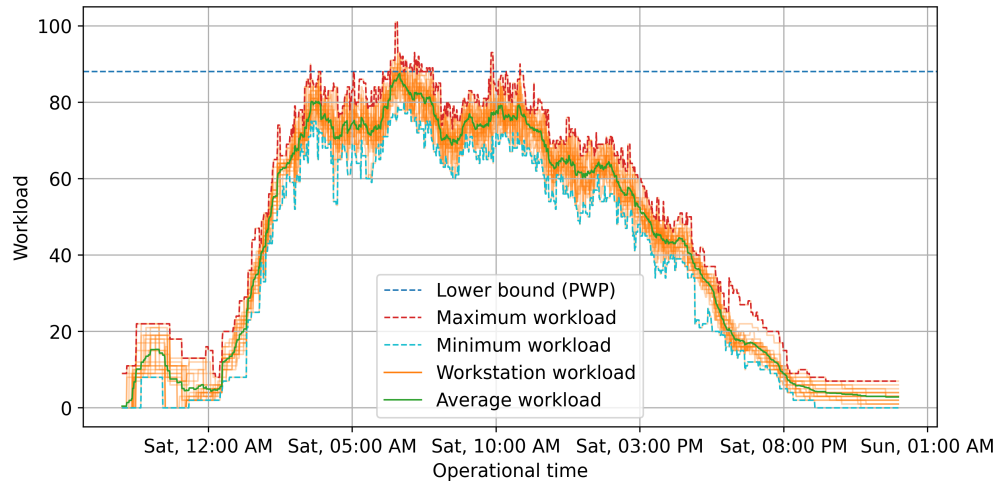


Figure 2.16: Distribution of workload in the heuristic solution for (ADP) of Delta Air Lines 7 Sep 2019 instance.

Table 2.4: Comparison of (PWP) and (ADP) solutions for R50.

Instance		Quality indicator ratio of (PWP) / (ADP)				
K	Count	Max. range	Avg. range	St.dev. range	Peak workload	Abs. workload dev.
6	6	2.47	2.68	2.68	0.93	2.76
7	13	2.43	2.37	2.65	0.92	2.57
8	5	2.26	2.35	2.77	0.90	2.55
10	1	1.72	2.28	2.61	0.81	2.50
11	6	1.87	1.97	2.10	0.86	2.12
12	17	1.66	1.96	1.98	0.83	2.10
23	1	1.86	1.62	2.02	0.89	1.91
25	4	1.90	1.66	2.27	0.86	1.97
26	1	1.88	1.74	2.42	0.82	2.04
27	2	1.91	1.52	2.23	0.85	1.85
28	7	1.69	1.56	2.19	0.88	1.90
29	3	1.68	1.45	2.11	0.85	1.84
31	1	1.87	1.80	2.34	0.85	2.09
33	6	1.71	1.76	2.13	0.87	2.13
34	7	1.70	1.73	2.18	0.85	2.09
35	8	1.64	1.70	2.04	0.84	2.04
36	11	1.65	1.60	2.08	0.82	1.92
37	3	2.14	2.33	2.59	0.81	4.95
38	2	1.64	1.60	2.01	0.80	1.89
39	1	1.72	1.63	2.69	0.81	1.93
40	6	1.45	1.50	1.79	0.84	1.87
41	9	1.47	1.51	1.89	0.84	1.86
42	5	1.52	1.56	2.02	0.86	1.89
43	11	1.55	1.55	1.92	0.83	1.88
44	10	1.53	1.46	1.87	0.83	1.86
45	5	1.35	1.45	1.63	0.83	1.84
47	7	1.37	1.39	1.82	0.82	1.70
48	4	1.32	1.34	1.54	0.81	1.74
49	9	1.45	1.40	1.68	0.83	1.78
50	13	1.23	1.28	1.43	0.82	1.64
51	11	1.36	1.27	1.54	0.81	1.61
52	1	1.50	1.31	1.80	0.92	1.58
53	1	1.54	1.35	1.58	0.82	1.63
54	1	1.39	1.38	1.58	0.83	1.67
55	3	1.39	1.44	1.57	0.77	1.69
56	5	1.43	1.39	1.54	0.80	1.62
57	3	1.34	1.29	1.67	0.63	2.17
58	2	1.52	1.38	1.65	0.82	1.63
59	4	1.44	1.30	1.40	0.77	1.54
60	14	1.34	1.27	1.43	0.77	1.51
67	1	1.17	1.16	1.25	0.76	1.47
79	1	1.17	1.09	1.12	0.77	1.37
Overall	231	1.62	1.60	1.95	0.83	1.97

Table 2.5: Comparison of (PWP) and (ADP) solutions for R100.

Instance		Quality indicator ratio of (PWP) / (ADP)				
K	Count	Max. range	Avg. range	St.dev. range	Peak workload	Abs. workload dev.
3	19	3.25	3.46	3.56	0.97	3.53
4	5	2.61	3.10	3.39	0.96	3.21
5	7	2.32	2.80	2.73	0.94	2.97
6	17	2.19	2.58	2.57	0.93	2.73
11	1	3.61	2.72	3.90	0.88	2.90
12	4	3.01	2.58	3.47	0.93	2.68
13	3	2.87	2.40	3.38	0.90	2.59
14	10	2.99	2.41	3.72	1.05	2.51
15	1	2.65	2.72	3.71	0.90	2.67
16	6	2.82	2.64	3.47	0.91	2.84
17	15	2.92	2.62	3.82	0.97	2.76
18	14	2.55	2.51	3.53	0.90	2.79
19	3	2.38	2.32	3.47	0.92	2.62
20	15	2.46	2.22	3.35	0.90	2.48
21	16	2.40	2.24	3.38	0.90	2.50
22	15	2.37	2.11	3.13	0.90	2.47
23	7	2.50	2.02	3.57	0.90	2.26
24	13	2.17	1.96	2.91	0.90	2.32
25	24	2.22	1.83	2.74	0.91	2.14
26	2	2.08	1.95	2.63	0.91	2.11
27	4	2.21	1.90	2.39	0.84	2.09
28	8	2.26	1.89	2.43	0.86	2.08
29	6	2.18	1.82	2.38	0.85	1.98
30	14	2.06	1.68	2.23	0.85	1.84
33	1	1.73	1.55	2.08	0.82	1.71
39	1	1.97	1.47	1.98	0.89	1.61
Overall	231	2.49	2.29	3.07	0.91	2.48

lutions. Neither the existing literature nor the official guidelines provide established standards for workload balance among flight dispatchers. However, even if such a standard were established, it would likely not be strict, as some fluctuations in workload are inevitable due to the dynamic nature of flight operations, with varying departure and arrival times. As such, these fluctuations do not necessarily indicate imbalance unless they are present for extended periods.

We further analyze the balance in terms of number of flights and workload per workstation over all instances. Since these two statistics are variable over time, we plot the minimum, maximum and average distributions for each carrier using box plots in Figures 2.17 and 2.18 for (PWP), and Figures 2.19 and 2.20 for (ADP). Looking at the average flight box plots (Figures 2.17 and 2.19 in orange), it is noticeable that there is little variation in the average number of flights among workstations. This is not surprising as the average is a characteristic of the instance and not the solution. These box plots do tell us, however, that each airline operates similar daily schedules. The maximum and minimum statistics are an outcome of the solution and their proximity to the average may indicate

whether the workstations in the same instance have similar number of flights over time. Except of Endeavor, all other solutions show some variability that we further analyze based on workload.

Similar to flights, the average workload is independent of the solution, as it is a function of the total load requirements of the flights and their schedule. On the other hand, high deviations of the minimum and maximum workloads from the average indicate that some workstations are being overloaded relative to others. R50 instances show more variability, which may be explained by the fact that there are about double the number of workstations compared to R100. The range in workload also depends on the magnitude of load. For example, on average, Alaska Airlines flights fly longer distances compared to others, which results in its workload values to be higher since the load is based on distance. Hence, higher workload from one airline to another is not an indicator of workload imbalance. It does highlight the limitation of how we set the number of workstations based on a preset ratio. While this may not be realistic for all operations, it does not limit the applicability of our models and methodology since the number of workstations is an input.

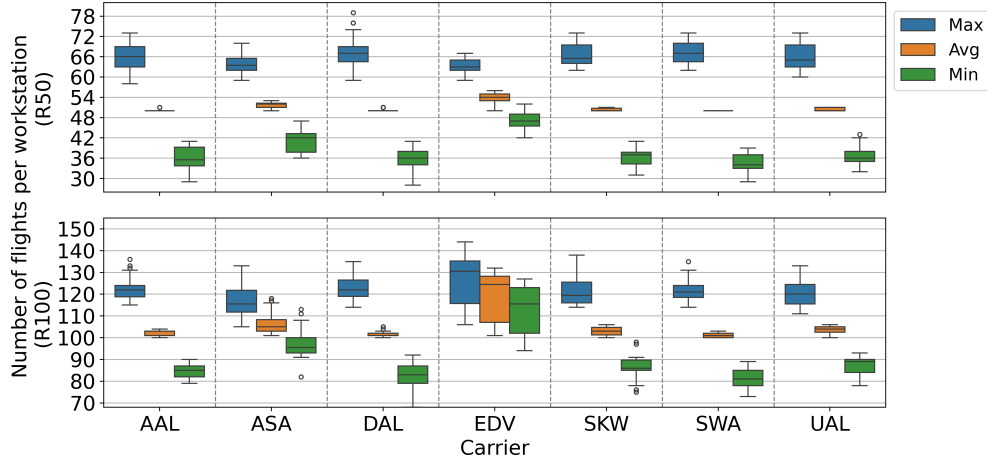


Figure 2.17: Distribution of the number of flights per workstation for different carriers in the FSI-GA solutions for (PWP).

2.6 Conclusions

We addressed a new workload balancing problem for flight dispatchers, aiming to assign flights to workstations to ensure the workload is evenly distributed among the flight dis-

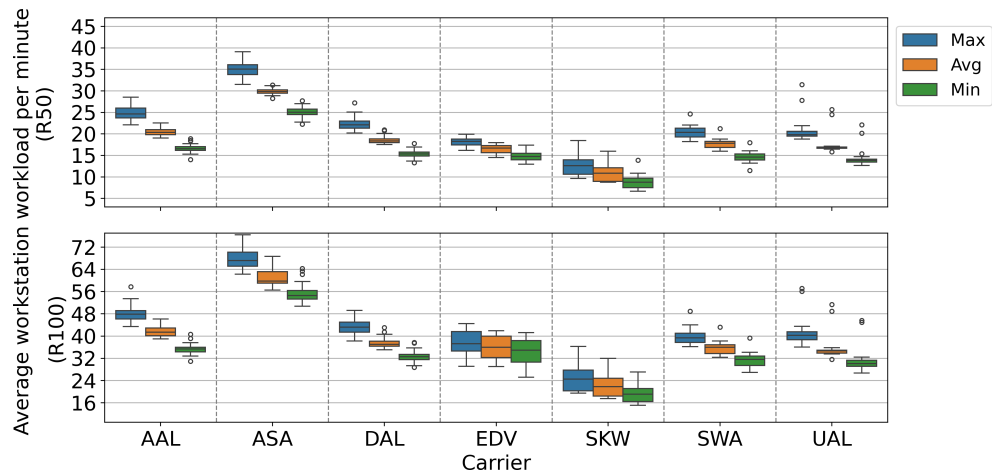


Figure 2.18: Distribution of the workload for different carriers in the FSI-GA solutions for (PWP).

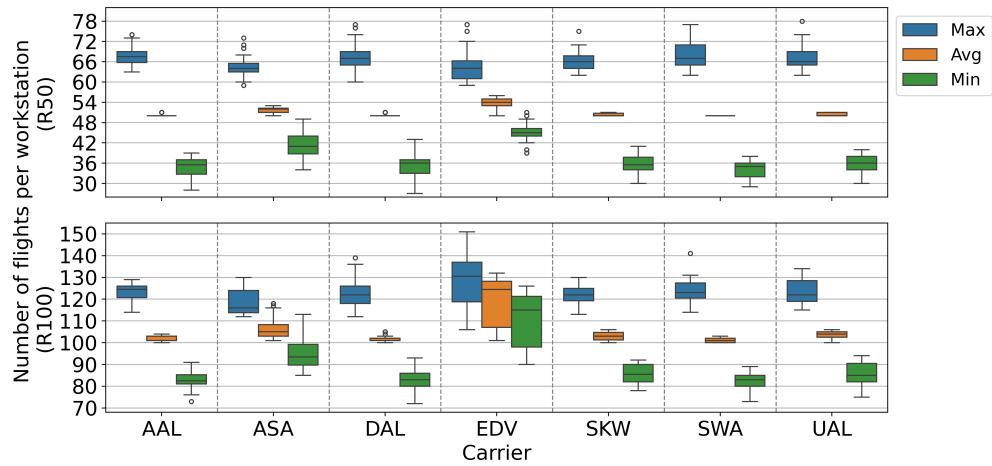


Figure 2.19: Distribution of the number of flights per workstation for different carriers in the FSI-GA solutions for (ADP).

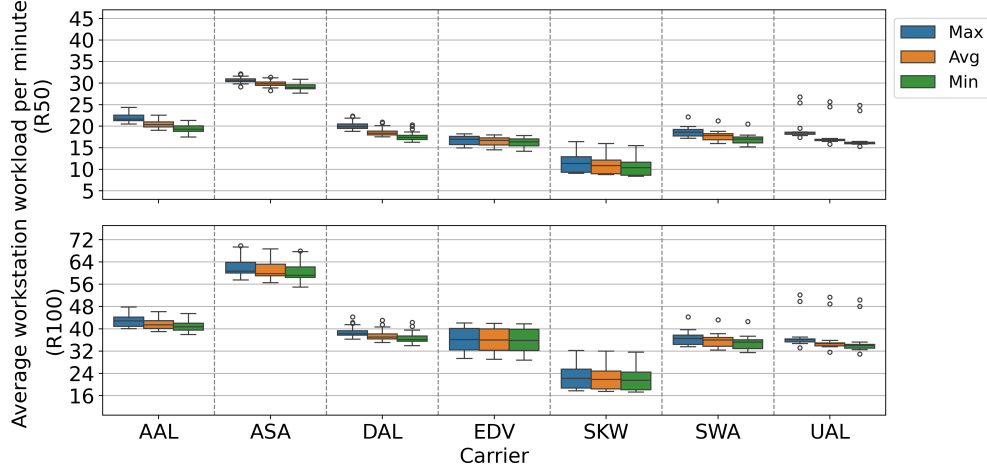


Figure 2.20: Distribution of the workload for different carriers in the FSI-GA solutions for (ADP).

patchers throughout the day. Dynamic and multi-tasking nature of the flight dispatcher tasks distinguishes this problem from the studies found in the current workload balancing and flight dispatcher literature. To our knowledge, this is the first study to introduce a multi-tasking model for flight dispatcher assignments. We developed mixed-integer linear programming models with two nonlinear workload balance measures, derived a Lagrangian-based lower bound, and proposed a fast heuristic that delivers high-quality solutions for real-world instances within seconds. Extensive tests on real-world data from 2019 U.S. domestic flights Bureau of Transportation Statistics (2023b) validates the effectiveness of our approach.

Numerical testing demonstrates the efficiency of the proposed approach in that the Lagrangian lower bound is very tight, and the heuristic finds proven optimal solutions in 33.4% of the instances and is on average 3.5% away from optimality. The direct solution using a commercial solver performed poorly on the peak workload objective and failed to solve even the smallest instances for the absolute deviation objective. Moreover, when assessing the quality of the peak workload and the absolute deviation objectives in terms of workload balance, we found that the latter achieves better workload balance throughout the day at the expense of slightly higher overall peak workload.

The current work opens the door to several new research extensions. Considering other workload balance measures, modelling additional practical features related to flight, dispatcher and workstation requirements, and embedding the proposed Lagrangian approach within an exact solution method that is capable of solving large sizes, are important di-

rections. Moreover, the definition as well as the measure of workload deserves further study and would require experimental data through ethnography studies. As we focus on assigning flights to workstations in our work, there is opportunity to work on the follow up problem, which is to assignment of dispatchers to workstations. The latter is an important research question that needs to take into account dispatcher related aspects like union rules, and work preferences.

Chapter 3

A Multi-Objective Framework for Flight Dispatcher Workload Balancing

As established in Chapter 1, managing flight dispatcher workload is a critical and complex concern for airline operational safety, efficiency, and resilience. Chapter 2 of this thesis introduced and analyzed models focused on balancing this workload, primarily through optimization approaches targeting specific workload metrics such as peak load or deviation from an average.

However, the operational environment of flight dispatch often requires consideration of a broader set of factors beyond a single workload balancing measure. Effective dispatcher assignment may also need to account for other operational efficiencies, practical constraints, and the potential for synergistic task management. Relying solely on a single-objective view of workload balance can overlook these additional important considerations.

Although dispatcher workload has been recognized as a key factor for safety and efficiency (Rayner, 2021), many existing approaches remain single-objective—often centering on the even distribution of tasks among dispatchers or simply ensuring a feasible assignment. This focus can overlook important operational considerations, such as how flight-path similarities might ease monitoring tasks or how certain pairing restrictions can complicate the assignment process. In contrast, a multi-objective framework captures the trade-offs between multiple criteria (e.g., balancing workload, grouping similar flights, and minimizing restricted assignments) and thus offers more realistic solutions.

To address these challenges, this chapter proposes a multi-objective framework that explicitly considers the intertwined aspects of dispatcher workload and operational requirements. Unlike single-objective formulations that focus solely on workload balancing

or feasibility, our approach integrates two additional factors: *(i)* flight-path similarities, which prioritize grouping flights passing through similar weather regions to reduce the area a dispatcher must monitor, and *(ii)* practical assignment restrictions (e.g., close-departing flights, insufficient dispatcher experience for a flight), which may be employed as hand-rules to avoid brief spikes in the workload. By optimizing for the workload balance while addressing the practical preferences and constraints, this multi-objective perspective helps preserve dispatcher well-being, minimize safety risks, and adapt to unexpected events such as weather-related disruptions, all while improving the overall airline efficiency.

3.1 Literature Review

Airline operations literature has long examined task allocation and scheduling problems across a range of domains. In the context of flight dispatchers, however, the literature remains relatively limited, especially regarding models that integrate operational constraints with dispatcher workload balancing. Approaches often focus on single objectives such as minimizing peak workload, or just finding a feasible solution, and do not usually consider more complex factors such as flight-path overlaps, restricted pairings, or regulatory constraints, which might affect their real-world applicability and performance.

Early work on dispatcher scheduling has primarily emphasized distributing flight assignments that aimed to balance dispatcher workload by minimizing extreme workload disparities. Tan et al. (2024a) analyzed the workload balancing of flight dispatchers in the context of scheduled domestic flights, revealing that practical considerations, such as certification constraints, union regulations, and route similarities, play a crucial role in real-world task assignment. Although some feasibility-based methods have been explored in broader contexts (C. A. Santos et al., 2011; Agnetis and Smriglio, 2000), many of these single-objective or simplified approaches do not account for the interaction of flights with similar paths on the dispatcher workload.

In parallel, research into aviation personnel scheduling continues to underscore the importance of accounting for multiple operational factors in optimization models. Recent flight crew scheduling studies embed pilot fatigue considerations into mixed integer programming models, balancing operational costs and safety risks (Lina et al., 2024; Yildiz, Gzara, and Elhedhli, 2017). Advances in workload prediction for air traffic controllers employ data-driven graph methods to manage dynamic task loads (Pang et al., 2023). Similarly, nurse rostering problems demonstrate that satisfying fairness constraints and skill requirements is critical in large-scale scheduling (Burke, De Causmaecker, et al., 2004).

These examples stress that single-objective formulations may overlook hidden trade-offs in highly constrained operational environments.

Motivated by these findings, our work proposes a multi-objective framework that explicitly balances dispatcher workload with operational restrictions and preferences. Unlike prior single-objective formulations, the proposed model integrates practical aspects (e.g., flight path similarities and operational constraints) and examines how workload balancing may influence aspects of flight operations. Real-world events such as recent dispatcher strikes and weather-induced disruptions show why a broader approach is necessary. By adapting the insights from dispatcher scheduling, flight crew planning, and air traffic controller workload studies, our framework aims to develop methods and models capable of addressing the practical demands of modern airline operations.

Our proposed framework adopts a multi-objective strategy that balances dispatcher workload with other key operational indicators. The next section reviews key optimization methods that form the methodological basis for our approach.

3.1.1 Multi-objective Optimization Methods

Multi-objective optimization addresses problems with two or more competing objectives. Instead of a single optimal solution, the goal is to identify a set of Pareto-efficient solutions, each reflecting different trade-offs. A solution is Pareto-efficient if no other feasible solution can improve one objective without worsening another (Ehrgott, 2005).

A common group of approaches are scalarization methods, which combine multiple objectives into a single one by assigning weights. For instance, the weighted sum method creates a composite objective using non-negative weights (Marler and Arora, 2010). This approach is simple to implement but can only identify solutions along the convex envelope of the Pareto front. In contrast, the ϵ -constraint method optimizes a primary objective while treating the remaining objectives as constraints with preset bounds, allowing exploration of non-convex regions at the cost of increased computational cost (Chankong and Haimes, 2008). Similarly, the Chebyshev method seeks solutions that minimize the maximum deviation from ideal objective values, balancing the worst trade-offs among objectives; however, it demands careful weight selection and is computationally intensive (Steuer and Choo, 1983).

Evolutionary algorithms have been widely adopted for complex, large-scale problems. For example, NSGA-II (Deb, Pratap, et al., 2002) uses non-dominated sorting and a crowding distance mechanism in approximating the Pareto front. Recent advances include methods such as NSGA-III (Deb and Jain, 2013) and MOEA/D (Q. Zhang and H. Li, 2007),

which have shown promise in applications like air traffic control and workload balancing (Cai, Ang, and Alam, 2022; Gulbaz et al., 2021). While evolutionary algorithms offer flexibility and the ability to capture diverse solutions, they require careful tuning of parameters and do not always guarantee convergence to the true Pareto front.

In summary, the choice of a multi-objective optimization method depends on the specific problem characteristics and decision-maker preferences. While scalarization methods are straightforward, they cannot obtain interior Pareto solutions. In contrast, constraint-based and evolutionary approaches, such as NSGA-II (Deb, Pratap, et al., 2002), NSGA-III (Deb and Jain, 2013), and MOEA/D (Q. Zhang and H. Li, 2007), are capable of capturing a broader range of trade-offs but generally incur higher computational costs and require careful parameter tuning.

In this chapter, we employ three multi-objective optimization approaches: a weighted sum method, an ϵ -constraint method, and NSGA-II. FSI-GA, as proposed in Chapter 2, is adapted as the main solution method within the weighted sum framework. The ϵ -constraint method follows the approach proposed by Mavrotas (2009), ensuring controlled exploration of non-dominated solutions. While NSGA-III has been introduced as an improvement over NSGA-II, it was not considered in this work because it is primarily designed for many-objective problems, and our study focuses on problems with fewer objectives where NSGA-II performs adequately.

3.2 Problem Setting

Flight dispatcher workload is a critical operational concern that arises from the need to assign flights to dispatchers in a manner that balances both workload and operational demands. For example, consider a scenario in which a dispatcher is responsible for monitoring several flights that depart within a short time span. In such a case, the dispatcher must evaluate weather conditions, coordinate flight releases, and manage real-time communications for multiple flights simultaneously. In this context, workload is defined as a function of the task requirements associated with the flights assigned to the dispatcher. These task requirements include the frequency and timing of departures, the complexity of flight routes, and the demands imposed by monitoring weather and airspace conditions.

In Chapter 2, workload was defined by focusing on two balance measures. One measure aimed at minimizing the peak workload encountered by any dispatcher, while the other focused on minimizing the absolute deviation from the average workload (ADP). These measures were developed with the intent of ensuring an even distribution of tasks and

reducing the potential for overload during critical operational periods. Although the peak workload measure captures the maximum instantaneous burden, it may overlook persistent variations in task demands. Likewise, while the ADP measure tends to smooth the distribution of workload, it may not fully capture other operational factors that influence dispatcher performance. The conclusions in Chapter 2 suggest that these measures provide insights that call for a more complex assessment of dispatcher workload.

In practice, several additional factors impact the overall workload. For instance, when multiple flights traverse similar routes or encounter comparable weather patterns, a dispatcher may exploit shared information to streamline assessments and reduce redundant tasks. Conversely, assigning flights with differing operational demands may force a dispatcher to continually switch contexts, thereby increasing the overall workload. Assignment requirements that restrict the pairing of certain flights or the allocation of flights to specific workstations can also be integrated to support workload balance. These restrictions serve to control the operational complexity faced by dispatchers, complementing workload balancing measures by preventing problematic flight combinations or workstation allocations.

3.2.1 Flight Path Similarity

Multi-tasking is a characteristic feature of flight dispatcher work, as dispatchers are required to manage multiple flights simultaneously. Although each flight may entail a distinct combination of tasks, many flights share common operational characteristics. A dispatcher can leverage previously learned information when flights exhibit similar features, thereby reducing the overall amount of work needed. For instance, a dispatcher managing several transatlantic flights may only need to assess the North Atlantic weather patterns once rather than for each individual flight. In this framework, flight similarity is quantified through a metric that considers the proximity of flight paths. Recent studies have introduced various trajectory similarity measures based on path alignment and sequence-aware distance metrics Todoric and Mastelic (2023). There are many methods focus on point-based comparisons along trajectories, such as Dynamic Time Warping, Longest Common Subsequence, or Fréchet distance, which evaluate similarity based on the sequence and proximity of recorded points (Hu et al., 2023; Magdy et al., 2015). However, monitoring broad weather systems is the primary concern in the context of flight dispatchers, thus, precise point-by-point trajectory matching may be less relevant than assessing the overlap of the general operational regions. Empirical findings indicate that consolidating flights with similar paths can effectively reduce redundant workload and cognitive strain Tan et al. (2024a).

While the focus of this framework is on quantifying pairwise flight similarity to aid in co-assigning flights to a single workstation, it is useful to distinguish this from broader trajectory clustering methodologies. Clustering measures are often used with algorithms like DBSCAN (Ester et al., 1996) or hierarchical clustering (Gariel, Srivastava, and Feron, 2011), aim to partition an entire dataset of trajectories into groups based on shared path characteristics, such as flight path geometry or density. It is important to note that the goal of clustering is typically to identify common routes, traffic patterns, or anomalous flights within a large dataset. In contrast, the similarity measure developed in this chapter is specifically designed to evaluate the degree of overlap in monitoring tasks between any two given flights. As such, this pairwise assessment is better suited in the context of flight dispatchers, where the objective is to group a smaller number of flights with overlapping operational aspects onto the same dispatcher’s workload to reduce cognitive load and redundant tasks.

Consider the set of flights provided in Table 3.1 and illustrated in Figure 3.1. The paths shown in the figure represent the shortest flight paths on Earth between the respective origin and destination airports. These paths do not correspond to the actual flight trajectories, as the precise routes flown are not available. At a glance, flights A and B

Table 3.1: An example set of flights.

Flight ID	Origin	Destination	Departure	Arrival	Direction
Flight A	New York City (JFK)	Los Angeles (LAX)	07:55 AM	10:21 AM	East to West
Flight B	Los Angeles (LAX)	New York City (JFK)	08:00 AM	04:26 PM	West to East
Flight C	Miami (MIA)	Albuquerque (ABQ)	07:50 AM	11:44 AM	Southeast to Southwest
Flight D	Dallas (DFW)	San Francisco (SFO)	02:00 PM	03:44 PM	South to West
Flight E	Seattle (SEA)	Orlando (MCO)	11:00 PM	07:19 AM	Northwest to Southeast

exhibit the highest similarity due to their identical origin-destination airport pairs. Flights A and E, on the other hand, are distinctly dissimilar, as their paths span entirely different geographic regions. Flights C and D may be considered moderately similar, as their routes intersect over a limited region.

Given these observations, we introduce a flight similarity metric that, for a given pair of flights, assesses the proximity between flight paths to calculate a similarity score. Let δ_{ij} represent the similarity score between flights $i, j \in I$, and let ρ denote a given proximity parameter. Initially, the great circle route, which is the shortest flight path between the origin and destination airports of each flight, is calculated. Next, the region within a ρ -km distance of each flight path is determined.

Unlike traditional trajectory similarity methods, which primarily compare flight paths through waypoint alignment and sequence matching Todoric and Mastelic (2023), the

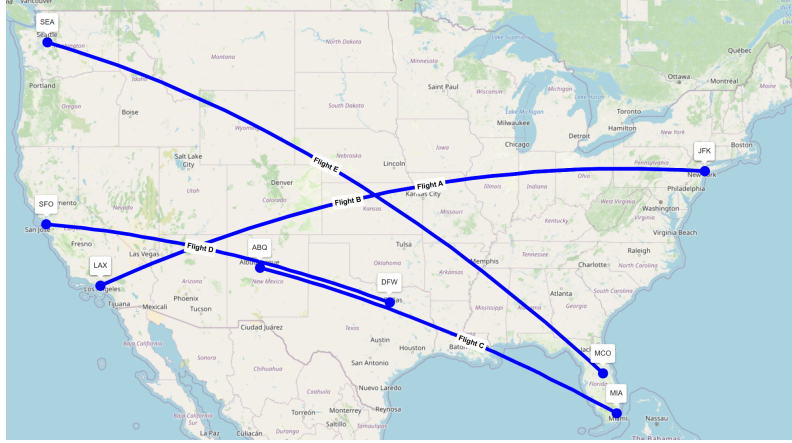


Figure 3.1: Illustration of the flight paths of the flights provided in Table 3.1.

proposed approach quantifies similarity based on the spatial intersection of operational regions. This methodology is particularly relevant for flight dispatchers, as it emphasizes shared operational characteristics rather than precise trajectory matching.

Furthermore, dispatchers have demonstrated a preference for managing flights exhibiting similar characteristics, such as aircraft type Pei et al. (2021). Grouping flights with overlapping paths can streamline tasks such as weather assessment and deciding on alternates, given that flights traversing identical or adjacent geo-spatial areas are typically subject to similar meteorological conditions. By facilitating a consolidated evaluation of weather data for multiple flights, this approach reduces redundant evaluations and cognitive demands, which is essential for effective workload management (Sheth, Amis, and Gutierrez-Nolasco, 2009).

For a given flight pair (i, j) and radius ρ , the similarity score δ_{ij} is calculated in Java as follows: the origin-destination coordinate pairs $(lat_{i1}, lon_{i1}), (lat_{i2}, lon_{i2})$ for flight i , and $(lat_{j1}, lon_{j1}), (lat_{j2}, lon_{j2})$ for flight j are set to the corresponding airport coordinates. A coordinate geometry is then established using the Java Topology Suite (JTS) (Davis, 2024), converting coordinate pairs into line geometries. Using the GeoTools library, these lines are projected from the World Geodetic System (WGS84) into a local Azimuthal Equidistant coordinate system (NGA, 2014; OSGeo, 2024), with the projection centered at the mean coordinates of all four endpoints to minimize distortion. Each projected line geometry is buffered by radius ρ , resulting in two spatial regions. Finally, the similarity score δ_{ij} is calculated as the ratio of the intersection area to the union area of these two regions using topological operations within the JTS library.

3.2.2 Assignment Restrictions

In addition to similarity-driven assignments, certain flight assignment restrictions affect the dispatcher workload. Such restrictions may often be imposed by airlines to prevent excessive complication of dispatcher tasks. For example, regulations requiring dispatchers be fully familiar with the weather conditions along the flight route may restrict them from releasing the flights (FAA, 2025). Restricting the assignment of flights with closely spaced departure times to the same dispatcher can help prevent overload (Munro and Mogford, 2018). Furthermore, dispatcher experience and preferences may lead to additional assignment guidelines, for instance, a less experienced dispatcher might be assigned predominantly regional, short-haul flights (L. Huang, Nneji, and Cummings, 2019).

Assignment restrictions can be categorized into two types. One type involves flight-to-flight restrictions that prevent certain pairs of flights from being assigned to the same workstation. The other type involves flight-to-workstation restrictions, which limit the assignment of flights to particular workstations. These rules are typically considered as soft constraints to avoid infeasible solutions.

Flight path similarity encourages assigning similar flights together while assignment restrictions do the opposite to reduce the workload. In our formulation, the workload balance objective, flight path similarity, and assignment restrictions are optimized together in a multi-objective setting. This combined approach aims to find a balanced assignment that accounts for both the operational efficiency gained through flight similarities, better-manageable cognitive load through assignment restrictions, and an overall balance among dispatchers through workload balance objectives.

3.2.3 Illustrative Example and Operational Objectives

To illustrate the practical integration of flight path similarity and assignment constraints, we consider a scenario comprising five flights, each characterized by distinct origins, destinations, and schedules across various regions in the United States. Flight path similarity is quantified based on the spatial overlap within a 150 km buffer surrounding each flight trajectory. Assignment constraints include departure timing restrictions, which prevent flights departing within five minutes of each other from being assigned to the same dispatcher, as well as workstation-specific restrictions. In particular, workstation 1 experiences a shift change between 09:30 and 10:00 AM, during which processing flight departures is difficult.

Table 3.2 summarizes the characteristics of these flights, including their origins, destinations, departure and arrival times, and geographic directions. Figure 3.2 presents the

shortest paths between origin and destination airports, accompanied by their respective 150 km buffers. These paths reflect great-circle distances rather than actual flight trajectories, which are unavailable. Flights A and B exhibit the highest similarity, with a buffer overlap of 53.8%, indicating a potential reduction in cognitive workload if assigned together. However, due to their departure times being only two minutes apart, these flights cannot be allocated to the same dispatcher. The overlaps between flights B and C (2.9%) and flights C and D (6.3%) suggest minimal operational similarity, while flights D and E, operating in distinct geographic corridors, share only a 4.2% overlap. Additionally, the workstation restriction impacts flight E, which departs at 10:08 AM, shortly after the shift change period at workstation 1, making it preferable to assign this flight to workstation 2 to avoid potential operational disruptions.

Table 3.2: An example set of flights.

Flight ID	Origin	Destination	Departure	Arrival	Direction
Flight A	Los Angeles (LAX)	Denver (DEN)	06:08 AM	09:23 AM	Western to Central
Flight B	San Diego (SAN)	Denver (DEN)	06:10 AM	09:30 AM	Western to Central
Flight C	Phoenix (PHX)	Chicago (ORD)	06:11 AM	10:56 AM	Western to Central
Flight D	Chicago (ORD)	New York (JFK)	08:10 AM	11:10 AM	Midwest to Eastern
Flight E	Miami (MIA)	Boston (BOS)	10:08 AM	01:25 PM	Southeast to Eastern

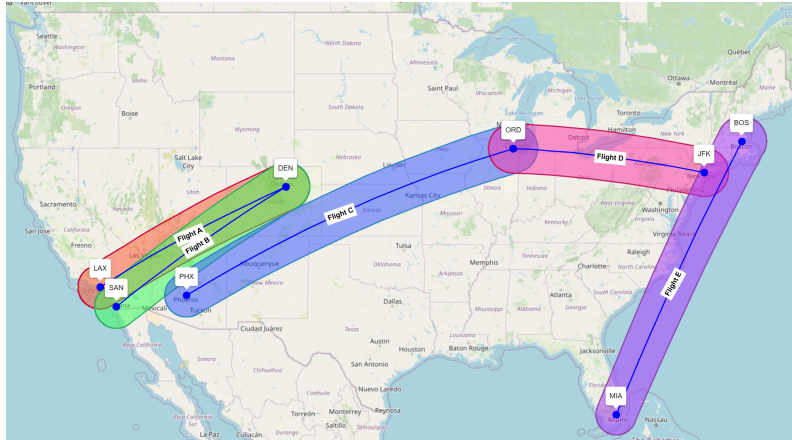


Figure 3.2: Illustration of flight paths and regions for radius $\rho = 150\text{km}$.

Considering these constraints, a suitable assignment strategy might involve assigning flights A and C to workstation 1, and flights B, D, and E to workstation 2. This allocation respects both the timing and workstation-specific constraints: flights A and C have

sufficient temporal separation and minimal overlap, enabling manageable dispatcher workloads at workstation 1. Conversely, workstation 2 efficiently handles flights B, D, and E, strategically circumventing the shift-change interval and maintaining a balanced workload by exploiting the limited spatial overlap.

This illustrative example highlights the critical importance of integrating flight path similarity and operational constraints into dispatcher assignment models. By explicitly accounting for both spatial and temporal considerations, the proposed modeling approach ensures realistic, efficient, and operationally viable dispatcher assignments, ultimately contributing to enhanced workload management and improved operational performance.

3.3 Mathematical Model

In this section, we present a comprehensive mathematical model for the Multi-objective Workload Balancing problem (MWBP). This problem integrates three critical aspects of flight dispatcher scheduling. First, it seeks to balance the workload across workstations as measured by the absolute deviation from the average workload (ADP). Second, it incorporates assignment restrictions in two forms: *(i)* flight–flight restrictions that prevent specific pairs of flights from being assigned to the same workstation and *(ii)* flight–workstation restrictions that preclude certain flights from being assigned to specified workstations. Third, it encourages the consolidation of flights with similar flight paths by maximizing a similarity score between flights assigned to the same workstation.

We denote by I the set of flights and by K the set of workstations. Let T be the set of discrete time points, e.g., each minute over the operational horizon. T is used to capture the dynamic nature of the workload, which varies over time as flights go through different stages. For each flight $i \in I$ and time point $t \in T$, let h_{it} represent the load requirement of flight i at time t . The average workload per workstation at time t is then given by

$$\eta_t = \frac{1}{|K|} \sum_{i \in I} h_{it}.$$

A binary decision variable x_{ik} is defined for each flight $i \in I$ and workstation $k \in K$, where $x_{ik} = 1$ if flight i is assigned to workstation k , and 0 otherwise. In addition, for each flight $i \in I$, let $S_i \subseteq I$ denote the set of flights that should not be assigned to the same workstation as i (flight–flight restrictions), and let $L_i \subseteq K$ denote the set of workstations from which flight i is restricted (flight–workstation restrictions). Finally, for any pair of flights $i, j \in I$, the similarity score δ_{ij} quantifies the degree of geo-spatial overlap between their respective

flight paths. This specific measure was chosen because it reflects operational synergies relevant to flight dispatchers, such as the potential for shared weather monitoring and coordinated regional planning when flights traverse similar airspace corridors. As detailed in Section 3.2.1, δ_{ij} is calculated based on the ratio of the intersection to the union of buffered geo-spatial regions constructed around the nominal flight paths. This approach emphasizes shared operational environments, which is particularly relevant as dispatcher needs to maintain situational awareness over common weather systems or restricted areas, rather than focusing on precise trajectory alignment.

The problem is modeled as a multi-objective formulation with four objectives. The first objective minimizes the total absolute deviation from the average workload across all workstations and time points. The second objective minimizes the violations of flight–flight restrictions. The third objective minimizes the violations of flight–workstation restrictions. The fourth objective maximizes the total similarity score of flights assigned together.

A nonlinear formulation of the problem is as follows:

$$\min \quad z_1 = \sum_{k \in K} \sum_{t \in T} \left| \sum_{i \in I} h_{it} x_{ik} - \eta_t \right| \quad (3.1)$$

$$\min \quad z_2 = \sum_{k \in K} \sum_{i \in I} \sum_{j \in S_i} x_{ik} x_{jk} \quad (3.2)$$

$$\min \quad z_3 = \sum_{i \in I} \sum_{k \in L_i} x_{ik} \quad (3.3)$$

$$\max \quad z_4 = \sum_{k \in K} \sum_{i, j \in I: i < j} \delta_{ij} x_{ik} x_{jk} \quad (3.4)$$

$$\text{s.t.} \quad \sum_{k \in K} x_{ik} = 1 \quad \forall i \in I \quad (3.5)$$

$$x_{ik} \in \{0, 1\} \quad \forall i \in I, k \in K. \quad (3.6)$$

A linear reformulation is achieved by introducing auxiliary variables. Let v_{tk} represent the absolute difference between workload of workstation $k \in K$ from the average workload η_t at time $t \in T$, and let y_{ijk} be a binary variable that indicates whether flights i and j are

assigned to the same workstation k . The linear model is then formulated as:

$$(MWBP) : \quad \min \quad z_1 = \sum_{k \in K} \sum_{t \in T} v_{tk} \quad (3.7)$$

$$\min \quad z_2 = \sum_{i \in I} \sum_{j \in S_i} \sum_{k \in K} y_{ijk} \quad (3.8)$$

$$\min \quad z_3 = \sum_{i \in I} \sum_{k \in L_i} x_{ik} \quad (3.9)$$

$$\max \quad z_4 = \sum_{k \in K} \sum_{i, j \in I: i < j} \delta_{ij} y_{ijk} \quad (3.10)$$

$$\text{s.t.} \quad \sum_{k \in K} x_{ik} = 1 \quad \forall i \in I \quad (3.11)$$

$$\sum_{i \in I} h_{it} x_{ik} - \eta_t \leq v_{tk} \quad \forall t \in T, k \in K \quad (3.12)$$

$$- \sum_{i \in I} h_{it} x_{ik} + \eta_t \leq v_{tk} \quad \forall t \in T, k \in K \quad (3.13)$$

$$y_{ijk} \leq x_{ik} \quad \forall i, j \in I : i < j, k \in K \quad (3.14)$$

$$y_{ijk} \leq x_{jk} \quad \forall i, j \in I : i < j, k \in K \quad (3.15)$$

$$y_{ijk} \geq x_{ik} + x_{jk} - 1 \quad \forall i, j \in I : i < j, k \in K \quad (3.16)$$

$$x_{ik}, y_{ijk} \in \{0, 1\}, g_{tk}^-, g_{tk}^+ \geq 0 \quad \forall i, j \in I : i < j, t \in T, k \in K, \quad (3.17)$$

In this formulation, constraints (3.12) and (3.13) enforce that, at each time point $t \in T$, the total workload assigned to workstation $k \in K$ is balanced against the average workload η_t , with g_{tk}^- and g_{tk}^+ capturing any deviations. Constraints (3.14)–(3.16) ensure that the auxiliary variable y_{ijk} correctly indicates whether flights i and j are assigned to the same workstation.

Overall, the model simultaneously optimizes four objectives: minimizing workload imbalance (objective z_1), minimizing violations of flight–flight restrictions (objective z_2), minimizing violations of flight–workstation restrictions (objective z_3), and maximizing the similarity of flights assigned together (objective z_4). This integrated approach provides a framework for scheduling that addresses both operational efficiency and dispatcher workload.

3.4 Solution Approaches

To address MWBP, we investigate three approaches: the weighted sum method, the ϵ -constraint method, and the Non-dominated Sorting Genetic Algorithm II (NSGA-II). A modified version of the Focus-Search-Improve Genetic Algorithm (FSI-GA), is employed to accommodate practical extensions and is primarily utilized in the first two methods.

3.4.1 Weighted Sum Method

The weighted sum method transforms multiple objectives into a single scalar objective by assigning weights to each objective, facilitating the exploration of trade-offs through variation of these weights. We employ a grid-search approach, generating a set of weight combinations derived from the payoff table (Chankong and Haimes, 2008). Specifically, objective 1 is assigned a constant weight of 1, while the ranges for weights of objectives 2, 3, and 4 are calculated using:

$$w_p \in \left\{ \left[\frac{z_1^*}{UB_p}, \frac{z_1^*}{LB_p} \right] \mid p \in \{2, 3, 4\} \right\},$$

where $z_1^* = LB_1$ since the first objective is minimized. Each weight combination $(w_1 = 1, w_2, w_3, w_4)$ defines a single-objective optimization problem:

$$\min \quad w_1 z_1 + w_2 z_2 + w_3 z_3 - w_4 z_4 \tag{3.18}$$

$$\text{s.t.} \quad (3.7) - (3.17), \tag{3.19}$$

which can be solved exactly by a solver or approximated using the modified FSI-GA. Solutions obtained via FSI-GA may include dominated solutions due to its stochastic nature, thus, a Pareto front is derived by selecting non-dominated solutions from all recorded outcomes.

Methodology for determining the weight ranges was proposed to facilitate a balanced exploration of the objective space. The payoff table yields the extreme values for each individual objective, providing a basis for establishing ranges for their respective weights in the objective. By setting $w_1 = 1$, z_1 effectively serves as the reference objective. The weights for the other objectives are then scaled relative to the optimal value of z_1 and their own individual objective ranges.

This scaling approach is important because different objectives possess different magnitudes. Without normalization the weighted sum could be disproportionately dominated by

objectives with numerically larger values. This scaling aims to ensure that each objective can exert a level of influence on the direction of the search. This allows the grid search over these normalized weight combinations to more effectively explore diverse trade-off solutions along the Pareto front, particularly when objective scales vary significantly (Chankong and Haimes, 2008).

3.4.2 ϵ -Constraint Method

The ϵ -constraint method optimizes a primary objective while constraining the others within specified bounds. Adopting the enhanced ϵ -constraint approach of Mavrotas (2009), we generate evenly spaced constraint values between LB_p and UB_p for each secondary objective $p \in \{2, 3, 4\}$. The resulting optimization problems are:

$$\min \quad z_1 \tag{3.20}$$

$$\text{s.t.} \quad (3.7) - (3.17), \tag{3.21}$$

$$z_p \leq e_p, \tag{3.22}$$

$$p \in 2, 3,$$

$$z_4 \geq e_4, \tag{3.23}$$

where e_p is the ϵ -constraint corresponding to objective $p \in P$. Efficiency of solutions is ensured by incorporating slack and surplus variables into an alternative formulation:

$$\min \quad z_1 - \epsilon \left(\frac{s_2}{r_2} + \frac{s_3}{r_3} + \frac{s_4}{r_4} \right) \tag{3.24}$$

$$\text{s.t.} \quad (3.7) - (3.17), \tag{3.25}$$

$$z_p + s_p = e_p, \tag{3.26}$$

$$p \in 2, 3,$$

$$z_4 - s_4 = e_4, \tag{3.27}$$

$$s_2, s_3, s_4 \geq 0, \tag{3.28}$$

where $r_p = UB_p - LB_p$ denotes the range of objective p . Unlike the weighted sum approach, adapting the FSI-GA to this formulation introduces significant complexity, as feasibility constraints inherently generate infeasible solutions. Extensive modifications to FSI-GA, such as feasibility-based penalties or solution repair mechanisms, are necessary for satisfactory performance. The FSI-GA can be adapted to handle these constraints by penalizing infeasible solutions in its fitness function, or a repairing procedure for the infeasible solutions, but this requires modifications to the algorithm.

3.4.3 Non-dominated Sorting Genetic Algorithm II (NSGA-II)

NSGA-II directly addresses multi-objective optimization without scalarizing objectives. Solutions are encoded as integer arrays representing assignments of flights to workstations. Objective values are computed from these assignments, and the genetic algorithm iteratively evolves an initial random population through selection, crossover, and mutation operations.

In alignment with NSGA-II methodology (Deb, Pratap, et al., 2002), population management employs non-dominated sorting to identify Pareto fronts and applies a crowding distance metric to maintain solution diversity. Tournament selection favors solutions with superior Pareto ranks or greater crowding distances when ranks are equivalent. Uniform crossover and mutation operators randomly alter flight assignments to generate offspring solutions.

3.4.4 FSI-GA Adaptations

Modifications of FSI-GA, essential for accommodating weighted sum and ϵ -constraint methods, involve tailored adjustments to fitness evaluations and solution handling. Specifically, the weighted sum approach requires adaptation of the fitness function to scalarized objectives, whereas the ϵ -constraint approach demands advanced techniques, such as penalization or repair strategies, to effectively manage the infeasibilities inherent in constrained optimization formulations.

In the numerical tests, the weighted sum method was implemented because it offers a straightforward scalarization of the objective vector. However, the ϵ -constraint method was not adopted since there are multiple ways to implement the requisite penalty or repair strategies, which become more complex when constraints apply directly to the objectives under optimization. Moreover, certain sets of constraints can result in an infeasible problem, adding significant implementation and computational challenges that fall beyond the scope of this study.

3.5 Numerical Analysis

This section presents a detailed analysis of the numerical results obtained from computational experiments designed to evaluate various solution methods for the multi-objective workload balancing problem. The section encompasses the benchmark instances used, the

experimental setup, a comparison of the performance of different solution methods, and an analysis of the problem over the set of benchmark instances.

3.5.1 Benchmark Instances, Experimental Setup, and Implementation Details

The benchmark instances utilized in this chapter are the same as those employed in Chapter 2, originating from data published by Bureau of Transportation Statistics (2023b) and comprising 6,205 historical flight schedules from 17 airlines. These schedules encompass all U.S. flights operated in 2019. To ensure computational tractability and diversity in test scenarios, a subset of 231 instances from seven airlines was selected. Table 3.3 summarizes these instances, detailing the number of cases per carrier and presenting key statistics on the flights and time-points within each instance. Specifically, the table includes the minimum, average, and maximum values for the number of flights and time-points, illustrating the scale and complexity of the selected problem instances. For each benchmark instance, two configurations are tested to reflect the impact of resource availability. In the first configuration, referred to as R50, the number of workstations is set to approximately one for every 50 flights. In the second configuration, referred to as R100, there is approximately one workstation for every 100 flights. Formally, the number of workstations is given by $|K| = \lfloor |I|/50 \rfloor$ for R50 and $|K| = \lfloor |I|/100 \rfloor$ for R100.

Table 3.3: Flight summary statistics of the test instances, categorized by carrier.

Carrier	Instances	Flights per Instance			Time-Points per Instance		
		Min.	Avg.	Max.	Min.	Avg.	Max.
Alaska Airlines	24	527	611	640	325	355	390
American Airlines	24	1,889	2,101	2,208	1,331	1,389	1,448
Delta Air Lines	83	1,706	2,276	2,627	1,151	1,262	1,330
Endeavor Air	24	313	378	431	591	651	731
SkyWest Airlines	22	1,415	1,774	2,363	1,091	1,166	1,249
Southwest Airlines	35	2,656	2,965	3,968	278	299	335
United Air Lines	19	1,161	1,416	1,870	1,107	1,176	1,334
Overall	231	313	1,873	3,968	278	955	1,448

The computational experiments were conducted on a system equipped with a 3.6 GHz Intel i7-4790 processor and 10 GB of RAM. The solution methods were implemented and

executed using Java 22, with the mathematical model solver utilizing CPLEX 22.11. The methods used to obtain sensitivity analysis results were implemented in Python 3.11.5 and run on Cedar, a server of Digital Research Alliance of Canada (2025b), using 12 cores of Intel Xeon E5-2683 v4 and 32 GB RAM.

Under the workload balance objective, the first operational rule requires that flights with scheduled departures within a five-minute interval cannot share the same workstation. This rule prevents the dispatcher from having to manage multiple simultaneous departures, thereby reducing the risk of workstation conflicts. In addition, flights exceeding six hours in duration are assigned to workstations different from those handling shorter flights. This approach simulates the practical assignment of international flights to dedicated workstations, which is relevant since the available dataset does not contain international flights. Both considerations aim to increase operational safety and effective workload management. Second, workstation availability is constrained according to airport flight volume. Specifically, 10% of workstations are restricted from handling flights associated with airports in the bottom 25% of daily flight volume, while an additional 10% are reserved exclusively for flights involving the five airports with the highest traffic volume. This differentiation is motivated by practical considerations observed in other areas of aviation operations, where higher-traffic environments typically require more experienced personnel. The airport flight volume is computed as the total number of incoming and outgoing flights for the given set of flights. Third, similarity scores are computed using buffer regions of 150 kilometers around flight paths. The similarity score δ_{ij} is calculated as the ratio of intersection to union of these buffers for flights i and j .

In the remainder of this section, we first analyze the performance of solution methods and the trade-offs between objectives on a small-scale instance, where optimal solutions can be obtained. We then compare these results to a larger problem instance to illustrate the increase in problem complexity and the impact on solution quality. Following this, we present an analysis of the objective space over a set of benchmark instances, using approximations obtained from NSGA-II. While these results are not optimal, they provide insight into the range and distribution of objective values, supported by the small-instance analysis demonstrating that NSGA-II solutions closely approximate the Pareto front. Finally, we conduct a sensitivity analysis on the small problem instance to evaluate how alternative operational rules affect solution method performance and objective trade-offs.

3.5.2 Analysis on a Representative Small Instance

An instance derived from Allegiant Air’s historical flight schedule for February 9th is used for this analysis. It consists of 50 flights, randomly selected from an original set of 200,

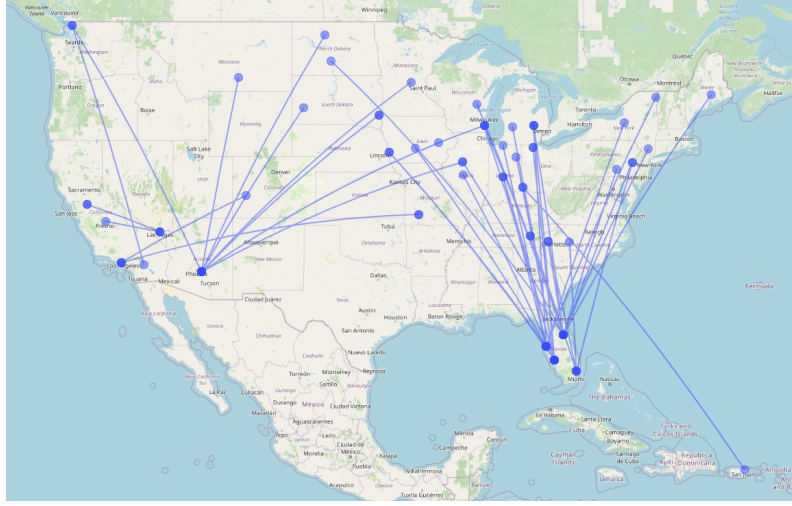


Figure 3.3: Flight paths and airport connections.

with two available workstations and a total of 173 distinct time points. Figure 3.3 depicts the flight network, where nodes represent airports and lines indicate flight paths. The shading intensity of each line reflects the frequency of flights along each route, indicating traffic density. Distances between airports are computed using great-circle approximations, as actual flight path data is not available. This visualization provides spatial context for the optimization results discussed in this subsection.

Due to the size of this problem instance and the predominance of short flights, the long-haul separation rule is applied with a threshold of 4 hours. Furthermore, the traffic-based airport-workstation restrictions are simplified, with one workstation dedicated to flights involving the busiest airports and the other assigned to flights from airports with the lowest traffic volumes.

Table 3.4 summarizes key metrics for the Pareto front obtained from the multi-objective optimization methods. These metrics include the total number of unique non-dominated solutions (Size), the proportion of each method’s solutions that are non-dominated (Own), the percentage contribution of each method to the overall Pareto front (Total), and the achieved ranges for the four objectives z_1 through z_4 .

The results demonstrate that NSGA-II generated a substantially larger number of unique non-dominated solutions (1,878) within a computational time of 300 seconds. The weighted sum and modified ϵ -constraint methods required significantly longer computation times, often exceeding one hour, and yielded far fewer unique solutions. In particular, the

Table 3.4: Pareto front statistics for the Allegiant Air instance

Method	Time (s)	Size	% Non-dom.		Objective Ranges			
			Own	Total	z_1	z_2	z_3	z_4
ϵ -constraint	3,973	39	92.31%	2.14%	(2,865, 33,653)	(4, 16)	(8, 12)	(71.58, 142.62)
NSGA-II	300	1,878	86.90%	97.03%	(2,955, 29,185)	(3, 16)	(4, 18)	(66.34, 141.36)
FSI-GA w/Solver weights	11,075	5	0.00%	0.00%	(4,565, 6,659)	(3, 3)	(5, 7)	(74.89, 80.55)
FSI-GA w/NSGA-II weights	1,245	36	13.89%	0.30%	(3,725, 7,203)	(3, 7)	(4, 12)	(70.83, 96.36)
Solver w/Solver weights	9,864	19	73.68%	0.83%	(3,101, 7,203)	(3, 7)	(4, 10)	(74.62, 96.36)
Solver w/NSGA-II weights	3,557	4	100.00%	0.24%	(3,797, 5,501)	(3, 3)	(5, 7)	(81.10, 82.30)

ϵ -constraint method identified only 39 unique solutions in 3,973 seconds. FSI-GA variants with solver-derived or NSGA-II-derived weights also produced a limited number of non-dominated solutions, with varying computational times. Similarly, solver-based methods with different weight assignments found only a small set of solutions.

The variation in computational times and the disparity in the number of non-dominated solutions highlight substantial differences in the efficiency and effectiveness of the methods applied to this problem instance. The weighted sum method, while computationally straightforward, is limited by its difficulty in handling non-convex Pareto fronts and often requires systematic variation of weights to approximate the front. The ϵ -constraint method, although theoretically capable of identifying all Pareto-optimal solutions, becomes computationally intensive depending on the discretization of ϵ values. In contrast, NSGA-II efficiently explores the objective space and identifies a diverse set of high-quality solutions within a relatively short computation time, making it well-suited to this instance.

Figures 3.4 and 3.5 illustrate trade-offs between the workload balance objective z_1 and the similarity score z_4 . Figure 3.4 presents the non-dominated solutions identified by each tested method, considering these two objectives only. The NSGA-II solutions span a broad range of both ADP and similarity score values, with a few extreme solutions extending the ADP range. Assigning all flights to a single workstation maximizes the similarity score objective, but this assignment directly conflicts with workload balance. On the other hand, an assignment with good workload balance results in lower similarity scores, highlighting the trade-off between these objectives. Figure 3.5 extends this analysis by comparing the non-dominated z_1 and z_4 objectives obtained by each method to the overall non-dominated solutions identified across all four objectives. This comparison illustrates how the consideration of operational restriction objectives z_2 and z_3 alters the structure of the Pareto front. Solutions that perform well in terms of similarity and workload balance alone often fail to satisfy these operational restrictions. This highlights that considering all objectives simultaneously is necessary to identify solutions that are both efficient and operationally feasible.

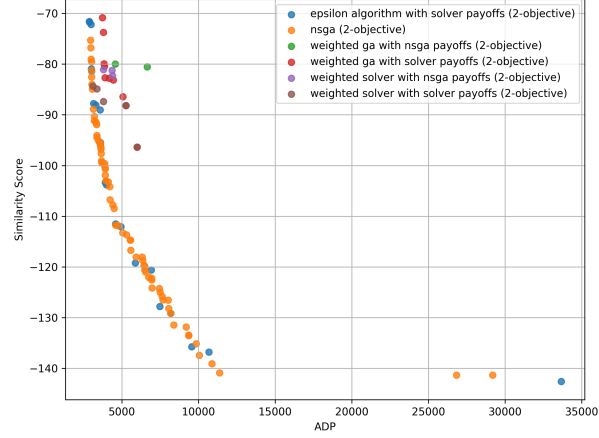


Figure 3.4: Non-dominated ADP z_1 and Similarity Score z_4 objectives from the tested methods in the Allegiant Air instance.

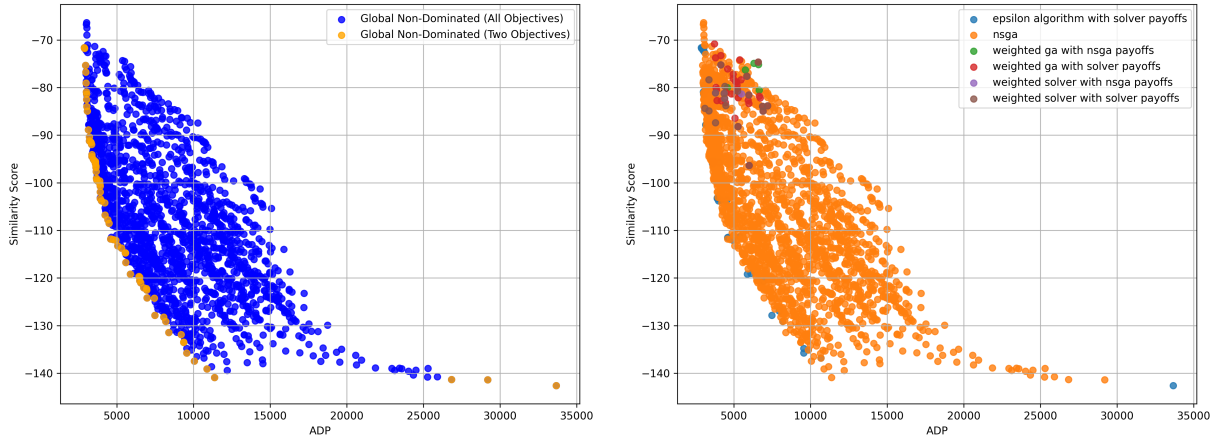


Figure 3.5: Comparison of non-dominated z_1 and z_4 objectives with overall non-dominated objectives from all methods in the Allegiant Air instance.

Table 3.5 reports the payoff table obtained by the solver for the Allegiant Air instance. Each cell in the table indicates the value of the objective listed in the column when the objective in the corresponding row is optimized. This structure highlights the trade-offs between objectives and the range of outcomes achievable when focusing on each objective individually.

Table 3.5: Payoff table obtained by solver for the instance Allegiant Air Feb 9 2019.

	z_1	z_2	z_3	z_4
min z_1	2,864	6	12	71.6
min z_2	3,024	3	5	121.7
min z_3	5,391	5	4	138.9
max z_4	33,653	16	8	142.6

When the workload balance objective z_1 is minimized, it takes the value 2,864, while the corresponding values for z_2 , z_3 , and z_4 are 6, 12, and 71.6, respectively. Maximizing the similarity score z_4 to 142.6 leads to a substantially higher z_1 value of 33,653, illustrating the conflict between workload balance and flight similarity. Minimizing the number of conflicting flights z_2 results in a z_1 value of 3,024 and a similarity score of 121.7, while minimizing workstation restriction violations z_3 yields a z_1 value of 5,391 and a similarity score of 138.9. These outcomes highlight the trade-offs among the objectives, where improvements in one dimension come at the expense of others.

The payoff table provides insight into the extreme points of the Pareto front, showing the best-case outcomes for each individual objective and the corresponding trade-offs in the other objectives. However, a single solution that simultaneously attains the optimal values for all objectives does not exist, highlighting the inherent conflicts between them.

3.5.3 Problem Complexity and Performance on a Large Instance

To examine problem complexity and evaluate performance on larger instances, an instance derived from United Airlines is considered. It consists of 1,504 flights, 30 workstations, and 1,189 time points, representing a significantly more complex scenario than the Allegiant Air instance.

Table 3.6 presents Pareto front statistics for the United Air Lines instance. NSGA-II generated 7,505 non-dominated solutions within the 300-second time limit. Exact methods failed to produce meaningful solutions, terminating with optimality gaps exceeding 95%

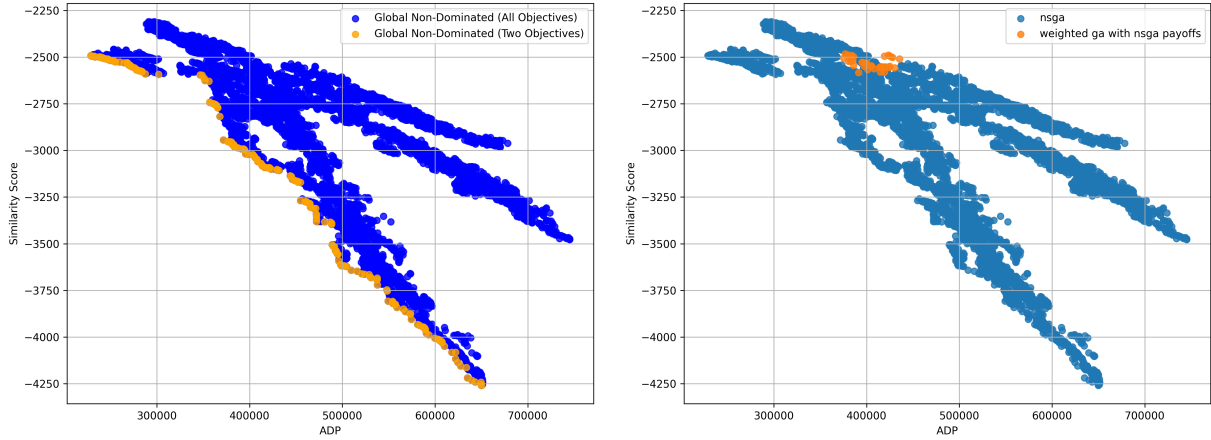


Figure 3.6: Comparison of non-dominated z_1 and z_4 objectives with overall non-dominated objectives from all methods in the United Air Lines instance.

after three hours of runtime. In contrast, the genetic algorithm variant with weights derived from NSGA-II produced only 37 non-dominated solutions in 5,545 seconds. NSGA-II solutions were entirely non-dominated and accounted for 99.65% of the combined Pareto front. The objective ranges obtained by NSGA-II were substantially broader than those of the GA variant, indicating more thorough exploration of the objective space. These results demonstrate the scalability and effectiveness of NSGA-II for larger multi-objective optimization problems.

Table 3.6: Pareto front statistics for the United Air Lines instance

Method	Time (s)	Size	% Non-dom.		Objective Ranges			
			Own	Total	z_1	z_2	z_3	z_4
NSGA-II	300	7,505	100.00%	99.65%	(228,920, 745,219)	(1,398, 5,794)	(0, 49)	(2,309, 4,260)
FSI-GA w/NSGA-II weights	5,545	37	70.27%	0.35%	(375,244, 435,538)	(3,750, 4,029)	(0, 4)	(2,482, 2,583)

Figures 3.6 and 3.7 compare the objective values obtained by NSGA-II and the FSI-GA with NSGA-II weights for the United Air Lines instance. Figure 3.6 presents the relationship between the workload balance objective z_1 and the similarity score z_4 , highlighting the trade-offs between these objectives.

NSGA-II covers a broad range of values for both objectives, demonstrating its ability to identify solutions that reflect varying balances between workload distribution and flight similarity. In contrast, the FSI-GA variant yields solutions concentrated in a narrower

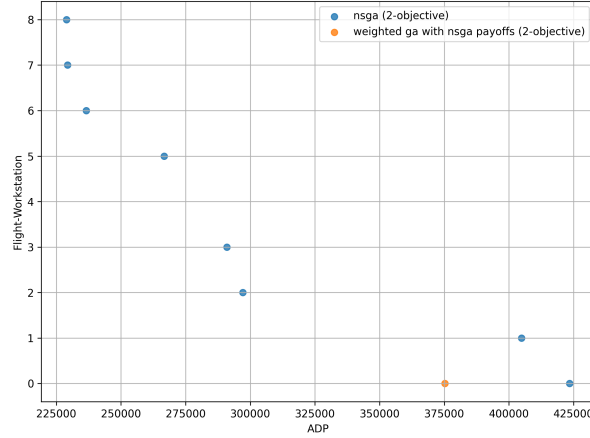


Figure 3.7: Non-dominated ADP z_1 and Similarity Score z_4 objectives from the tested methods in the Allegiant Air instance.

region of the objective space, indicating limited exploration and diversity.

Figure 3.7 compares the workload balance objective z_1 with workstation restriction violations z_3 . The GA with NSGA-II weights achieves lower z_3 values than NSGA-II within a specific range of z_1 , indicating stronger performance in reducing restriction violations under moderate workload balance conditions. These comparisons highlight differences in the methods' abilities to optimize particular objective pairs in the larger problem instance.

Table 3.7 reports the payoff table obtained by NSGA-II for the United Air Lines instance. As with the Allegiant Air instance, the table provides the optimal value of each objective when optimized individually. When z_1 is minimized, it attains a value of 328,947, with z_2 equal to 2,567, no workstation restriction violations, and a similarity score of 4,085.

Table 3.7: Payoff table obtained by NSGA-II for the instance United Air Lines Jan 8 2019.

	z_1	z_2	z_3	z_4
min z_1	328,947	3,725	4	2,575
min z_2	385,326	2,567	1	2,693
min z_3	377,970	3,018	0	3,744
max z_4	385,326	3,679	1	4,085

The corresponding values of the other objectives, when each objective is optimized individually, highlight the trade-offs involved. For instance, minimizing z_3 to zero results in an ADP of 377,970 and a similarity score of 3,744. These outcomes illustrate the conflicts

between objectives and demonstrate the necessity of a multi-objective approach to identify balanced compromise solutions.

A comparison between the results of the Allegiant Air and United Air Lines instances highlights the increasing complexity and computational challenges associated with larger problem sizes. While the small instance allowed for comprehensive exploration by multiple methods and confirmed the accuracy of NSGA-II approximations, exact methods were unable to find meaningful solutions for the larger instance, with optimality gaps exceeding 95% after three hours of runtime. NSGA-II produced a diverse set of non-dominated solutions for both instances and, based on its performance in the small instance, is expected to approximate the Pareto front well for larger instances. The trade-offs between objectives appear more pronounced in the larger problem, with more extreme objective values and sharper conflicts observed. These results underscore the importance of scalable, approximation-based solution methods capable of maintaining solution diversity and quality as problem size and complexity increase.

3.5.4 Multi-objective Analysis on Benchmark Instances

The scalability and performance of NSGA-II are analyzed on the R50 and R100 benchmark configurations. These configurations include instances with varying numbers of workstations, allowing for an examination of how solution quality, objective trade-offs, and Pareto front structure change with resource availability.

Tables 3.8 and 3.9 present Pareto front statistics obtained by NSGA-II for the R50 and R100 instances, respectively. Each table includes the mean, standard deviation, and range of Pareto front sizes across different numbers of workstations. In addition to front size statistics, the tables report normalized standard deviation metrics for each objective. The objective values are first normalized to eliminate the effect of different scales, and the standard deviation of these normalized values is then computed to measure the spread of each objective across the Pareto front. The tables show the mean normalized standard deviation for each objective, averaged over instances, as well as the standard deviation of these values across instances.

For the R50 instances, the mean Pareto front size generally decreases as the number of workstations increases, declining from 8,558 at $|K| = 6$ to 4,926 at $|K| = 50$. This trend indicates that increasing the number of workstations results in fewer solutions identified by the algorithm within the fixed runtime. It may also reflect that a larger number of workstations restricts the objective space.

Table 3.8: Pareto front statistics of the solutions obtained by NSGA-II for R50 instances.

K	Inst.	Pareto Front Size			Normalized Std. Dev. of Objective Values							
		Mean	Std.	Range	Mean per objective				Std. dev. per objective			
					z_1	z_2	z_3	z_4	z_1	z_2	z_3	z_4
6	6	8,558	1,318	3,547	0.25	0.24	0.25	0.27	0.02	0.02	0.02	0.02
7	13	11,692	2,680	7,722	0.24	0.23	0.24	0.26	0.01	0.01	0.02	0.01
8	5	13,587	1,983	5,086	0.24	0.23	0.25	0.25	0.01	0.01	0.02	0.02
10	1	21,166	—	—	0.20	0.19	0.25	0.23	—	—	—	—
11	6	16,351	1,406	3,879	0.22	0.20	0.23	0.27	0.01	0.02	0.02	0.01
12	17	16,350	1,738	6,612	0.22	0.20	0.22	0.26	0.01	0.01	0.01	0.01
23	1	10,651	—	—	0.20	0.18	0.19	0.20	—	—	—	—
25	4	11,415	452	937	0.19	0.17	0.22	0.21	0.01	0.01	0.01	0.01
26	1	9,705	—	—	0.21	0.18	0.22	0.22	—	—	—	—
27	2	9,514	131	185	0.20	0.18	0.20	0.20	0.01	0.02	0.01	0.01
28	7	9,794	678	2,027	0.21	0.17	0.21	0.22	0.01	0.01	0.02	0.01
29	3	9,032	682	1,198	0.20	0.18	0.20	0.21	0.01	0.01	0.02	0.01
31	1	6,726	—	—	0.21	0.18	0.20	0.20	—	—	—	—
33	6	6,417	782	2,011	0.22	0.19	0.19	0.20	0.02	0.01	0.01	0.01
34	7	7,390	568	1,474	0.20	0.17	0.21	0.21	0.01	0.02	0.02	0.01
35	8	6,817	551	1,809	0.21	0.18	0.20	0.21	0.01	0.01	0.02	0.01
36	11	6,732	647	1,777	0.21	0.18	0.20	0.20	0.01	0.01	0.01	0.01
37	3	6,516	282	515	0.20	0.19	0.20	0.21	0.00	0.01	0.01	0.01
38	2	6,426	297	420	0.21	0.17	0.20	0.21	0.01	0.01	0.02	0.00
39	1	5,957	—	—	0.21	0.19	0.22	0.21	—	—	—	—
40	6	5,558	479	1,368	0.21	0.18	0.19	0.21	0.01	0.01	0.02	0.01
41	9	5,651	731	2,385	0.21	0.18	0.19	0.21	0.01	0.01	0.01	0.01
42	5	5,218	212	559	0.21	0.19	0.20	0.21	0.01	0.01	0.01	0.01
43	11	4,999	451	1,406	0.21	0.19	0.20	0.20	0.01	0.01	0.02	0.01
44	10	5,047	434	1,325	0.21	0.19	0.20	0.21	0.01	0.01	0.01	0.01
45	5	5,455	309	807	0.21	0.18	0.19	0.21	0.01	0.01	0.01	0.01
47	7	4,982	532	1,626	0.20	0.18	0.20	0.21	0.01	0.02	0.02	0.01
48	4	4,970	758	1,770	0.20	0.18	0.20	0.20	0.01	0.01	0.01	0.02
49	9	4,913	367	883	0.21	0.19	0.20	0.21	0.01	0.02	0.01	0.01
50	13	4,926	638	2,597	0.21	0.18	0.20	0.21	0.02	0.01	0.01	0.01
51	11	4,715	449	1,431	0.21	0.18	0.20	0.21	0.01	0.01	0.01	0.01
52	1	4,483	—	—	0.20	0.18	0.21	0.21	—	—	—	—
53	1	8,273	—	—	0.22	0.17	0.19	0.21	—	—	—	—
54	1	7,179	—	—	0.20	0.18	0.19	0.20	—	—	—	—
55	3	7,236	551	967	0.20	0.19	0.19	0.21	0.02	0.00	0.01	0.00
56	5	8,323	188	464	0.20	0.19	0.18	0.22	0.01	0.01	0.00	0.01
57	3	7,796	251	497	0.21	0.18	0.18	0.20	0.01	0.01	0.01	0.01
58	2	7,367	443	626	0.21	0.19	0.18	0.22	0.00	0.01	0.01	0.00
59	4	6,597	278	623	0.22	0.18	0.18	0.21	0.01	0.01	0.01	0.00
60	14	6,906	398	1,386	0.21	0.18	0.19	0.21	0.01	0.01	0.01	0.01
67	1	5,872	—	—	0.20	0.18	0.18	0.22	—	—	—	—
79	1	4,771	—	—	0.20	0.18	0.18	0.21	—	—	—	—
Overall	231	7,905	765	2,314	0.21	0.19	0.20	0.22	0.01	0.01	0.01	0.01

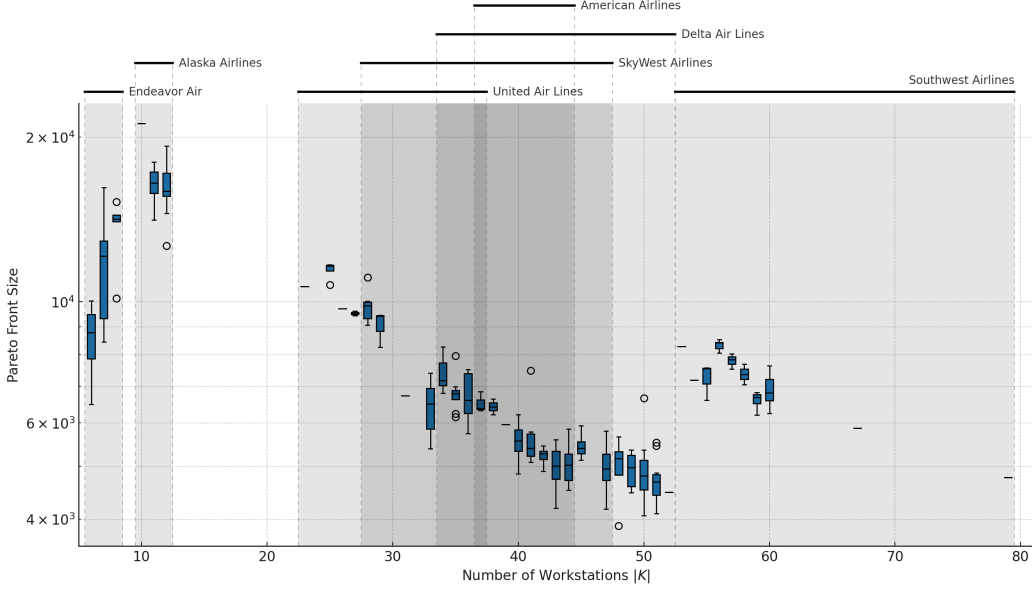


Figure 3.8: Distribution of Pareto front sizes obtained by NSGA-II in R50 instances, intervals highlight the carriers of instances.

Particularly, the mean front size is 21,166 at $|K| = 10$. This peak followed by a consistent downward trend for larger $|K|$, may suggest that the allocated runtime of 300 seconds is not sufficient for effectively exploring the solution space in instances larger than $|K| = 10$ for R50. Another anomaly is at $|K| = 53$, where the Pareto front sizes nearly double relative to earlier values. As Figure 3.8 illustrates, these breakpoints correlate with different carriers' workstation count intervals.

Instances beyond this threshold exclusively represent Southwest Airlines, which is the sole carrier within the benchmark set operating more than 2,650 flights. As presented in Table 3.3, Southwest Airlines instances average 2,965 flights but only 299 time points, whereas Delta Air Lines instances, despite fewer flights, average 1,262 time points. The fewer time points in Southwest Airlines instances reflect a more structured scheduling approach, characterized by departures and arrivals clustered around rounded intervals. This decreases complexity and contributes to larger Pareto front sizes, suggesting that structured scheduling can simplify the optimization tasks downstream. The flight schedules also play a significant role in determining how problem difficulty scales, with sparser distributions leading to faster increases in complexity.

The normalized standard deviation of objective values provides insight into the vari-

ability of each objective across the Pareto fronts. For most values of $|K|$, mean normalized standard deviations for z_1 , z_2 , z_3 , and z_4 are similar, generally ranging from 0.17 to 0.27. Slightly elevated values for z_4 appear in several instances, indicating greater variability in similarity scores among non-dominated solutions. The standard deviations of these metrics across different instances remain low, typically around 0.01 or 0.02, demonstrating the consistent capability of NSGA-II to explore similar ranges of objectives across varying problem instances.

Table 3.9: Pareto front statistics of the solutions obtained by NSGA-II for R100 instances.

$ K $	Inst.	Pareto Front Size			Normalized Std. Dev. of Objective Values							
		Mean	Std.	Range	Mean per objective				Std. dev. per objective			
					z_1	z_2	z_3	z_4	z_1	z_2	z_3	z_4
3	19	7,591	1,725	6,683	0.27	0.24	0.25	0.27	0.02	0.01	0.02	0.02
4	5	9,620	1,172	2,794	0.26	0.23	0.25	0.26	0.02	0.01	0.02	0.02
5	7	16,059	1,246	3,744	0.22	0.20	0.22	0.25	0.01	0.01	0.02	0.01
6	17	16,355	2,199	7,450	0.22	0.19	0.22	0.25	0.01	0.01	0.01	0.01
11	1	14,967	—	—	0.21	0.19	0.23	0.21	—	—	—	—
12	4	11,748	891	1,972	0.21	0.18	0.20	0.22	0.01	0.01	0.01	0.00
13	3	11,732	2,333	4,567	0.21	0.17	0.22	0.20	0.01	0.00	0.02	0.00
14	10	10,559	648	2,251	0.21	0.18	0.21	0.21	0.01	0.01	0.02	0.01
15	1	7,781	—	—	0.22	0.19	0.18	0.22	—	—	—	—
16	6	7,243	558	1,576	0.22	0.19	0.19	0.21	0.01	0.01	0.02	0.01
17	15	7,689	950	3,226	0.22	0.18	0.21	0.20	0.01	0.01	0.01	0.01
18	14	7,498	615	2,119	0.21	0.18	0.21	0.21	0.01	0.01	0.01	0.01
19	3	6,214	1,692	3,147	0.21	0.17	0.21	0.21	0.00	0.02	0.03	0.01
20	15	6,251	667	2,510	0.21	0.18	0.21	0.20	0.01	0.01	0.01	0.01
21	16	5,697	508	2,168	0.21	0.18	0.20	0.21	0.01	0.01	0.01	0.01
22	15	5,553	564	1,911	0.21	0.18	0.20	0.20	0.01	0.01	0.01	0.01
23	7	5,200	639	1,528	0.21	0.17	0.21	0.21	0.01	0.01	0.02	0.02
24	13	5,209	648	2,046	0.21	0.18	0.20	0.21	0.01	0.01	0.01	0.01
25	24	5,247	639	2,313	0.21	0.18	0.20	0.21	0.01	0.01	0.01	0.01
26	2	5,990	1,149	1,625	0.21	0.19	0.21	0.22	0.01	0.01	0.02	0.01
27	4	7,147	424	860	0.21	0.17	0.19	0.20	0.01	0.01	0.02	0.01
28	8	6,516	786	2,368	0.22	0.18	0.18	0.22	0.02	0.01	0.01	0.01
29	6	6,713	594	1,537	0.21	0.18	0.19	0.20	0.01	0.01	0.01	0.01
30	14	6,046	483	1,625	0.21	0.18	0.19	0.21	0.01	0.01	0.01	0.01
33	1	5,000	—	—	0.20	0.18	0.18	0.21	—	—	—	—
39	1	3,696	—	—	0.21	0.19	0.20	0.21	—	—	—	—
Overall	231	7,793	903	2,971	0.22	0.19	0.21	0.22	0.01	0.01	0.01	0.01

For the R100 instances, the same general pattern observed in R50 persists, where the largest Pareto fronts are found at smaller values of $|K|$ and a downward trend follows as

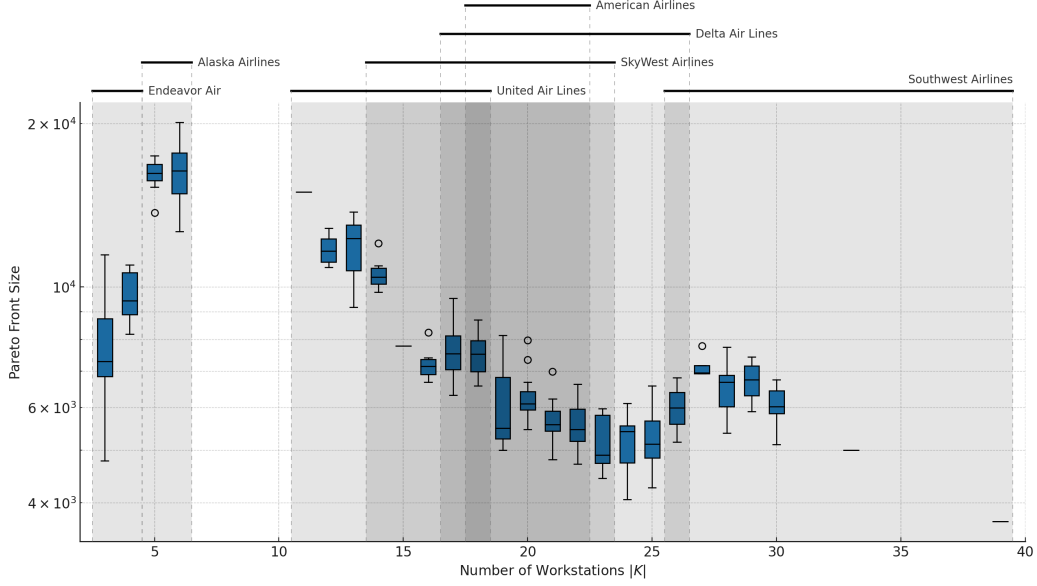


Figure 3.9: Distribution of Pareto front sizes obtained by NSGA-II in R100 instances, intervals highlight the carriers of instances.

$|K|$ increases. As shown in Figure 3.9, anomalies in this trend again align with changes in carrier composition. Compared to R50, the Pareto fronts are slightly smaller, which is possibly due to the reduced solution space as a result of having fewer workstations available. The normalized standard deviation of objective values shows similar behavior to that observed in R50. Mean normalized standard deviations for z_1 , z_2 , z_3 , and z_4 generally fall between 0.17 and 0.27. Slightly higher values for z_1 are observed in some configurations, suggesting greater variability in workload balance when resource constraints are tighter. The standard deviations of these metrics across instances remain low, typically around 0.01 or 0.02, demonstrating that NSGA-II consistently explores similar objective ranges across different problem instances.

Hypervolume analysis was conducted on selected three-objective subsets of the Pareto fronts obtained by NSGA-II for the R50 and R100 instances. The analysis is limited to three objectives at a time due to the substantial computational requirements of hypervolume calculations in four dimensions. Tables 3.10 and 3.11 present the mean (M), standard deviation (S), and range (R) of Pareto front sizes and corresponding hypervolume values for the four possible three-objective combinations: (z_1, z_2, z_3) , (z_1, z_2, z_4) , (z_1, z_3, z_4) , and (z_2, z_3, z_4) . Hypervolume is computed as the proportion of the objective space dominated by the Pareto front relative to the nadir point. Higher hypervolume values indicate Pareto

fronts that are closer to the ideal front, with better coverage and diversity.

Table 3.10: Summary of NSGA results for R50 instances for different multi-objective sets.

$ K $	Inst.	Pareto Front Size												Hypervolume											
		(z_1, z_2, z_3)			(z_1, z_2, z_4)			(z_1, z_3, z_4)			(z_2, z_3, z_4)			(z_1, z_2, z_3)			(z_1, z_3, z_4)			(z_1, z_3, z_4)			(z_2, z_3, z_4)		
		M	S	R	M	S	R	M	S	R	M	S	R	M	S	R	M	S	R	M	S	R	M	S	R
6	6	33	11	30	2,389	146	386	2,388	179	471	1,086	284	620	0.23	0.10	0.27	0.36	0.01	0.02	0.50	0.02	0.05	0.55	0.01	0.03
7	13	40	13	52	2,990	363	1,350	2,589	327	1,109	1,285	338	1,015	0.29	0.17	0.51	0.35	0.02	0.05	0.49	0.02	0.07	0.54	0.02	0.07
8	5	37	7	19	3,032	397	1,040	2,670	287	590	1,367	279	709	0.24	0.25	0.57	0.34	0.02	0.04	0.49	0.02	0.05	0.52	0.02	0.05
10	1	343	—	—	5,006	—	—	3,470	—	—	957	—	—	0.52	—	—	0.43	—	—	0.51	—	—	0.66	—	—
11	6	377	59	145	4,237	304	757	3,856	296	849	1,587	253	741	0.35	0.05	0.13	0.42	0.02	0.05	0.49	0.02	0.07	0.65	0.03	0.07
12	17	511	198	775	4,350	427	1,676	3,822	435	1,961	1,518	408	1,351	0.40	0.07	0.23	0.41	0.02	0.08	0.48	0.02	0.09	0.64	0.03	0.10
23	1	802	—	—	3,435	—	—	1,883	—	—	509	—	—	0.47	—	—	0.31	—	—	0.41	—	—	0.28	—	—
25	4	746	92	224	4,437	184	432	1,670	340	707	582	323	687	0.47	0.03	0.06	0.36	0.03	0.07	0.46	0.06	0.15	0.36	0.03	0.06
26	1	861	—	—	4,065	—	—	1,662	—	—	764	—	—	0.42	—	—	0.35	—	—	0.43	—	—	0.33	—	—
27	2	852	56	79	3,834	292	413	1,770	329	465	378	139	196	0.41	0.05	0.07	0.34	0.01	0.01	0.47	0.06	0.08	0.31	—	0.01
28	7	634	228	677	3,829	462	1,319	1,576	352	1,092	527	90	287	0.41	0.08	0.25	0.38	0.04	0.13	0.46	0.03	0.10	0.35	0.08	0.22
29	3	702	95	177	3,634	540	1,051	1,493	224	444	395	78	136	0.44	0.05	0.10	0.38	0.02	0.03	0.45	0.02	0.04	0.36	0.03	0.06
31	1	213	—	—	2,287	—	—	2,211	—	—	370	—	—	0.23	—	—	0.25	—	—	0.45	—	—	0.22	—	—
33	6	274	65	179	2,671	545	1,406	1,580	170	427	424	104	285	0.23	0.04	0.12	0.38	0.02	0.07	0.49	0.01	0.04	0.27	0.04	0.10
34	7	582	239	624	3,132	385	1,178	1,725	167	497	341	114	337	0.36	0.04	0.10	0.34	0.05	0.14	0.45	0.04	0.14	0.27	0.07	0.20
35	8	390	179	481	2,734	329	839	1,574	147	458	446	123	326	0.30	0.09	0.28	0.36	0.03	0.09	0.46	0.03	0.11	0.24	0.04	0.12
36	11	478	186	463	2,701	489	1,514	1,694	157	537	341	122	366	0.32	0.08	0.23	0.34	0.04	0.14	0.45	0.05	0.17	0.25	0.03	0.13
37	3	782	136	246	3,029	197	380	1,328	57	99	233	29	58	0.39	0.08	0.16	0.37	0.05	0.10	0.44	0.08	0.16	0.27	0.07	0.12
38	2	670	236	334	2,606	217	307	1,476	38	54	189	52	74	0.36	—	—	0.41	0.01	0.01	0.51	0.03	0.05	0.24	0.05	0.06
39	1	600	—	—	2,430	—	—	1,404	—	—	223	—	—	0.38	—	—	0.37	—	—	0.52	—	—	0.30	—	—
40	6	495	83	224	2,544	312	869	1,198	126	346	242	41	114	0.37	0.04	0.10	0.36	0.02	0.05	0.47	0.04	0.11	0.27	0.06	0.15
41	9	501	53	130	2,418	203	530	1,408	244	788	238	110	354	0.35	0.03	0.09	0.36	0.03	0.09	0.48	0.03	0.08	0.24	0.08	0.29
42	5	521	83	207	2,366	128	336	1,226	204	552	215	36	93	0.35	0.05	0.11	0.35	0.04	0.12	0.48	0.04	0.10	0.23	0.06	0.17
43	11	504	104	319	2,321	175	580	1,163	160	484	235	65	202	0.34	0.05	0.15	0.37	0.03	0.08	0.45	0.06	0.22	0.24	0.04	0.11
44	10	477	70	198	2,287	209	705	1,191	88	273	246	83	254	0.36	0.05	0.15	0.36	0.03	0.08	0.42	0.05	0.15	0.25	0.03	0.11
45	5	429	129	324	2,479	243	629	1,284	119	289	319	99	256	0.32	0.04	0.10	0.37	0.02	0.06	0.44	0.03	0.07	0.25	0.07	0.14
47	7	505	184	531	2,409	304	872	1,210	178	451	238	90	266	0.30	0.06	0.18	0.35	0.04	0.11	0.44	0.04	0.13	0.24	0.05	0.14
48	4	405	60	145	2,260	166	378	1,069	159	352	219	56	123	0.34	0.02	0.04	0.38	0.02	0.04	0.45	0.03	0.07	0.24	0.06	0.13
49	9	465	72	225	2,329	98	310	1,153	191	559	260	61	168	0.35	0.04	0.13	0.35	0.02	0.08	0.42	0.05	0.15	0.22	0.05	0.15
50	13	465	59	191	2,367	265	1,056	1,126	150	537	225	39	127	0.34	0.03	0.12	0.35	0.03	0.09	0.42	0.03	0.10	0.22	0.04	0.12
51	11	500	78	287	2,227	218	603	1,175	101	346	236	83	270	0.32	0.04	0.12	0.36	0.03	0.09	0.42	0.04	0.11	0.21	0.04	0.12
52	1	593	—	—	2,167	—	—	984	—	—	241	—	—	0.30	—	—	0.31	—	—	0.36	—	—	0.20	—	—
53	1	390	—	—	3,680	—	—	2,223	—	—	477	—	—	0.20	—	—	0.37	—	—	0.43	—	—	0.23	—	—
54	1	250	—	—	3,483	—	—	2,279	—	—	460	—	—	0.19	—	—	0.39	—	—	0.48	—	—	0.23	—	—
55	3	404	74	138	3,198	423	839	2,145	159	313	459	93	171	0.23	0.03	0.05	0.37	0.05	0.10	0.48	0.02	0.04	0.23	0.04	0.07
56	5	312	99	238	3,579	287	727	2,154	83	195	568	177	439	0.19	0.02	0.04	0.36	0.05	0.11	0.44	0.05	0.12	0.24	0.02	0.04
57	3	404	75	144	3,184	301	544	2,203	51	91	445	138	250	0.23	0.02	0.04	0.40	0.01	0.02	0.47	0.01	0.02	0.22	0.05	0.11
58	2	391	136	192	3,076	366	518	2,219	64	90	694	124	175	0.22	—	—	0.28	0.02	0.02	0.45	0.03	0.04	0.24	0.06	0.08
59	4	353	66	159	2,905	276	646	2,012	225	510	499	115	280	0.22	0.05	0.11	0.38	0.03	0.08	0.45	0.04	0.10	0.21	0.06	0.12
60	14	342	54	172	3,055	207	742	1,985	197	684	421	79	315	0.23	0.04	0.11	0.38	0.03	0.08	0.45	0.03	0.09	0.23	0.04	0.15
67	1	337	—	—	2,626	—	—	1,689	—	—	402	—	—	0.23	—	—	0.40	—	—	0.45	—	—	0.21	—	—
79	1	336	—	—	2,343	—	—	1,384	—	—	365	—	—	0.24	—	—	0.36	—	—	0.40	—	—	0.21	—	—
Overall	231	433	98	282	2,925	283	847	1,844	195	620	554	138	401	0.32	0.06	0.17	0.37	0.03	0.08	0.46	0.03	0.10	0.32	0.04	0.12

For the R50 instances in Table 3.10, hypervolume values decrease as the number of workstations increases. This pattern aligns with the reduction in Pareto front sizes and indicates that with more workstations, the Pareto front becomes more constrained or is not explored as thoroughly within the fixed runtime. The largest hypervolumes are observed at lower $|K|$ values, where the smaller number of workstations results in a broader range of trade-offs. Across different objective subsets, combinations that include z_1 and z_4 , particularly (z_1, z_2, z_4) , yield higher hypervolume values. This indicates that including workload balance and similarity objectives leads to more diverse and well-distributed solutions. Subsets that exclude z_1 show consistently lower hypervolumes, suggesting that solutions without workload balance considerations span a narrower portion of the objective

space.

Table 3.11: Summary of NSGA results for R100 instances for different multi-objective sets.

$ K $	Inst.	Pareto Front Size												Hypervolume											
		(z_1, z_2, z_3)			(z_1, z_2, z_4)			(z_1, z_3, z_4)			(z_2, z_3, z_4)			(z_1, z_2, z_3)			(z_1, z_2, z_4)			(z_1, z_3, z_4)			(z_2, z_3, z_4)		
		M	S	R	M	S	R	M	S	R	M	S	R	M	S	R	M	S	R	M	S	R	M	S	R
3	19	62	23	72	1,989	288	982	2,367	561	2,433	1,234	308	1,281	0.50	0.18	0.69	0.38	0.03	0.11	0.50	0.02	0.11	0.54	0.01	0.06
4	5	60	16	36	2,791	194	465	2,360	408	990	1,305	467	1,115	0.25	0.16	0.39	0.35	0.02	0.05	0.49	0.03	0.07	0.52	0.02	0.04
5	7	606	192	588	4,491	512	1,382	3,584	365	888	941	396	1,234	0.48	0.09	0.26	0.44	0.02	0.06	0.53	0.01	0.03	0.63	0.05	0.15
6	17	618	233	781	4,766	642	2,215	3,782	438	1,876	1,155	387	1,468	0.46	0.11	0.44	0.43	0.02	0.05	0.52	0.01	0.04	0.64	0.03	0.10
11	1	1,020	—	—	5,525	—	—	3,717	—	—	666	—	—	0.47	—	—	0.31	—	—	0.42	—	—	0.38	—	—
12	4	919	199	368	4,315	646	1,391	2,259	307	710	643	99	232	0.47	0.01	0.02	0.33	0.03	0.07	0.43	0.05	0.10	0.30	0.03	0.07
13	3	814	268	490	4,456	448	874	2,330	236	471	650	446	837	0.46	0.04	0.08	0.32	0.01	0.01	0.39	0.04	0.08	0.30	0.05	0.11
14	10	850	266	864	4,098	590	2,107	2,011	329	1,023	583	179	515	0.43	0.06	0.22	0.34	0.03	0.12	0.44	0.05	0.15	0.32	0.07	0.21
15	1	421	—	—	2,626	—	—	1,986	—	—	664	—	—	0.42	—	—	0.33	—	—	0.49	—	—	0.27	—	—
16	6	278	35	87	2,762	283	757	1,963	380	1,000	566	98	287	0.28	0.03	0.07	0.33	0.05	0.14	0.48	0.04	0.10	0.26	0.06	0.14
17	15	541	338	1,003	3,074	473	1,735	1,908	308	1,098	464	173	600	0.35	0.07	0.21	0.33	0.05	0.15	0.46	0.05	0.17	0.27	0.06	0.22
18	14	684	302	831	3,141	430	1,334	1,775	210	819	314	136	431	0.35	0.06	0.22	0.35	0.04	0.12	0.46	0.06	0.21	0.26	0.07	0.25
19	3	720	130	251	2,929	390	779	1,448	364	682	187	29	52	0.40	0.01	0.02	0.40	0.01	0.02	0.54	0.05	0.10	0.26	0.09	0.16
20	15	661	93	368	2,786	348	1,103	1,438	223	702	208	68	214	0.39	0.04	0.11	0.37	0.04	0.13	0.47	0.06	0.20	0.25	0.06	0.22
21	16	689	97	361	2,629	223	961	1,382	170	509	279	99	336	0.36	0.04	0.16	0.37	0.04	0.12	0.45	0.04	0.14	0.20	0.03	0.10
22	15	611	179	592	2,478	285	873	1,324	173	599	240	96	337	0.35	0.03	0.12	0.36	0.04	0.14	0.45	0.05	0.17	0.22	0.05	0.20
23	7	589	112	270	2,432	382	904	1,320	198	483	232	99	282	0.36	0.08	0.24	0.35	0.05	0.16	0.44	0.04	0.10	0.19	0.08	0.20
24	13	600	135	518	2,371	259	884	1,170	166	581	254	95	380	0.38	0.03	0.10	0.35	0.03	0.11	0.45	0.04	0.14	0.22	0.06	0.19
25	24	632	117	492	2,428	240	912	1,273	206	843	240	86	338	0.36	0.05	0.16	0.35	0.05	0.23	0.43	0.06	0.27	0.21	0.06	0.23
26	2	566	139	197	2,657	396	560	1,349	369	522	525	167	236	0.37	0.09	0.13	0.37	0.05	0.07	0.42	0.10	0.14	0.26	0.05	0.07
27	4	475	84	183	3,183	182	401	1,777	188	393	481	126	273	0.28	0.01	0.02	0.40	0.01	0.02	0.48	0.02	0.05	0.23	0.04	0.10
28	8	403	114	318	2,905	476	1,511	1,881	231	792	507	104	315	0.28	0.06	0.19	0.35	0.06	0.16	0.45	0.05	0.14	0.23	0.04	0.14
29	6	470	46	103	3,114	205	591	1,640	81	212	383	59	156	0.28	0.03	0.08	0.40	0.04	0.12	0.48	0.03	0.09	0.23	0.04	0.10
30	14	407	62	221	2,595	361	1,134	1,587	255	893	430	116	371	0.27	0.05	0.14	0.36	0.05	0.16	0.46	0.04	0.15	0.23	0.04	0.13
33	1	428	—	—	2,547	—	—	1,220	—	—	348	—	—	0.23	—	—	0.42	—	—	0.42	—	—	0.20	—	—
39	1	372	—	—	1,879	—	—	1,151	—	—	233	—	—	0.26	—	—	0.39	—	—	0.45	—	—	0.22	—	—
Overall	231	548	146	461	2,975	360	1,161	1,885	274	958	524	164	554	0.37	0.06	0.22	0.37	0.04	0.12	0.46	0.04	0.14	0.31	0.05	0.16

For the R100 instances in Table 3.11, hypervolume values follow the same trend observed in R50, with higher values at lower workstation counts and a gradual decline as $|K|$ increases. Objective subsets that include z_1 and z_4 , particularly (z_1, z_2, z_4) , produce the largest hypervolumes, reinforcing that these combinations support more diverse solution spaces. Hypervolume values in R100 are generally lower than those in R50 at comparable workstation counts, consistent with the smaller Pareto front sizes observed in these instances. The variability in hypervolume values across instances also decreases as $|K|$ increases, which may reflect either more uniform solution structures or limitations in the solver’s ability to explore the solution space efficiently for larger problem instances within the fixed runtime.

These results demonstrate that NSGA-II is capable of producing diverse and well-distributed Pareto fronts, particularly when the number of workstations is low. However, as problem complexity increases or the number of workstations grows, both solution diversity and coverage tend to decrease, likely due to the algorithm’s limited ability to explore larger and more constrained objective spaces within the fixed runtime. The patterns observed in the hypervolume metrics are consistent with trends in Pareto front sizes and objective variability, highlighting that the number of workstations and scheduling structure play a key role in shaping the complexity and trade-off richness of the problem.

3.5.5 Sensitivity Analysis of Operational Assumptions

To assess the performance of the solution methodology and evaluate the impact of modeling assumptions on solution quality and trade-off structures, a comprehensive sensitivity analysis is conducted on the Allegiant Air instance introduced in Section 3.5.2.

For the workload balance objective, two formulations are evaluated: the absolute deviation penalty (ADP), which minimizes deviations from the average workload, and a peak workload minimization objective, which reduces the maximum workload across workstations. The timing conflict rule, which prohibits flights with departures within a five-minute window from being assigned to the same workstation, is tested under a more restrictive ten-minute window. Additionally, the long-haul separation rule, applied to flights exceeding 3.5 hours for this problem instance due to the relatively shorter flight durations, is evaluated with a reduced threshold of 2.5 hours, as well as in a scenario where no such separation is enforced. In all cases, scenarios without these restrictions are also considered.

Workstation assignment restrictions are adapted to the two-workstation setup using several alternative rules. First, stricter traffic-based rules are considered by assigning one workstation to flights associated with the airports with the highest and another to those with the lowest traffic volumes, considering the top and bottom 3 or 5 airports. In case of ties, the traffic stats in the full instance are used. Second, region-based rules are introduced, where one workstation is assigned flights originating from the westernmost 25% or 50% of flights in the instance (ranked by longitude), with the other workstation assigned the corresponding easternmost portions. In both cases, scenarios without these restrictions are also evaluated. These variations aim to test the impact of geo-spatial and traffic-based segmentation on solution diversity and objective trade-offs. Finally, the similarity score calculation is tested with buffer sizes of 50, 150, and 300 kilometers to assess how changes in the definition of geo-spatial overlap influence the Pareto front structure.

Table 3.12 summarizes the configuration settings and their sizes. The size in this context refers to the number of restrictions or preferences implied by the configuration in the objective, and max size is the largest size that a configuration can take. Long-haul configuration has the largest impact on the objective, with 2.5h separation setting imposing restrictions on about 40% of all flight pairs. Departure proximity configurations remain relatively less impactful as they impose restrictions on less than 3% of the pairs. Workstation assignment restrictions based on airport traffic and longitude ranking are moderately sized. Similarity score configurations cover a wide range, with the largest tested buffer size of 300 kilometers containing non-zero scores between 61% of all pairs. Among the non-zero similarity scores, the average is 12%, 21%, and 31%, with standard deviations of 17%, 27%, and 31%, respectively, for 50, 150, and 300 km buffer radii.

Table 3.12: Overview of the configurations tested.

Objective	Configuration	Tested Value	Size	% of Max Size
Flight-Flight Restrictions	Long-haul Separation	2.5h	496	40.49
		3.5h	15	1.22
	Departure Proximity	5min	15	1.22
		10min	29	2.37
Workstation-Flight Restrictions	Airport Traffic	Top/Bottom 3	17	34.0
		Top/Bottom 5	26	52.0
	Longitude Ranking	25%	24	48.0
		50%	50	100.0
Similarity Score		50 km	477	39.02
		150 km	664	54.29
		300 km	749	61.22

These settings resulted in a total of 486 different configurations to test. NSGA-II is used as the solution method for its ability in capturing a large portion of the Pareto front as well as smaller computation time compared to the solver. The NSGA-II implementation was run with a fixed iteration limit of 3,000. Tables 3.13 and 3.14 present summaries for the algorithm time and the final Pareto front size per category.

Table 3.13 indicates that the average solution time across all configurations is approximately 9.2 minutes, with a minimum of 6.4 minutes and a maximum of 13.5 minutes. The standard deviation is 1.3 minutes, suggesting moderate variability across the tested configurations. Among the categories, airport traffic restrictions with the top/bottom 5 setting exhibit the highest average computation time with 9.8 minutes, whereas the absence of a long-haul separation rule yields the lowest average with 8.8 minutes. Workload balance settings also influence runtime: the absolute deviation balance (ADP) averages 9.6 minutes, while the peak workload balance (PWP) requires only 8.8 minutes on average. Although these differences show that adding or tightening certain rules can raise the computational complexity, the overall range in average times remains relatively narrow, indicating that while rule selection has a noticeable effect on solution difficulty, it does not introduce drastic computational difficulties for this problem instance on NSGA-II.

Table 3.14 shows that the average Pareto front size across all configurations is 566 solutions, with a minimum of 7 solutions and a maximum of 1,811 solutions. This wide range underscores the variability in solution diversity that arises from different configurations. Among the analyzed factors, workload balance using the absolute deviation penalty (ADP) yields the largest average Pareto front of 789 solutions, whereas the peak workload minimization (PWP) setting results in a much lower average of 343 solutions. The de-

Table 3.13: Algorithm run time (s) per configuration category.

Category	Tested Value	Min.	Avg.	Max.	Std. Dev.
Workload Balance	ADP	445	577	812	79
	PWP	383	529	744	74
Long-haul Separation	2.5h	472	582	812	75
	3.5h	383	549	792	78
	None	423	527	746	79
Departure Proximity	5min	430	559	741	79
	10min	438	569	812	85
	None	383	531	744	73
Airport Traffic	Top/Bottom 3	430	545	781	75
	Top/Bottom 5	429	586	812	82
	None	383	528	733	73
Longitude Ranking	25%	417	564	792	79
	50%	383	544	812	81
	None	429	551	746	80
Similarity Score	50 km	428	560	746	82
	150 km	383	547	812	77
	300 km	423	553	792	82
Overall		383	553	812	80

departure proximity constraint at a 10-minute threshold also produces a notably large mean Pareto front (765), compared to the 5-minute threshold (587) or no proximity constraint (347). Similar trends appear in workstation assignment categories, where restricting to the top/bottom 3 or 5 airports (or the westernmost 25% of flights) generally leads to larger Pareto sets than more relaxed options. Likewise, the 300 km similarity buffer exhibits a higher average number of non-dominated solutions compared to smaller distances, suggesting that broader definitions of geo-spatial overlap introduce more trade-offs for the optimizer to balance. Standard deviations remain substantial across categories, ranging from 280 to 420 solutions, indicating that the combined effects of constraints and objective functions substantially impact the breadth of the solution space.

Figures 3.10 and 3.11 provide violin plots of the distributions underlying the average values in Tables 3.13 and 3.14, respectively. Dots and dashes correspond to median and averages of the distributions, respectively. These visualizations confirm that the ADP setting not only leads to higher average algorithm times but also exhibits greater variability, as indicated by the wider spread in the violin plot. In contrast, PWP remains comparatively less variable, with fewer extreme outliers.

Figure 3.12 and Table 3.15 compares each category’s Pareto front through Hypervol-

Table 3.14: Pareto front size statistics per configuration category.

Category	Tested Value	Min.	Avg.	Max.	Std. Dev.
Workload Balance	ADP	58	789	1811	356
	PWP	7	343	1148	280
Long-haul Separation	2.5h	26	588	1532	372
	3.5h	10	582	1811	402
	None	7	528	1463	394
Departure Proximity	5min	30	587	1516	365
	10min	45	765	1811	393
	None	7	347	1354	285
Airport Traffic	Top/Bottom 3	46	633	1811	373
	Top/Bottom 5	69	732	1571	377
	None	7	333	1175	299
Longitude Ranking	25%	20	724	1811	420
	50%	46	543	1463	374
	None	7	431	1069	311
Similarity Score	50 km	8	534	1588	356
	150 km	10	575	1532	395
	300 km	7	589	1811	415
Overall		7	566	1811	390

ume measure, which is derived by comparing the Hypervolume of the fronts obtained by individual configurations to the overall front obtained by all configurations under particular category values. For instance, the first row corresponds to the configurations with long-haul separation of 2.5 hours, computing the Hypervolume ratio as the ratio of the volume between the front obtained by 2.5h to the overall front obtained by 3.5h, 2.5h, and no separation rule. Each distribution placed along the x-axis corresponding to configurations with the second category indicated on the x-axis. The no-rule categories are omitted from these figures as they form the best front in about all of the configurations, forming narrower distributions around 95-99%, difficult to draw comparisons to others. Overall, the shape and spread of each distribution show that more rules generally result in a , maintaining relatively high HV coverage without overly fragmenting the solution space.

Shape of the distributions across different configuration settings indicate how one particular configuration impacts the efficient solutions and the solution space. In the remainder of this section, we analyze the Pareto front obtained in different configurations.

Figures 3.13–3.19 present how specific parameter combinations change the shape and distribution of solutions in the projection of the Pareto front on workload balance and similarity score objectives. Figure 3.13 illustrate a somewhat expected result. ADP so-

Table 3.15: Average Hypervolume ratio measures across categories.

Category 1		Lh. Sep.		Dep. Prx.		Ap. Trf.		Lng. Rk.	
Category 2	Level	2.5	3.5	5	10	3	5	25	50
Wd. Bal.	ADP	78%	89%	85%	83%	85%	80%	84%	80%
	PWP	76%	88%	84%	82%	83%	79%	83%	76%
Lh. Sep.	None			88%	85%	86%	82%	87%	81%
	2.5	77%		77%	77%	78%	73%	77%	70%
	3.5		89%	88%	86%	87%	83%	87%	83%
Dep. Prx.	None	76%	91%			85%	80%	85%	78%
	5	77%	88%	84%		84%	79%	83%	78%
	10	77%	86%		83%	82%	78%	83%	77%
Ap. Trf.	None	80%	95%	90%	89%			91%	89%
	3	78%	87%	84%	82%	84%		82%	77%
	5	73%	83%	79%	78%		79%	78%	68%
Lng. Rk.	None	84%	95%	91%	90%	93%	92%		
	25%	77%	87%	83%	83%	82%	78%	84%	
	50%	70%	83%	78%	77%	77%	68%		78%
Sim. Sc.	150	77%	89%	84%	82%	84%	79%	83%	78%
	300	77%	88%	84%	83%	83%	78%	83%	77%
	50	77%	89%	85%	84%	85%	80%	85%	78%

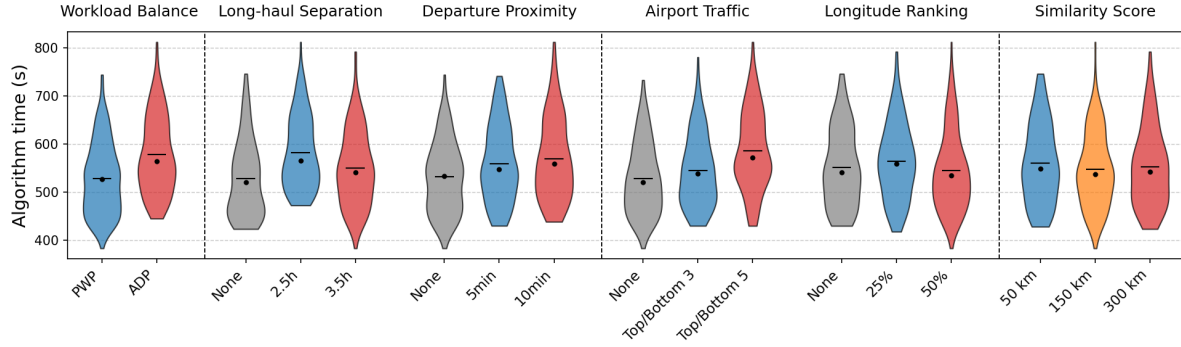


Figure 3.10: Algorithm time distributions across tested categories.

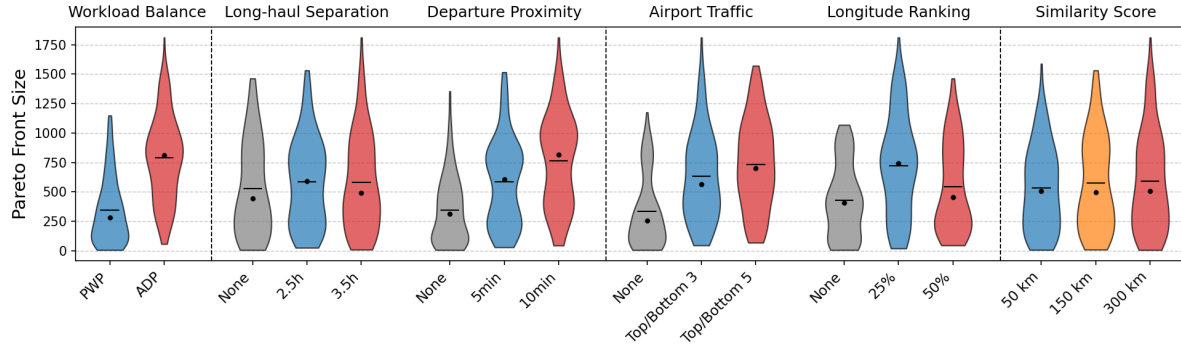


Figure 3.11: Pareto front size distributions across tested categories.

lutions perform generally worse in PWP. However, PWP solutions able to obtain slightly better similarity score but with a huge tradeoff for ADP. This is likely due to PWP having a better trade-off with similarity score at higher values as indicated by the shape of the fronts. Figure 3.14 supports this pattern, also showing that ADP may fall behind and miss a significant portion of the efficient solutions while exploring the extreme solutions. Figure 3.15 shows an example where ADP and PWP perform relatively better at the opposite ends of their respective fronts.

In terms of similarity score, the results are generally similar, while the level currently being optimized provides the better performing front in most cases, there are still exceptions. Figure 3.16 shows an example where 300 km buffer radius performs consistently worse than 50 km and 150 km settings, with introduction of longitude ranking of 50% producing a more significant difference. These results are not specific to 300 km as Figure 3.17



Figure 3.12: Comparison of Hypervolume ratio measures across categories.

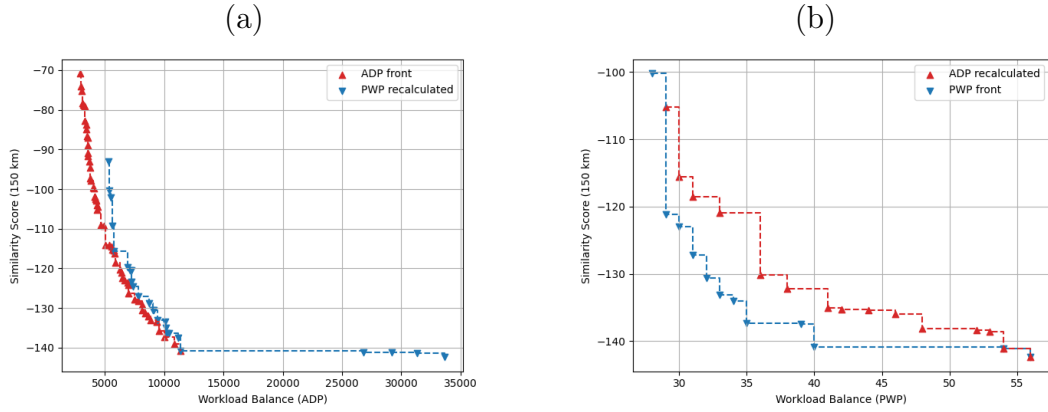


Figure 3.13: Comparison of (a) ADP and (b) PWP solutions for (2.5h, 5min, 3 airports, 25%, 150km) configuration.

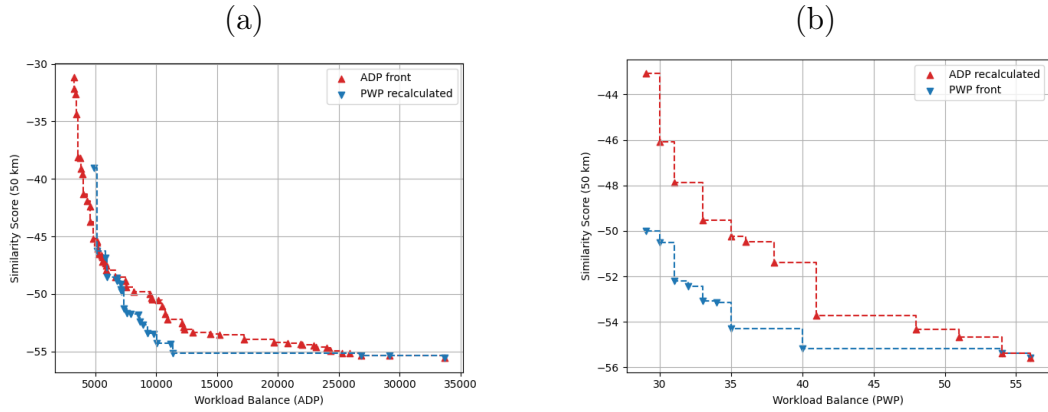


Figure 3.14: Comparison of (a) ADP and (b) PWP solutions for (2.5h, 5min, 5 airports, 25%, 50km) configuration.

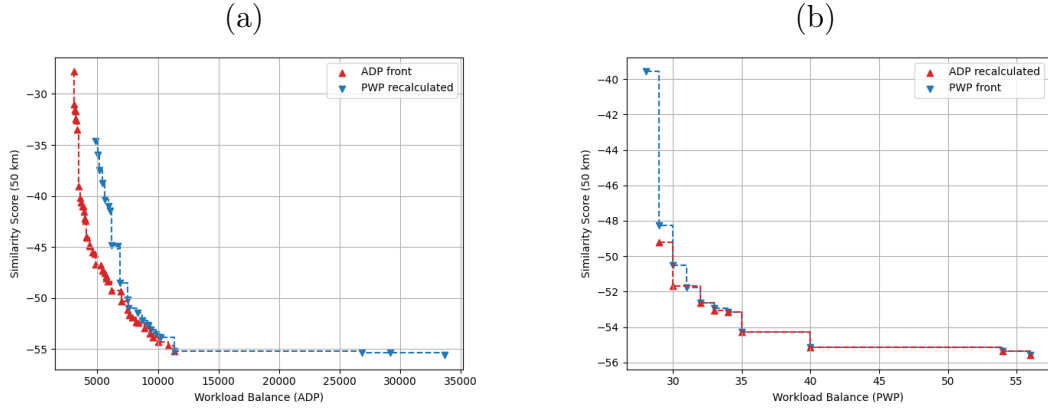


Figure 3.15: Comparison of (a) ADP and (b) PWP solutions for (2.5h, 5min, 3 airports, 25%, 50 km) configuration.

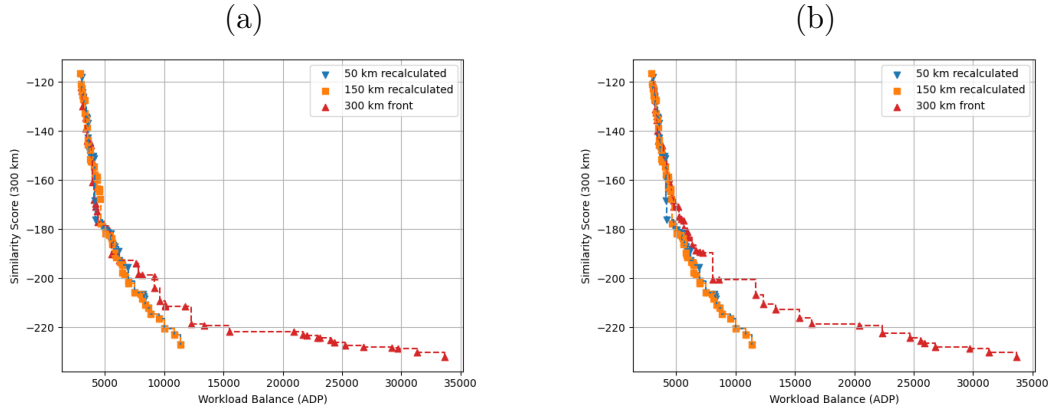


Figure 3.16: Comparison of 50-150-300 km solutions on 300 km similarity score objective for (5min, 5 airports, (a) 0% and (b) 50%, 300 km, ADP) configurations.

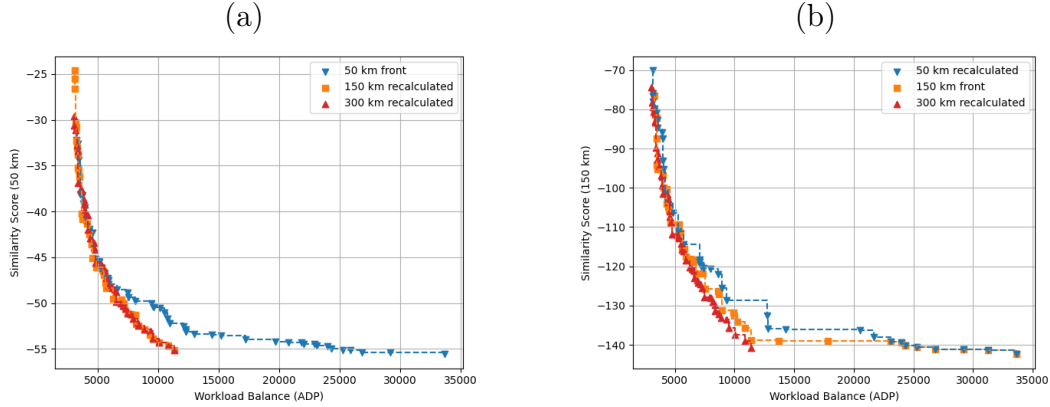


Figure 3.17: Comparison of 50-150-300 km solutions on (a) 50 km and (b) 150 km similarity score objective for (5min, 5 airports, 25%, ADP) configurations.

shows.

In terms of other objectives, introduction of additional rules may sometimes yield better results. Figure 3.18, for example, shows that introduction of long-haul separation constraints yielding a better front at certain intervals, with results improving with more strict separation limits. This result is well documented in the literature of multi-objective evolutionary heuristics under the use of auxiliary or helper objectives, which work as pivot points when exploring the solution space, consequently making difficult regions easier to reach (Jensen, 2004; Buzdalova, Matveeva, and Korneev, 2015). One possible explanation for this relation between long-haul separation and similarity scores in this case is that separated flights may follow similar routes at differing levels. This may introduce a correlation between the two objectives in terms of assigning these flights together, helping the exploration process to improve them consecutively using each other as anchor points.

One of the most prominent result is the steep decrease in the ADP performance as the similarity score increases. This often results in algorithm to explore parts with lower ADP as solutions are more concentrated in that region. Figure 3.19 illustrate two of such examples, also showing that more restrictive rules do not always act as auxiliary objectives in exploring others.

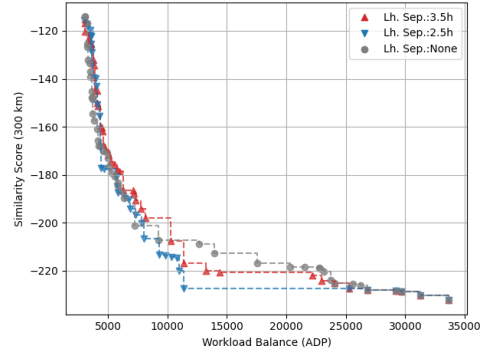


Figure 3.18: Comparison of long-haul separation solutions for (5min, 3 airports, 25%, 300 km, ADP) configuration.

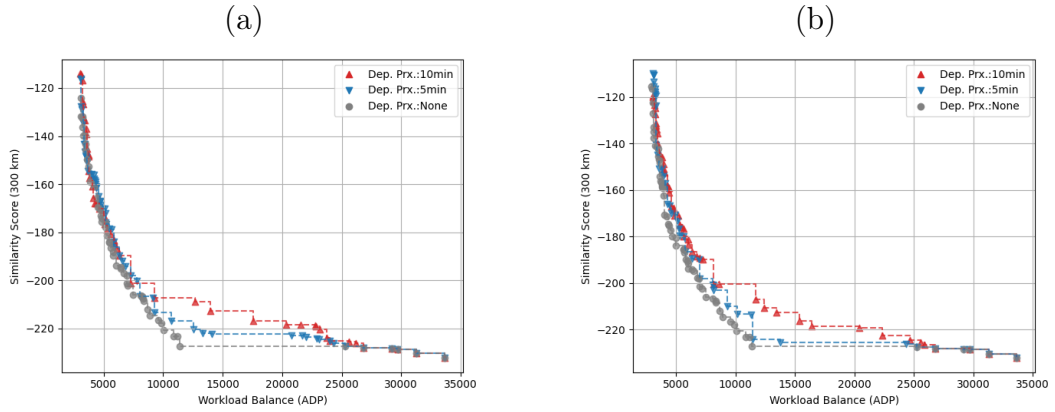


Figure 3.19: Comparison of departure proximity solutions for (3 airports, (a) 25% and (b) 50%, 300 km, ADP) configuration.

3.6 Conclusions

This chapter has introduced a multi-objective framework for flight-dispatcher assignment that addresses workload imbalance, operational restrictions, and flight-path similarity. By formulating the problem to include both flight–flight and flight–workstation restrictions, the proposed model captures the complex interplay between regulatory requirements, dispatcher fatigue considerations, and operational efficiencies. The inclusion of a flight-path similarity objective recognizes that dispatchers benefit from monitoring geographically similar flights, thereby reducing redundant cognitive tasks and improving overall workload distribution. Three solution methods were investigated: a weighted sum approach, an ϵ -constraint approach, and NSGA-II. To manage the larger and more realistic instances, we adapted a previously proposed genetic algorithm (FSI-GA) for the first two methods and compared it with an out-of-the-box NSGA-II implementation. Key conclusions from the computational experiments are as follows.

Exact methods (using a mixed integer programming solver) were effective for small instances but struggled with larger instances and often terminated with high optimality gaps. In contrast, NSGA-II consistently produced diverse sets of high-quality, non-dominated solutions for instances encompassing thousands of flights, demonstrating its scalability. NSGA-II outperformed scalarization-based methods in terms of both Pareto front size and coverage. Although the ϵ -constraint method can find the entire Pareto front through systematic exploration, in practice, it yielded fewer unique non-dominated solutions and incurred higher computational costs when tackling multiple objectives and large problem instances. Weighted-sum method generally explored a significantly restricted part of the solution space with a high computational cost.

Results highlight trade-offs between workload balance and flight-path similarity objectives. Assigning many flights to a single dispatcher can maximize similarity-based efficiency but sacrifices workload balance. Conversely, distributing flights inherently lowers the similarity score. Including flight–flight and flight–workstation restrictions adds further complexity among objectives while reflecting operational preferences. A detailed sensitivity analysis on a small instance revealed that adjusting constraints (e.g., minimum departure spacing, thresholds for long-haul flights, or workstation assignment rules) can substantially affect both solution quality and computational performance. Tighter restrictions in one dimension can either reduce or enhance efficiency in another if they guide the algorithm toward promising parts of the search space. This underscores the importance of carefully tuning operational policies when deploying multi-objective workload management models in practice. Airlines can tailor the models to emphasize preferred objectives such as prioritizing workload balance or concentrating on operational efficiency by grouping similar

flights. The multi-objective approach enables stakeholders to examine trade-offs before selecting an assignment policy that aligns with specific organizational priorities.

Overall, this work demonstrates that adopting a multi-objective perspective on flight-dispatcher assignment yields operationally relevant solutions that more accurately capture the realities of dispatcher workload. Incorporating flight-path similarity, mandatory restrictions, and multiple workload measures produces a broader spectrum of feasible and efficient assignments, offering decision-makers the flexibility to adapt as operational conditions evolve.

Future research could explore dynamic or real-time reassignment strategies under disruptions (e.g., weather, mechanical delays), refining flight path-similarity metrics to account for flight route structures, or embed stochastic elements to accommodate flight-uncertainty for workload. Additionally, integrating human-centric factors such as individual dispatcher expertise or fatigue models could further enhance the practical relevance of these assignment frameworks.

Chapter 4

A Machine Learning-based Column Generation Approach for Flight Dispatcher Workload Balancing

As shown in Chapter 2, the absolute workload deviation balancing problem achieves better balanced workloads compared to the peak minimization problem. It is, however, computationally challenging, the absolute deviation formulation is significantly more difficult to solve using traditional optimization methods, often failing to converge even for small-sized instances. In its Lagrangian relaxation, the primary computational bottleneck arises in the solution of the subproblems. To overcome this limitation, this chapter explores the integration of machine learning (ML) techniques with column generation. Specifically, we generate subproblem solutions using machine learning techniques. Let us recall the subproblem:

$$\min \sum_{t \in T} \left| \sum_{i \in I} h_{it} x_i - \eta_t \right| - \sum_{i \in I} \mu_i x_i \quad (4.1)$$

$$\text{s.t. } x_i \in \{0, 1\} \quad \forall i \in I. \quad (4.2)$$

Instead of solving it as an integer program, we train a neural network to predict solutions. The input data consists of Lagrangian multipliers μ_i , flight load data h_{it} , and average workload data η_t . A key focus is the training process, particularly the loss functions that effectively capture the subproblem's objective. These ML-driven predictions are then integrated into the column generation framework to accelerate convergence by providing effective column candidates.

The remainder of the chapter is structured as follows. Section 4.1 provides a literature review covering the intersection of machine learning and combinatorial optimization that is on applications in column generation and on solving unconstrained binary optimization problems. Section 4.2 details the proposed methodology, including the input data preprocessing, the solution prediction approach using neural networks, alternatives for modeling the objective within the loss function for training, and the neural network architecture. Section 4.3 analyzes the performance of the ML-assisted column generation approach against traditional methods. Section 4.4 concludes and suggests future research directions in integrating machine learning within optimization frameworks.

4.1 Literature Review

Balancing flight dispatcher workload using the absolute deviation measure poses substantial computational challenges, as even relatively small instances can be difficult to solve, and conventional optimization algorithms often fail to produce non-trivial solutions or near-optimal results. While decomposition techniques such as Lagrangian relaxation coupled with column generation can help mitigate complexity by iteratively building solutions (Morabit, Desaulniers, and Lodi, 2021; Bengio, Lodi, and Prouvost, 2018), the repeated solution of the Lagrangian subproblem often becomes the principal computational bottleneck (Shen et al., 2022). For the average deviation workload balancing problem, relaxing assignment constraints yields an unconstrained binary optimization subproblem with a nonlinear absolute value objective (4.1), which can account for a large fraction of the overall runtime.

In view of these challenges, a growing body of research explores the integration of machine learning approaches within optimization frameworks (Bengio, Lodi, and Prouvost, 2018; Shen et al., 2023). By approximating solutions or guiding the optimization process more effectively, machine learning techniques have demonstrated the potential for notable reductions in total solution time, with some reports indicating significant performance improvements (Abolghasemi et al., 2021). The following sections review foundational machine learning applications in combinatorial optimization, examine the use of machine learning in column generation, discuss its relevance to unconstrained binary optimization problems, and then present recent work on objective-informed loss function design.

4.1.1 Machine Learning in Combinatorial Optimization

Machine learning, spanning both supervised, unsupervised, and reinforcement learning paradigms, has shown promise for a wide range of combinatorial optimization problems,

often by overcoming scalability limitations or obviating the need for specialized problem-dependent heuristics (Bengio, Lodi, and Prouvost, 2018; Khalil et al., 2017; H. P. Wang et al., 2022).

Graph Neural Networks are prominent in problems that naturally map to graph structures, such as the Traveling Salesman Problem or Vehicle Routing Problem, where they can efficiently leverage structural information (Cappart et al., 2023; Gao et al., 2024). In contrast, the ADP Lagrangian subproblem involves time-varying loads (h_{it}), average loads (η_t), and Lagrangian multipliers (μ_i) rather than an explicit graph representation. Hence, neural architectures suited to high-dimensional inputs without a predefined graph structure may be more appropriate. As in many ML-based combinatorial optimization approaches, generalizing to problem instances that differ substantially from the training distribution remains a serious concern (Luo et al., 2023), particularly in dynamic operational environments such as flight dispatch.

4.1.2 Machine Learning for Column Generation

Research integrating machine learning techniques into column generation frameworks focuses mainly on selecting promising columns, predicting subproblem solutions, and stabilizing dual variables (Morabit, Desaulniers, and Lodi, 2021; Shen et al., 2023). Table 4.1 summarizes selected recent contributions, illustrating the variety of machine learning approaches (for instance, reinforcement learning, support vector machines, and neural networks) and target problems (such as Vehicle Routing, Cutting Stock, or Scheduling). Many of these studies report total runtime or iteration count reductions of 50% or more (Kraul, Seizinger, and Brunner, 2023; Y. Huang et al., 2023; W. Zhang, M. Li, and J. Wang, 2023).

Although these works frequently improve column-generation iterations or dual stabilization, they do not always reduce the complexity of the pricing (subproblem) computation itself. When, as in the ADP formulation, solving the subproblem is particularly difficult, the overall performance impact of these strategies may be limited. For instance, Morabit, Desaulniers, and Lodi (2021) used reinforcement learning for column selection in scheduling and routing, attaining up to 30% runtime reduction but assuming relatively tractable subproblems.

In contrast, several recent efforts aim directly at easing subproblem difficulty, bringing them closer to the objectives of this thesis. Specifically, machine learning models, trained offline or incrementally refined, have been employed to predict near-optimal solutions for pricing subproblems (Shen et al., 2023). Neural Column Generation approaches likewise use neural networks to estimate high-quality feasible solutions (Y. Huang et al., 2023; W.

Zhang, M. Li, and J. Wang, 2024), resulting in up to 50% runtime reductions. However, these techniques often assume a smoother or simpler objective than the absolute value function in the subproblem (4.1), prompting the need for careful design of the training loss.

Table 4.1: Summary of key machine learning applications in column generation

Paper	Problem	ML Method	Focus	Outcomes
Shen et al., 2022	Graph Coloring	SVM - Hybrid	CP	Improved convergence speed
Kraul, Seizinger, and Brunner (2023)	Cutting Stock Problem	Feedforward Neural Networks	DVP	Up to 50% reduction in runtime, mixed results in convergence
Y. Huang et al., 2023	Microgrid Management	Deep Neural Networks	DVP	Up to 50% reduction in runtime
Shen et al., 2023	Graph Coloring	Adaptive Logistic Regression	CP	Improved heuristic accuracy and speed
W. Zhang, M. Li, and J. Wang, 2023	Networked Microgrids	Graph Neural Networks	CS	Up to 60% reduction in runtime
Chen et al., 2024	Rider Planning and Scheduling	Attention-Gated GNN	RPP	Significant reduction in runtime and better convergence
J. Lu et al., 2024	Joint Rider Trip Planning and Crew Shift Scheduling	Attention-Gated GNN	RPP	Reduction in runtime with mixed results in convergence

CS: Column Selection

CP: Column Prediction

DVP: Dual Variable Prediction

RPP: Restricting Pricing Problem

4.1.3 Machine Learning for Unconstrained Nonlinear Binary Optimization

The Lagrangian subproblem of the absolute deviation problem belongs to the class of unconstrained binary optimization problems. Neural networks are widely studied for unconstrained binary optimization (Bengio, Lodi, and Prouvost, 2021), but the inclusion of non-differentiable components like absolute values can complicate gradient-based training (Paasivirta et al., 2024). A common strategy is to smooth or approximate the absolute value function, which allows more direct application of standard backpropagation methods, although specialized network architectures and non-smooth optimization techniques have

also been proposed (Karmita et al., 2023; Imaizumi and Fukumizu, 2019). Choice of loss function is central to such approaches, given its role in translating the problem’s objective structure to the training process, as discussed in the next subsection.

Efforts in related research areas such as difference-of-convex optimization and proximal methods also indicate that piecewise linear or non-smooth objectives, including absolute values, can be handled with suitable approximations and training strategies (Iutzeler and Malick, 2020; Louizos, Welling, and Kingma, 2017; Imaizumi and Fukumizu, 2019). These insights inform the development of specialized procedures for nonlinear unconstrained binary optimization problems, including the ADP subproblem.

4.1.4 Objective-Informed Loss Function Design

A critical aspect of training neural networks for combinatorial optimization problems is ensuring that the loss function aligns closely with the underlying optimization objective (Wilder, Dilkina, and Tambe, 2019). Generic regression or classification losses may fail to guide the network effectively when the goal is to minimize a complex objective, as in the absolute deviation problem.

One established approach for alignment involves embedding the problem’s objective directly into the NN loss function (Garmendia et al., 2024; Wilder, Dilkina, and Tambe, 2019). Other methods penalize deviation from known optimal or target values, or incorporate constraint-based terms within the loss itself (Keegan and Abolghasemi, 2023). In energy-based or quadratic unconstrained binary optimization formulations, Hamiltonian-inspired losses have been introduced to mirror the structure of the problem’s objective (Jiang et al., 2024; Bello et al., 2016).

Decision-focused learning, an end-to-end methodology, ties the loss function directly to the quality of the decisions produced by the network. The network’s outputs are evaluated against the true combinatorial objective, rather than a proxy, so that each update step in training aims to improve ultimate decision quality (Wilder, Dilkina, and Tambe, 2019).

In summary, machine learning has demonstrated its ability to enhance large-scale optimization methods, including column generation, by offering acceleration through useful subproblem predictions and other adaptive strategies (Shen et al., 2022; Yuan, Fang, and Song, 2024; Chi et al., 2022; H. Xu et al., 2024). Yet, many existing approaches are geared toward structurally simpler or graph-based combinatorial optimization problems and often assume smooth objectives. The Lagrangian subproblem of absolute deviation problem, for instance, presents distinct challenges arising from its non-smooth absolute deviation objective and its dependence on Lagrangian multipliers, both of which complicate standard

solution approaches. Recent advances in nonlinear unconstrained binary optimization, decision-focused learning, and tailored loss functions, however, provide promising avenues for addressing these issues.

Applying such machine learning techniques and leveraging these recent advances for the flight dispatcher workload balancing problem nevertheless remains challenging. This is due to the problem’s inherent computational complexity and the particularly non-smooth nature of the absolute deviation objective in its Lagrangian subproblem, which does not align well with standard graph-based neural architectures. Furthermore, while some ML-column generation integrations exist, few directly address the specific complexities of such non-linear, non-smooth subproblems. Building on insights from objective-informed loss design and non-smooth optimization strategies, the methodology detailed in the following sections combines machine learning with column generation. It outlines approaches that include using neural networks for direct subproblem solution prediction and implementing an alternative model that allows these methods to restrict the subproblem instead of predicting solutions, thereby handling the subproblem’s specific complexities and effectively incorporating its objective into the training process.

4.2 Methodology

In column generation, subproblems are repeatedly solved for different Lagrangian multipliers. A diagram illustrating the overall procedure and the interaction between the master problem, subproblem, and machine learning modules is provided in Figure 4.1. Each iteration starts by solving the master problem, dual solution of which yields a new set of Lagrangian multipliers that drive the subproblem. The loop continues until no column with negative reduced cost is found. The figure highlights four ML modules that can accelerate this process: Lagrangian multiplier prediction (blue), predicting multipliers to be used in the master problem; Column prediction (green), given the current multipliers, generating candidate columns expected to have negative reduced cost; Restricting subproblem (yellow), model predicts a subset of variables that are likely to take non-zero values in an entering column, so the exact subproblem is solved on a reduced search space, e.g., for fewer variables; and Column selection (red), when there are several eligible columns, a model chooses a subset to be added to the master to keep its size manageable. In this chapter we propose to use a neural network to find negative reduced cost-solutions for the subproblem. In addition, a sign-relaxation formulation is presented that transforms the absolute deviations into linear constraints, that can be partially guided by a learned or heuristic-based assignment of sign variables.

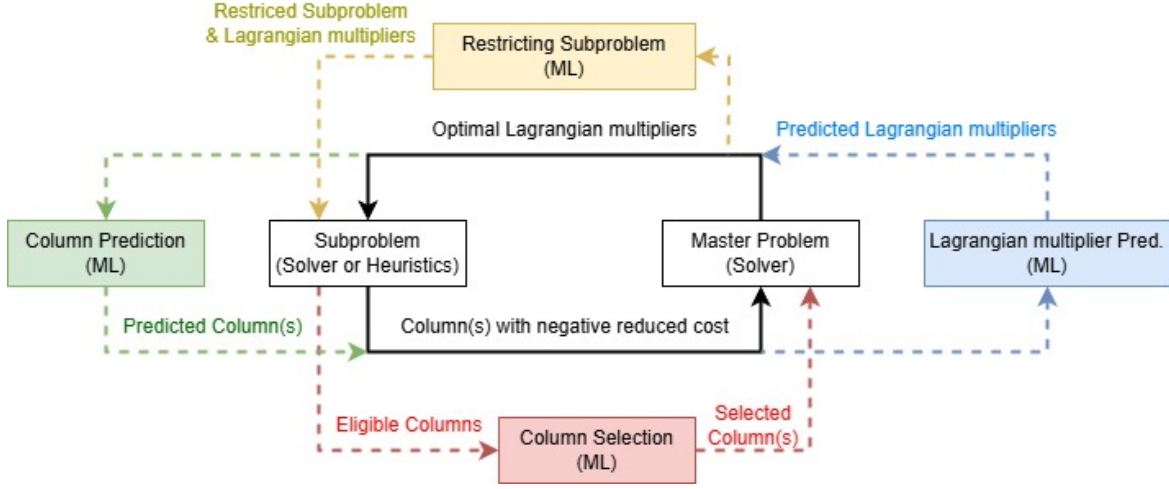


Figure 4.1: Overview of machine learning applications in column generation.

Let us start by recalling the master and subproblems corresponding to the Lagrangian relaxation:

$$\max \quad |K| \theta + \sum_{i \in I} \mu_i \quad (4.3)$$

$$\text{s.t.} \quad \theta + \sum_{i \in I} x_i^h \mu_i \leq \sum_{t \in T} \left| \sum_{i \in I} h_{it} x_i^h - \eta_t \right| \quad \forall h \in H, \quad (4.4)$$

$$\theta, \mu_i \in \mathbb{R} \quad \forall i \in I, \quad (4.5)$$

where x^h is a binary vector, $|K|$ is the set of dispatchers or workstations, θ is an auxiliary variable, and μ_i are the dual variables. The subproblem then seeks a column, corresponding to solution x^h , with negative reduced cost by solving:

$$\min_{x_i \in \{0,1\}} \quad \sum_{t \in T} \left| \sum_{i \in I} h_{it} x_i - \eta_t \right| - \sum_{i \in I} \mu_i x_i.$$

If the objective of the subproblem is strictly less than the current θ , the resulting column has negative reduced cost.

4.2.1 Column Prediction with Neural Networks

To solve the Lagrangian subproblem, a feedforward neural network is employed to generate a binary vector x_i from the inputs h_{it} , μ_i , and η_t . The network architecture is modular and can be optimized using systematic hyperparameter search algorithms, such as Hyperband (L. Li et al., 2018) or Bayesian optimization (Snoek, Larochelle, and Adams, 2012). Each input is processed by an embedding subnetwork composed of a dense layer whose output dimension is selected from a linear search space. The input h_{it} is processed through two independent embedding branches. One applies a pooling operation over the index set I , and the other over the time index set T . These operations yield distinct feature representations.

The resulting embeddings, together with the processed μ_i and η_t inputs, are fused in a subsequent processing layer. The combined representation is passed through a dense layer with a sigmoid activation function to produce the candidate solution vector x_i . In addition, a reordering branch is incorporated using a Sinkhorn network (Mena et al., 2018). This branch uses two dense layers to generate auxiliary vectors whose outer product forms logits that are normalized by a Sinkhorn operator. The number of normalization iterations and the temperature parameter are selected via hyperparameter tuning.

Training the network is conducted using a composite loss function, which may include both a solution-based component and an objective-based component. The following loss functions are considered. Where $\hat{\mathbf{x}}$ denote the predicted solution vector, $\mathbf{x}^*$ the optimal solution, and $z(\mathbf{x})$ the subproblem objective value evaluated on $\mathbf{x}$:

$$(\text{BC}) : \sum_{i \in I} x_i^* \log(\hat{x}_i) + (1 - x_i^*) \log(1 - \hat{x}_i) \quad (4.6)$$

$$(\text{MSE SP}) : (z(\hat{\mathbf{x}}) - z(\mathbf{x}^*))^2 \quad (4.7)$$

$$(\text{BPEN}) : \sum_{i \in I} \hat{x}_i(1 - \hat{x}_i) \quad (4.8)$$

$$(\text{OPT}) : z(\hat{\mathbf{x}}) \quad (4.9)$$

These loss functions, which can be used individually or in a composite form, are designed to allow prioritizing different goals in the training. The Binary Cross-Entropy (BC) loss (4.6) was chosen as a standard approach to effectively train the network to replicate the component-wise binary structure of known high-quality or optimal solutions ($\mathbf{x}^*$). To more directly align network training with the goal of finding solutions that perform well in terms of the subproblem’s objective function, the Mean Squared Error of Subproblem Objective (MSE SP) loss (4.7) was considered, which penalizes deviations in the objective

value of the predicted solution relative to the optimal. As an unsupervised alternative, the Direct Subproblem Objective (OPT) loss (4.9) was included for its potential to guide the network to minimize the subproblem’s cost function directly, possibly discovering solutions while directly addressing the objective. Finally, the Binarization Penalty (BPEN) (4.8) was selected as a practical auxiliary term to encourage the network’s continuous activations towards discrete 0 or 1 values.

The subproblem objective contains absolute value terms, which are non-differentiable and thus not directly suitable for gradient-based optimization. To address this, smooth approximations are employed in the MSE SP and OPT loss formulations during training. The absolute deviation term $|a_t|$ is replaced by one of the following functions: the Huber loss (Huber, 1992), the log-cosh loss (R. A. Saleh and A. Saleh, 2022), or a simple quadratic loss. These functions are illustrated in Figure 4.2. They provide smooth surrogates that preserve the semantics of deviation while enabling differentiable optimization. The final evaluation of model performance continues to use the original, non-smoothed subproblem objective.

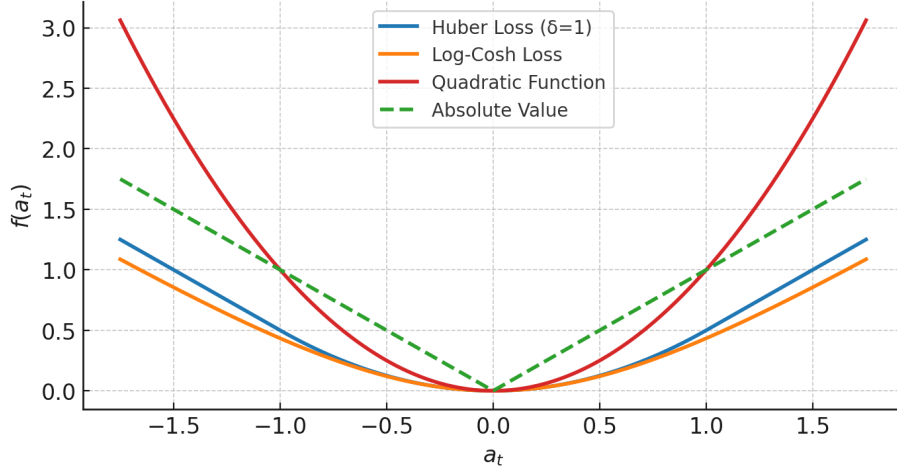


Figure 4.2: Visualization of the alternative smooth approximations and the absolute value function.

The modular architecture and fine-tuning framework enable systematic exploration of design choices for both the network and the loss function, with the aim of identifying models that can reliably generate near-optimal solutions to the subproblem.

4.2.2 An Alternative Sign Model

The difficulty of the Lagrangian subproblem can be alleviated by forcing the signs of nonlinear terms in the objective, effectively restricting the solution space. This leads to an easier model due to the restricted solution space, which is formulated by introducing an indicator variable

$$s_t = \begin{cases} 1 & \text{if } \sum_{i \in I} h_{it} - \eta_t > 0 \\ -1 & \text{otherwise} \end{cases}$$

with this, the subproblem becomes

$$\min \sum_{t \in T} s_t \left(\sum_{i \in I} h_{it} x_i - \eta_t \right) - \sum_{i \in I} \mu_i x_i \quad (4.10)$$

$$\text{s.t. } s_t = \text{sign} \left(\sum_{i \in I} h_{it} x_i - \eta_t \right) \quad \forall t \in T \quad (4.11)$$

$$x_i \in \{0, 1\} \quad \forall i \in I. \quad (4.12)$$

If optimal s_t^* is known, the sign constraint can be expanded, which leads to the following binary linear problem:

$$\min \sum_{t \in T} s_t^* \left(\sum_{i \in I} h_{it} x_i - \eta_t \right) - \sum_{i \in I} \mu_i x_i \quad (4.13)$$

$$\text{s.t. } \sum_{i \in I} h_{it} x_i - \eta_t \leq 0 \quad \forall t \in T : s_t^* = -1 \quad (4.14)$$

$$\sum_{i \in I} h_{it} x_i - \eta_t \geq 0 \quad \forall t \in T : s_t^* = 1 \quad (4.15)$$

$$x_i \in \{0, 1\} \quad \forall i \in I. \quad (4.16)$$

The objective (4.13) is further simplified to:

$$\sum_{i \in I} \left(\sum_{t \in T} s_t^* h_{it} - \mu_i \right) x_i. \quad (4.17)$$

If partial or uncertain information on s_t^* is available, a hybrid model may fix signs of those with known s_t^* to $\{-1, 1\}$ and let the solver handle others that remain unknown. This can reduce search complexity when the fixed signs are predicted accurately. In practice, such predictions can be generated by a learned classifier. The benefit of partially fixing the signs is a smaller feasible region for the solver to explore. However, inaccuracies in sign

predictions can degrade solution quality, highlighting a trade-off between computational savings and prediction risk.

4.3 Numerical Results

The majority of the computational tests were performed on servers of the Digital Research Alliance of Canada (Digital Research Alliance of Canada, 2025b). Unless otherwise noted, hyperparameter tuning algorithms were run on systems equipped with an NVidia A100SXM4 40GB GPU, 64 GB of RAM, and 16 cores of an AMD EPYC 7413 CPU (Zen 3). Inference and numerical tests were carried out on an NVidia RTX 4090 24GB GPU, 96 GB of RAM, and an Intel i9-11900k CPU. Column generation algorithms for the small problems were executed on Narval server nodes with 64 GB of RAM and an AMD EPYC 7532 (Zen 2) 2.4 GHz CPU. The algorithms for the full problems were deployed on two servers, Niagara and Cedar, which collectively offered more than six configurations, each containing at least 48 GB of RAM and CPU speeds ranging from 2.1 GHz to 2.4 GHz, further details are provided by Digital Research Alliance of Canada (2025a). The total computation time for the small instances was approximately 9.5 core-years, and for the full instances approximately 30 core-years. One core-year corresponds to using a single CPU core for one year. All methods were implemented in Python 3.11.5, using TensorFlow 2.17.0, KerasTuner 1.3.5, CUDA 11.3, and CUDNN 8.9.4. The exact solution of the mathematical models was provided by Gurobi 12.0.0 through its Python interface.

4.3.1 Results of the Column Generation

The baseline column generation algorithm, which employed an exact subproblem solver, was applied to both small (subset) and full benchmark instances in order to generate data and establish reference performance. On the small instances, the method generally reached optimality within one day, except for one problem instance. The full instances, however, posed much greater computational challenges and frequently failed to converge, even after a full week of processing on high-performance computing systems.

A prominent observation across all instances was the difficulty of the Lagrangian subproblem. Even when the overall algorithm made progress, the exact subproblem solver often spent substantial time reducing the optimality gap, especially when improving the solution from a 1% gap to a 0.1% gap. To address this during data generation, the subproblem solver was configured to check, once per minute, whether the incumbent solution provided a negative reduced cost. This strategy allowed the algorithm to advance

more quickly on relatively easier instances but did not succeed in producing any negative reduced-cost columns during the first iteration for any instances derived from United Air Lines, SkyWest Airlines, American Airlines, or Delta Air Lines, thus underscoring the subproblem’s difficulty for certain datasets.

These experiments obtained approximately two million column generation iterations. However, the dataset was inherently biased toward instances that were computationally less demanding, since those instances produced more iterations (data points) in the same time frame. To mitigate this bias and emphasize the more challenging instances, a sample-weighting scheme was adopted in training the machine learning models. Each data sample (iteration) was assigned a weight proportional to the computational effort, measured in GHz-core-seconds (for instance, an iteration requiring five seconds on a 2.4 GHz CPU with 16 cores corresponds to $5 \times 2.4 \times 16 = 192$ GHz-core-seconds). Samples were then ranked by this difficulty measure and assigned discrete weights from 0.1 to 1.0 in increments of 0.1, ensuring that more difficult iterations received higher priority in the learning process.

Examination of the solution times for the subproblem and master problem throughout the column generation algorithm revealed a characteristic dynamic. Figure 4.3 illustrates typical results for the Alaska Airlines Feb 11th R100 instance. At the outset, the SP dominates the runtime. As columns are generated, the SP becomes more tractable for a span of iterations, while the MP expands in size and complexity, thus increasing its solution time. After several thousand iterations, however, SP difficulty often resumes its growth and in many cases exceeds its initial levels late in the run. This behavior underlines the persistent challenge that the subproblem imposes on the entire optimization process.

4.3.2 Results of the Solution Prediction Approach

An initial set of tests was performed using the neural network-based solution prediction method described in Section 4.2. Figure 4.4 displays results on a small instance with 50 flights from Alaska Airlines (Jan 16, 2019) and two workstations. The neural network in these tests was trained with the Mean Squared Error of the subproblem objective values (MSE SP) as the loss function. The plots depict the progression of the upper bound (MP objective) over iterations and over wall-clock time. The neural network predicts a value between 0 and 1 for each x_i , these values are first rounded. If the corresponding solution does not have a negative reduced cost, values are used as individual Bernoulli probabilities, from which 200 samples are collected. If no solution among them has a negative reduced cost, the method reverts to the solver for that iteration.

In this example, the first fallback to the solver occurred at iteration 293, where the upper

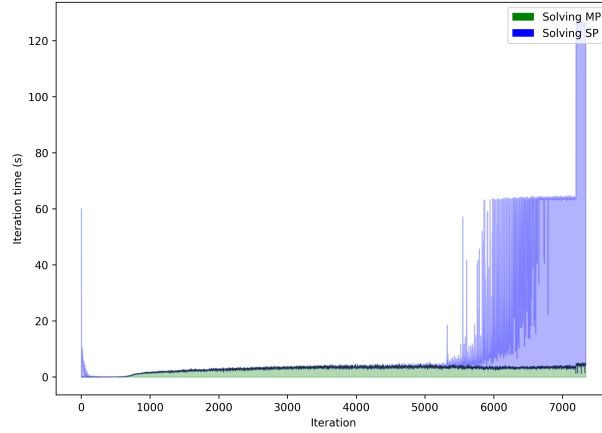


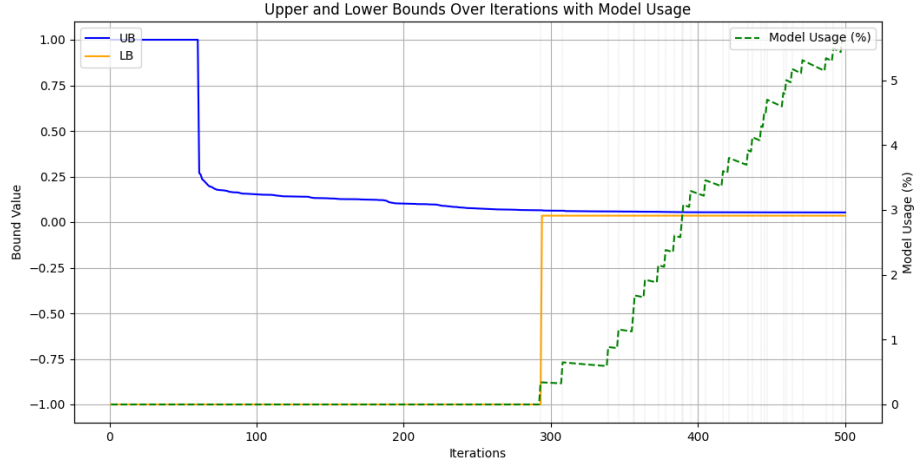
Figure 4.3: Solution time distribution for the column generation algorithm on the Alaska Airlines Feb 11th R100 instance, showing Subproblem (SP) and Master Problem (MP) solution times per iteration.

bound was 0.065. By iteration 400, the upper bound had decreased to 0.053. Between iterations 293 and 400, a total of 24 fallback events were recorded, corresponding to a rate of approximately 22.5%. Across the full run, the neural network generated 1,099 columns, while the exact solver contributed 24 columns during fallback iterations.

These results represent an early-stage evaluation of the proposed methodology. They are limited to a single network architecture trained with the (MSE SP) loss on a small-scale instance. Several components described in Section 4.2 remain to be implemented or systematically assessed. These include:

- Testing additional loss functions, such as binary cross-entropy (BC), direct objective-based loss (OPT), and their respective variants with binary penalties.
- Evaluating the impact of smooth approximations of the absolute value function, including Huber, log-cosh, and quadratic loss formulations, as described in Section 4.2 and visualized in Figure 4.2.
- Performing comprehensive hyperparameter tuning for the neural network architecture.
- Scaling the solution prediction approach to large benchmark instances, where exact column generation is computationally most expensive.

(a) Convergence over Iterations



(b) Convergence over Time

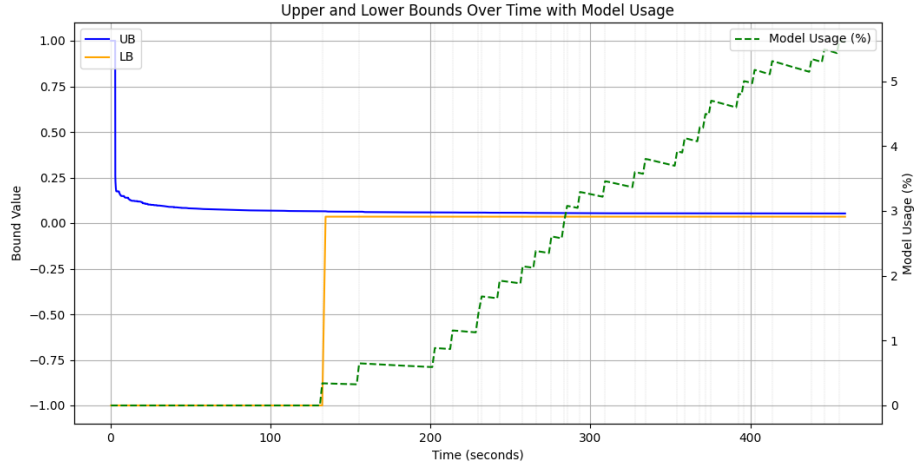


Figure 4.4: Column generation using the neural network predictor (trained with MSE SP loss) on a 50-flight Alaska Airlines 2019-1-16 instance with two workstations: (a) Upper bound per iteration, (b) Upper bound over time. Fallbacks to the exact solver occur when no negative reduced-cost column is found in 200 attempts.

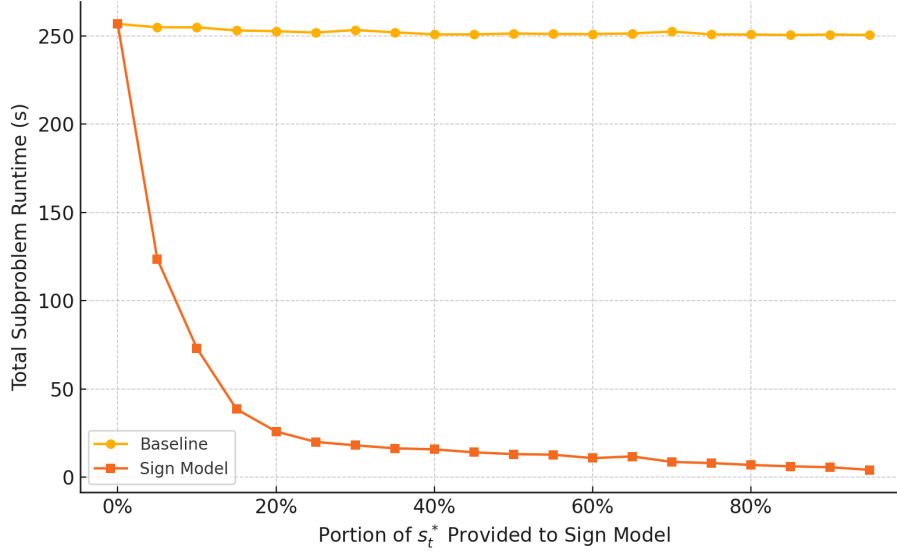


Figure 4.5: Subproblem runtime as a function of the portion of s_t^* signs provided to the sign model.

- Conducting a broad comparison between machine-learning-assisted and exact-solver-based column generation methods.

These extensions are critical not only for establishing the general effectiveness, limitations, and potential advantages of machine learning in this specific application but also for mitigating the risk of generalization bias that could arise from the current early-stage and limited-scope evaluation.

4.3.3 Preliminary Results of the Sign Model

An exploratory study was conducted to examine the potential runtime savings of the sign model introduced in Section 4.2. To conduct this study, a small problem instance proposed by Rayner (2021) was used. At each iteration, the subproblem was first solved to optimality to obtain an optimal solution $\mathbf{x}^*$, from which the optimal signs $s_t^* = \text{sign}(\sum_{i \in I} h_{it}x_i - \eta_t)$ were derived. The sign model was then solved on a modified version of the subproblem, where a fixed fraction of s_t^* values were provided to the model, while the remaining signs were left to be enforced through the original constraints in the linear reformulation. The fraction of signs provided ranged from 0% to 95% in increments of 5%, with a new random selection of time indices $t \in T$ sampled at each iteration to avoid bias.

Table 4.2: Runtimes for baseline and sign model under varying portions of s_t^* provided.

Portion s_t^* known (%)	Baseline Model Runtime (s)	Sign Model Runtime (s)
0	256.755	256.755
5	254.876	123.674
10	254.831	73.162
15	253.080	38.645
20	252.646	25.994
25	251.864	20.093
30	253.311	18.149
35	251.990	16.455
40	250.834	15.856
45	250.844	14.183
50	251.341	13.160
55	251.095	12.856
60	251.036	10.902
65	251.361	11.859
70	252.459	8.743
75	250.833	8.101
80	250.767	7.022
85	250.507	6.244
90	250.726	5.752
95	250.532	4.248

In Figure 4.5 and Table 4.2, baseline runtimes correspond to the original subproblem with no sign information, while the sign model runtimes are the time it took to solve the partially fixed formulation. Results demonstrate substantial reductions in runtime as the portion of known signs increases. Notably, providing just 10% of the s_t^* values—computed from the optimal solution of the full model—halved the runtime. At 20% known signs, the runtime dropped by nearly 90%, and further reductions continued as more signs were fixed. These findings indicate that even partial knowledge of the sign structure can yield significant computational gains.

In practical settings, however, such sign information would likely come from predictions made by a learned model, such as a classifier or reinforcement learning agent. This introduces a key trade-off where correct predictions speed up the subproblem while incorrect ones may degrade solution quality or lead to infeasibility. A complete assessment of the sign model must therefore evaluate this balance, quantifying not only runtime improvements but also the performance of solutions under imperfect sign information.

These initial results provide strong support for the value of incorporating sign structure into the solution process, motivating further study into prediction accuracy, error tolerance, and integration with learning-based approaches.

4.3.4 Next Steps on Neural Network Architecture and Fine-Tuning Parameters

The previous sections detailed initial findings from the column generation algorithm, an early-stage evaluation of a solution prediction neural network based on a single architecture, and preliminary results from a sign model. These explorations indicated the potential benefits of machine learning assistance while also highlighting the necessity for more systematic development and thorough evaluation of the neural network component itself.

Although an empirical validation of the fine-tuned neural network architecture extends beyond the primary experimental scope of this chapter, outlining this intended design and optimization process is relevant for several reasons. It provides a concrete roadmap for subsequent research phases. The proposed architectural considerations and the associated fine-tuning strategy represent a considered methodological approach. This plan illustrates how one might systematically address the task of designing a neural network tailored for predicting solutions to this Lagrangian subproblem. Presenting this plan also offers important context for the preliminary results shared earlier and it demonstrates the depth of consideration given to critical design choices. It should be noted that, in practice, the

final selection of a specific architecture and its parameters would depend on empirical performance on a validation set.

The proposed neural network architecture will be determined through a fine-tuning process implemented via KerasTuner (O’Malley et al., 2019). The design and hyperparameter search plan is described as follows. For a better readability of tuner parameters, let $a : b : c$ denote a parameter that is searched within the linear space defined from a to c (inclusive) in steps of b .

The network accepts the inputs h_{it} , μ_i , and η_t . Each input is processed by a dense embedding layer whose output dimension is selected from the linear space 32:32:512. For the input h_{it} , two independent embedding layers are employed. One embedding layer is followed by a maximum pooling operation over the index I and the other is followed by a maximum pooling operation over the time index T .

The outputs of the two h_{it} embedding layers, together with the processed μ_i and η_t inputs, are combined to form a set of candidate features. A subsequent processing layer integrates these features. The hyperparameter search considers 64 distinct network structures based on the possible combinations of feature integration; variations in layer sizes are explored independently.

The primary branch produces a dense layer with a dimension equal to the number of elements in I . This layer uses a sigmoid activation function to generate the candidate binary solution x_i . A secondary branch is implemented to reorder the candidate solution. In this branch, two dense layers of dimension equal to the number of elements in I generate auxiliary vectors v_1 and v_2 . Their outer product produces logits that are normalized by a Sinkhorn operator. The Sinkhorn operator is tuned with the number of iterations selected from the linear space 5:2:25 and the temperature parameter from the linear space 0.05:0.05:0.3.

Two types of loss functions are considered. The solution-based loss quantifies the discrepancy between the predicted x_i and a reference solution using metrics such as mean squared error or cross-entropy. The objective-based loss evaluates the subproblem cost on the predicted x_i or on its smooth approximation. A hybrid loss function that combines both types may also be considered. Final hyperparameter values and network topology will be selected based on performance on a validation set.

4.4 Conclusion

The chapter addressed the computational challenges associated with the Flight Dispatcher Absolute Workload Deviation Balancing Problem, focusing on the bottleneck that arises from solving the Lagrangian subproblem within a column generation framework. Recognizing the limitations of conventional optimization approaches on large-scale instances, the research explored the integration of machine learning techniques as a means of accelerating the solution process. Specifically, neural networks were employed to predict and generate high-quality candidate solutions for the computationally intensive subproblem, thereby reducing the reliance on exact solvers. A sign-relaxation reformulation was also introduced as a complementary strategy, potentially guided by ML-based sign predictions to simplify the subproblem structure.

Computational experiments confirmed that standard column generation struggled to solve the absolute deviation formulation efficiently, with large instances often failing to converge within practical time limits. The subproblem solver was responsible for a considerable fraction of the overall runtime, which reinforced the motivation to incorporate machine learning. The baseline runs served as valuable training data for the ML models and revealed the need to address data imbalances, for example through sample weighting. Preliminary tests involving a neural network trained with an objective-based loss function (MSE SP) on a small instance indicated the feasibility of the ML-assisted approach. Although the trained network successfully generated improving columns for the master problem, fallback calls to the exact solver remained necessary, indicating that machine learning predictions were not always sufficient. Conceptual analysis of the sign relaxation suggested potential speedups if sign information were accurately determined, warranting further study.

These results are preliminary, given that the exploration was limited to a single machine learning configuration (one neural network architecture and a single loss function) on a small-scale problem instance. A comprehensive evaluation necessitates testing a broader range of machine learning methods, encompassing additional loss functions, smooth objective approximations, and the sign relaxation approach potentially combined with an ML-based sign predictor. It is also essential to conduct extensive experimentation on large-scale benchmark instances, where existing methods fail and the comparative performance of various machine learning strategies can be thoroughly assessed. In addition, investigating the generalization capabilities of the trained models on unseen data and studying the impact of potential prediction errors, especially within the sign relaxation context, are crucial for an assessment.

In conclusion, this work represents an initial step toward an ML-enhanced optimization

framework for the ADP. By demonstrating the potential of machine learning to expedite the Lagrangian subproblem, it contributes to the growing body of research at the intersection of machine learning and combinatorial optimization. Although substantial research and development remain to be undertaken, the preliminary findings suggest that hybrid approaches, combining established decomposition methods with modern machine learning, have promise for addressing large-scale, complex operational challenges such as flight dispatcher workload balancing, ultimately aiming for more efficient and equitable resource allocation in the airline industry.

Chapter 5

Conclusions and Future Research Directions

This thesis addressed the complex problem of balancing workload among airline flight dispatchers, a role that is essential for operational safety and efficiency in commercial aviation. As highlighted in Chapter 1, traditional methods for managing dispatcher assignments often rely on heuristics or simplified models and do not fully capture the dynamic, multi-tasking nature of the job or the impact of workload on dispatcher performance. This research addressed these limitations by developing realistic optimization models and solution methodologies that support systematic and balanced workload distribution.

Chapter 2 established a formal definition of the flight dispatcher workload balancing problem. The work considered the simultaneous management of multiple flights with load requirements that change over time. Two measures of workload balance were investigated. One measure focused on reducing the highest workload to avoid extreme peaks. The other measure aimed at minimizing the total deviation from the average workload to achieve a consistent distribution of tasks. A Lagrangian relaxation method produced a closed form lower bound for the maximum workload measure while a three phase genetic algorithm obtained near optimal solutions for large scale instances. This chapter suggests that future studies explore different workload balance functions and simpler yet effective measures. An empirical study in a flight dispatch centre may provide insight into how flights influence dispatcher workload and reveal factors that differentiate dispatcher performance.

Chapter 3 extended the analysis by incorporating operational complexities into a multi objective framework. The research integrated the workload balance measure with flight path similarity and assignment restrictions. Flight path similarity encourages the group-

ing of flights in the same region so that redundant monitoring is reduced. Assignment restrictions address conflicts that arise from close departure times and other operational constraints. Three solution methods were compared. A weighted sum approach and an epsilon constraint approach were evaluated against a non dominated sorting genetic algorithm. The non dominated sorting genetic algorithm produced a wide range of trade off solutions that clearly show the balance between workload and flight path similarity. This chapter invites further investigation into the criteria used by real systems during assignment, the risks considered during the process and the priorities set by different airline carriers.

Chapter 4 focused on the computational challenges associated with the workload deviation measure. The repeated resolution of a large Lagrangian subproblem was identified as the main obstacle. A hybrid approach combined machine learning with column generation. Neural networks were trained to predict near optimal solutions using Lagrangian multipliers and flight data. The predictions were integrated into the column generation process to reduce reliance on exact solution methods and to accelerate convergence. This work points to the potential of testing more advanced neural network models such as recurrent or generative architectures and evaluating alternative loss functions that capture the objective function along with other important factors. It also calls for research on effective training strategies in an iterative framework where each step influences the next.

The chapters together provide a comprehensive study of flight dispatcher workload balancing. The work shows that focusing solely on reducing the highest workload does not yield a balanced outcome. Minimizing the total deviation from the average workload produces a more even distribution of tasks even though it demands more computational effort. The multi objective framework captures additional operational factors that affect assignments while the integration of machine learning into optimization offers a promising way in dealing with the computational challenges.

Future research should build on these contributions to further improve flight dispatcher workload management. In relation to the first study, researchers may explore different workload balance functions and simplified measures that retain balance quality while reducing computational effort. A study conducted in a flight dispatch centre may help to understand how flights affect workload and which factors distinguish dispatcher performance. The second study calls for further investigation into the practical criteria that guide assignment decisions in operational settings. Research should examine the risks that decision makers consider and the different ways that airline carriers set their priorities. The third study opens the possibility of testing advanced neural network architectures and alternative training methods. Evaluating various loss functions and adjusting training strategies in an iterative framework will be essential to handle the interdependent nature

of the column generation process. These future research directions aim to refine the models and methods for flight dispatcher workload balancing so as to enhance the efficiency, resilience and safety of airline operations.

Further research could also explore the application of reinforcement learning (RL) techniques to address the challenges in solving the Lagrangian subproblem or to guide the column generation process more adaptively. As discussed in the broader literature (e.g., Khalil et al. (2017), Morabit, Desaulniers, and Lodi (2021)), RL has shown potential in learning heuristics for combinatorial optimization problems. For the flight dispatcher workload balancing problem, an RL agent could potentially learn a policy to generate promising columns directly by treating variable assignments as sequential decisions within a Markov Decision Process (Bøhn, 2022). This might involve designing a reward function that aligns with the subproblem objective. Alternatively, RL could be used to dynamically adjust parameters within the column generation framework or to guide the selection of signs in the sign-model approach. Developing such an RL framework, including the agent design, state and action spaces, and reward mechanisms tailored to this specific problem, represents a significant direction for future investigation.

References

- ABBB (2020). *Global Fact Sheet*. Accessed: 2023-09-02. URL: <https://www.iata.org/en/iata-repository/pressroom/fact-sheets/fact-sheet-benefits-aviation-statistics/>.
- Abdelghany, Ahmed, Khaled Abdelghany, and Vitaly S Guzhva (2024). “Schedule-level optimization of flight block times for improved airline schedule planning: A data-driven approach”. In: *Journal of Air Transport Management* 115, p. 102535.
- Abolghasemi, Mahdi et al. (2021). “How to effectively use machine learning models to predict the solutions for optimization problems: lessons from loss function”. In: *arXiv preprint arXiv:2105.06618*.
- ACI (May 2024). *Top 10 busiest airports in the world shift with the rise of international air travel demand*. Accessed: 2024-05-17 ([Archived](#)). URL: <https://aci.aero/2024/04/14/top-10-busiest-airports-in-the-world-shift-with-the-rise-of-international-air-travel-demand/>.
- Agnetis, Alessandro and Stefano Smriglio (2000). “Optimal assignment of high multiplicity flight plans to dispatchers”. In: *Naval Research Logistics (NRL)* 47.5, pp. 359–376.
- Alharasees, Omar and Utku Kale (2024). “Aviation Operators’ Total Loads Analysis by Multi-Criteria Decision-Making”. In: *Journal of Air Transport Management* 118, p. 102596.
- Anh, Nguyen Duy (2022). “Workload Balancing Among Healthcare Workers Under Uncertain Service Time Using Distributionally Robust Optimization”. In: *Asia-Pacific Journal of Operational Research* 39.05, p. 2150045.
- ATAG (2020). *Facts & Figures*. Accessed: 2023-09-02. URL: <https://www.atag.org/facts-figures>.
- Barnhart, Cynthia and Amy Cohn (2004). “Airline schedule planning: Accomplishments and opportunities”. In: *Manufacturing & Service Operations Management* 6.1, pp. 3–22.
- Bartels, Matt (2019). “The Other Half of Your Flight - Flight dispatch and Operations Management”. In: *Presented at FlightSimExpo, Orlando, June 8, 2019*, Accessed: 2023-12-08. URL: <https://www.youtube.com/watch?v=dEjryXWgWc0>.

- Bello, Irwan et al. (2016). “Neural combinatorial optimization with reinforcement learning”. In: *arXiv preprint arXiv:1611.09940*.
- Bengio, Yoshua, Andrea Lodi, and Antoine Prouvost (2018). “Machine Learning for Combinatorial Optimization: a Methodological Tour d’Horizon”. In: *arXiv preprint arXiv:1811.06128*.
- (2021). “Machine learning for combinatorial optimization: a methodological tour d’horizon”. In: *European Journal of Operational Research* 290.2, pp. 405–421.
- Bøhn, Eivind (2022). “Reinforcement learning for optimization of nonlinear and predictive control”. PhD thesis. Norwegian University of Science and Technology.
- Bouwer, Jaap, Vik Krishnan, and Steve Saxon (2020). *Will airline hubs recover from covid-19?* Accessed: 2023-09-02. URL: <https://www.mckinsey.com/industries/travel-logistics-and-infrastructure/our-insights/will-airline-hubs-recover-from-covid-19>.
- Bouwer, Jaap, Vik Krishnan, Steve Saxon, and Caroline Tufft (2022). *Taking stock of the pandemic’s impact on Global Aviation*. Accessed: 2023-09-02. URL: <https://www.mckinsey.com/industries/travel-logistics-and-infrastructure/our-insights/taking-stock-of-the-pandemics-impact-on-global-aviation>.
- Bureau of Transportation Statistics (2023a). *General - Airline Domestic Market Share June 2022 - May 2023*. Accessed: 2023-09-02 ([Archived](#)). URL: <https://www.transtats.bts.gov/>.
- (2023b). *Reporting Carrier On-Time Performance (1987-present)*. Accessed: 2023-09-02 ([Archived](#)). URL: https://www.transtats.bts.gov/Fields.asp?gnoyr_VQ=FGJ.
- Burke, Edmund K, Patrick De Causmaecker, et al. (2004). “The state of the art of nurse rostering”. In: *Journal of Scheduling* 7.6, pp. 441–499.
- Burke, Edmund K, Jingpeng Li, and Rong Qu (2010). “A hybrid model of integer programming and variable neighbourhood search for highly-constrained nurse rostering problems”. In: *European Journal of Operational Research* 203.2, pp. 484–493.
- Buzdalova, Arina, Anna Matveeva, and Georgiy Korneev (2015). “Selection of auxiliary objectives with multi-objective reinforcement learning”. In: *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation*, pp. 1177–1180.
- Cai, Qing, Haojie Ang, and Sameer Alam (2022). “A multiobjective optimization approach for air traffic flow management for airspace safety enhancement”. In: *2022 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, pp. 1–9.
- Cappart, Quentin et al. (2023). “Combinatorial Optimization and Reasoning with Graph Neural Networks”. In: *Journal of Machine Learning Research* 24.130, pp. 1–61.
- Chankong, Vira and Yacov Y Haimes (2008). *Multiobjective decision making: theory and methodology*. Courier Dover Publications.

- Chen, Ming et al. (2024). “Learning to solve quadratic unconstrained binary optimization in a classification way”. In: *Advances in Neural Information Processing Systems* 37, pp. 114478–114509.
- Chi, Cheng et al. (2022). “A deep reinforcement learning framework for column generation”. In: *Advances in Neural Information Processing Systems* 35, pp. 9633–9644.
- Collier, Mike and Matt Bartels (2021). “Flight Planning & Dispatch Q&A”. In: *Presented at FlightSimExpo, San Diego, September 24, 2021*, Accessed: 2023-12-08. URL: <https://www.youtube.com/watch?v=zQ0s5pomn38>.
- (2023). “Flight Dispatch - The Other Half of Your Flight”. In: *Presented at FlightSimExpo, Houston, June 25, 2023*, Accessed: 2023-12-08. URL: <https://www.youtube.com/watch?v=5tWpcjdbJxI>.
- Davis, Martin (2024). *JTS Topology Suite*. Available at <https://github.com/locationtech/jts>.
- Deb, Kalyanmoy and Himanshu Jain (2013). “An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: solving problems with box constraints”. In: *IEEE transactions on evolutionary computation* 18.4, pp. 577–601.
- Deb, Kalyanmoy, Amrit Pratap, et al. (2002). “A fast and elitist multiobjective genetic algorithm: NSGA-II”. In: *IEEE transactions on evolutionary computation* 6.2, pp. 182–197.
- Digital Research Alliance of Canada (2025a). *Alliance Doc / Technical Documentation*. Accessed: 2025-02-19. URL: https://docs.alliancecan.ca/wiki/Technical_documentation.
- (2025b). *Home Page / Digital Research Alliance of Canada*. Accessed: 2025-02-19. URL: <https://alliancecan.ca/en>.
- Dove, N. (2024). *Air Transat flight dispatchers vote to strike. Will it affect flights?* <https://globalnews.ca/news/10459408/air-transat-flight-dispatchers-strike/>. Accessed: 2025-02-23.
- Ehrgott, Matthias (2005). *Multicriteria optimization*. Vol. 491. Springer Science & Business Media.
- Ester, Martin et al. (1996). “A density-based algorithm for discovering clusters in large spatial databases with noise”. In: *kdd*. Vol. 96. 34, pp. 226–231.
- FAA (2023). *Air Traffic By The Numbers*. https://www.faa.gov/air_traffic/by_the_numbers. Accessed: 2025-02-24.
- (2025). *Operating Requirements: Domestic, Flag, and Supplemental Operations, Dispatch release: Flag and domestic operations, 14 C.F.R. §121.599*. Accessed: 2025-03-02. URL: <https://www.ecfr.gov/current/title-14/section-121.599>.

- Gao, Chengrui et al. (2024). “Neural Solver Selection for Combinatorial Optimization”. In: *arXiv preprint arXiv:2410.09693*.
- Gariel, Maxime, Ashok N Srivastava, and Eric Feron (2011). “Trajectory clustering and an application to airspace monitoring”. In: *IEEE Transactions on Intelligent Transportation Systems* 12.4, pp. 1511–1524.
- Garmendia, Andoni I et al. (2024). “Marco: A memory-augmented reinforcement framework for combinatorial optimization”. In: *arXiv preprint arXiv:2408.02207*.
- Geoffrion, Arthur M (1974). “Lagrangian relaxation for integer programming”. In: *Approaches to integer programming*. Springer, pp. 82–114.
- Gov. of Canada (2025). *Flight Dispatcher Job Description*. <https://www.jobbank.gc.ca/marketreport/occupation/22706/ca>. Accessed: 2025-02-23.
- Gulbaz, Rohail et al. (2021). “Balancer genetic algorithm—A novel task scheduling optimization approach in cloud computing”. In: *Applied Sciences* 11.14, p. 6244.
- Harper, Lewis (2023). *Airline business covid-19 recovery tracker: August 2023 update*. Accessed: 2023-09-02. URL: <https://www.flightglobal.com/airlines/airline-business-covid-19-recovery-tracker-august-2023-update/154530.article>.
- Hart, Sandra G and Lowell E Staveland (1988). “Development of NASA-TLX (Task Load Index): Results of empirical and theoretical research”. In: *Advances in psychology*. Vol. 52. Elsevier, pp. 139–183.
- Hope, Allison (2017). “Your pilot isn’t actually in charge of your flight”. In: *Condé Nast Traveler*. Accessed: 2023-09-02 (*Archived*). URL: <https://www.cntraveler.com/story/this-person-is-more-in-control-of-your-flight-than-the-pilot>.
- Hu, Danlei et al. (2023). “Spatio-temporal trajectory similarity measures: A comprehensive survey and quantitative study”. In: *IEEE Transactions on Knowledge and Data Engineering* 36.5, pp. 2191–2212.
- Huang, Lixiao, Victoria Chibuogu Nneji, and Mary Cummings (2019). “How airline dispatchers manage flights: A task analysis in distributed and heterogeneous network operations”. In: *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*. Vol. 63, pp. 1389–1393.
- Huang, Yifan et al. (2023). “Learning Assisted Column Generation for Model Predictive Control”. In: *Energy Reports* 9, pp. 88–102.
- Huber, Peter J (1992). “Robust estimation of a location parameter”. In: *Breakthroughs in statistics: Methodology and distribution*. Springer, pp. 492–518.
- IATA (2021). *Latest travel restrictions and guidance*. Accessed: 2023-09-02. URL: <https://www.iata.org/en/publications/newsletters/iata-knowledge-hub/latest-travel-restrictions-and-guidance/>.
- (May 2024). *Passenger Demand Up 13.8% in March*. Accessed: 2024-05-17 (*Archived*). URL: <https://www.iata.org/en/pressroom/2024-releases/2024-05-01-01/>.

- IATA Economics (2022). *Air Passenger Market Analysis*. Accessed: 2023-09-02. URL: <https://www.iata.org/en/iata-repository/publications/economic-reports/air-passenger-monthly-analysis---april-2022/>.
- Ibáñez-Gijón, Jorge et al. (2023). “Experimental validation of COMETA model of mental workload in air traffic control”. In: *Journal of Air Transport Management* 108, p. 102378.
- ICAO (2020). *The World of Air Transport in 2019*. Accessed: 2023-09-02. URL: <https://www.icao.int/annual-report-2019/Pages/the-world-of-air-transport-in-2019.aspx>.
- IFOD (2025). *Flight Dispatcher Training / Aircraft Dispatch Certification*. Accessed: 2025-03-14. URL: <https://airlinedispatcher.com/programs/>.
- Ikli, Sana et al. (2021). “The aircraft runway scheduling problem: A survey”. In: *Computers & Operations Research* 132, p. 105336.
- Imaizumi, Masaaki and Kenji Fukumizu (2019). “Deep neural networks learn non-smooth functions effectively”. In: *The 22nd international conference on artificial intelligence and statistics*. PMLR, pp. 869–878.
- Iutzeler, Franck and Jérôme Malick (2020). “Nonsmoothness in machine learning: specific structure, proximal identification, and applications”. In: *Set-Valued and Variational Analysis* 28.4, pp. 661–678.
- Jensen, Mikkel T (2004). “Helper-objectives: Using multi-objective evolutionary algorithms for single-objective optimisation”. In: *Journal of Mathematical Modelling and Algorithms* 3, pp. 323–347.
- Jiang, Xia et al. (2024). “UNCO: Towards Unifying Neural Combinatorial Optimization through Large Language Model”. In: *arXiv preprint arXiv:2408.12214*.
- Karmitsa, Napsu et al. (2023). “Nonsmooth Optimization-Based Hyperparameter-Free Neural Networks for Large-Scale Regression”. In: *Algorithms* 16.9, p. 444.
- Keegan, Mitchell and Mahdi Abolghasemi (2023). “Approximating Solutions to the Knapsack Problem Using the Lagrangian Dual Framework”. In: *Australasian Joint Conference on Artificial Intelligence*. Springer, pp. 455–467.
- Khalil, Elias et al. (2017). “Learning Combinatorial Optimization Algorithms over Graphs”. In: *Advances in Neural Information Processing Systems*, pp. 6348–6358.
- Kraul, Sebastian, Markus Seizinger, and Jens O. Brunner (2023). “Machine Learning-Supported Prediction of Dual Variables for the Cutting Stock Problem with an Application in Stabilized Column Generation”. In: *INFORMS Journal on Computing* 35.3, pp. 1234–1248.
- Kumar, Neeraj and Kripa Shanker (2001). “Comparing the effectiveness of workload balancing objectives in FMS loading”. In: *International Journal of Production Research* 39.5, pp. 843–871.
- Legrain, Antoine, Hocine Bouarab, and Nadia Lahrichi (2015). “The nurse scheduling problem in real-life”. In: *Journal of Medical Systems* 39, pp. 1–11.

- Level-1 (2022). *United Airlines Operations Center Power Protection and AV infrastructure - level-1 global solutions, LLC*. Accessed: 2023-09-02. URL: <https://level-1.com/projects/united-airlines-operations-center-power-protection-and-av-infrastructure/>.
- Li, Lisha et al. (2018). “Hyperband: A novel bandit-based approach to hyperparameter optimization”. In: *Journal of Machine Learning Research* 18.185, pp. 1–52.
- Lina, Hui et al. (2024). “Research on Optimization of Flight Crew Scheduling Considering Pilot Fatigue”. In: *International Journal of Industrial Engineering Computations* 15, pp. 171–188. DOI: [10.5267/j.ijiec.2023.10.005](https://doi.org/10.5267/j.ijiec.2023.10.005). URL: https://www.growingscience.com/ijiec/Vol15/IJIEC_2023_48.pdf.
- Louizos, Christos, Max Welling, and Diederik P Kingma (2017). “Learning sparse neural networks through L_0 regularization”. In: *arXiv preprint arXiv:1712.01312*.
- Lu, Da and Fatma Gzara (2015). “The robust crew pairing problem: model and solution methodology”. In: *Journal of Global Optimization* 62.1, pp. 29–54.
- Lu, Jiawei et al. (2024). “Boosting Column Generation with Graph Neural Networks for Joint Rider Trip Planning and Crew Shift Scheduling”. In: *arXiv preprint arXiv:2401.03692*.
- Luo, Fu et al. (2023). “Neural Combinatorial Optimization with Heavy Decoder: Toward Large Scale Generalization”. In: *arXiv preprint arXiv:2305.12345*.
- Lynch, Kerry (2024). *NTSB Calls for Certified Dispatchers, Expanded Manifests in Part 135*. <https://www.ainonline.com/aviation-news/business-aviation/2024-08-13/ntsb-calls-certified-dispatchers-expanded-manifests-part>. Accessed: 2025-02-24.
- Maenhout, Broos and Mario Vanhoucke (2010). “Branching strategies in a branch-and-price approach for a multiple objective nurse scheduling problem”. In: *Journal of Scheduling* 13, pp. 77–93.
- Magdy, Nehal et al. (2015). “Review on trajectory similarity measures”. In: *2015 IEEE seventh international conference on Intelligent Computing and Information Systems (ICICIS)*. IEEE, pp. 613–619.
- Mannino, Carlo, Andreas Nakkerud, and Giorgio Sartor (2021). “Air traffic flow management with layered workload constraints”. In: *Computers & Operations Research* 127, p. 105159.
- Marler, R Timothy and Jasbir S Arora (2010). “The weighted sum method for multi-objective optimization: new insights”. In: *Structural and multidisciplinary optimization* 41, pp. 853–862.
- Mavrotas, George (2009). “Effective implementation of the ε -constraint method in multi-objective mathematical programming problems”. In: *Applied mathematics and computation* 213.2, pp. 455–465.

- Mena, Gonzalo et al. (2018). “Learning latent permutations with gumbel-sinkhorn networks”. In: *arXiv preprint arXiv:1802.08665*.
- Morabit, Mouad, Guy Desaulniers, and Andrea Lodi (2021). “Machine-Learning-Based Column Selection for Column Generation”. In: *arXiv preprint arXiv:2107.00146*.
- Munro, Pamela and Richard Mogford (2018). “Managing variability: a cognitive ethnography of the work of airline dispatchers”. In: *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*. Vol. 62. SAGE Publications Sage CA: Los Angeles, CA, pp. 182–186.
- NGA (2014). *Department of Defense World Geodetic System 1984, Its Definition and Relationships With Local Geodetic Systems*. Tech. rep. National Geospatial-Intelligence Agency. URL: <https://earth-info.nga.mil/php/download.php?file=coord-wgs84>.
- O’Malley, Tom et al. (2019). *KerasTuner*. <https://github.com/keras-team/keras-tuner>.
- OSGeo (2024). *GeoTools: The Open Source Java GIS Toolkit*. Available at <https://github.com/geotools/geotools>.
- Paasivirta, Pauliina et al. (2024). “Predicting pairwise interaction affinities with l0-penalized least squares—a nonsmooth bi-objective optimization based approach”. In: *Optimization Methods and Software*, pp. 1–28.
- Pang, Yutian et al. (2023). “Air Traffic Controller Workload Level Prediction using Conformalized Dynamical Graph Learning”. In: *arXiv preprint arXiv:2307.10559*. URL: <https://arxiv.org/abs/2307.10559>.
- Parasuraman, Raja and Matthew Rizzo (2006). *Neuroergonomics: The brain at work*. Oxford University Press.
- Pavlović, Goran, Radosav Jovanović, and Milan Stanojević (2023). “Rostering of air traffic controllers—current practices in Europe, key performance trade-offs and future opportunities”. In: *Journal of Air Transport Management* 113, p. 102472.
- Pei, Shan et al. (2021). “Decision support system for the irregular flight recovery problem”. In: *Research in Transportation Business & Management* 38, p. 100501.
- Rayner, Rebecca Sarah (2021). “Workload balancing for flight dispatcher scheduling”. Accessed: 2024-06-27. MA thesis. University of Waterloo. URL: <https://uwspace.uwaterloo.ca/handle/10012/17295>.
- Saleh, Resve A and AK Saleh (2022). “Statistical properties of the log-cosh loss function used in machine learning”. In: *arXiv preprint arXiv:2208.04564*.
- Santos, C.A. et al. (Jan. 2011). “Optimal dispatcher workload distribution”. In: *HP Laboratories Technical Report*, pp. 1–26.

- Santos, Cipriano A. et al. (2011). *Optimal Dispatcher Workload Distribution*. Tech. rep. HPL-2011-195. Available at <https://shiftleft.com/mirrors/www.hpl.hp.com/techreports/2011/HPL-2011-195.pdf>. HP Laboratories.
- Schoenfelder, Jan et al. (2020). “Nurse scheduling with quick-response methods: Improving hospital performance, nurse workload, and patient experience”. In: *European Journal of Operational Research* 283.1, pp. 390–403.
- Schwerdfeger, Stefan and Rico Walter (2018). “Improved algorithms to minimize workload balancing criteria on identical parallel machines”. In: *Computers & Operations Research* 93, pp. 123–134.
- Sheffield Sch. of Aeronautics (2016). *Aircraft Dispatchers Battle Severe Weather*. <https://www.sheffield.com/articles/flight-dispatchers-battle-severe-weather>. Accessed: 2025-02-23.
- Shen, Yunzhuang et al. (2022). “Enhancing Column Generation by a Machine-Learning-Based Pricing Heuristic for Graph Coloring”. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 36.9, pp. 9926–9934. DOI: [10.1609/aaai.v36i9.21230](https://doi.org/10.1609/aaai.v36i9.21230).
- (2023). “Adaptive Solution Prediction for Combinatorial Optimization”. In: *European Journal of Operational Research* 309.3, pp. 1392–1408. DOI: [10.1016/j.ejor.2023.01.054](https://doi.org/10.1016/j.ejor.2023.01.054).
- Sheth, Kapil S, Thomas J Amis, and Sebastian Gutierrez-Nolasco (2009). “Analysis of probabilistic weather forecasts for use in air traffic management”. In: *Aviation, Range and Aerospace Meteorology, Special Symposium on Weather Air Traffic Management Integration*.
- Sloan, Chris (2013). *Inside United Airlines Network Operations Control*. Accessed: 2023-09-02. URL: <https://airwaysmag.com/inside-united-airlines-network-operations-control/>.
- Snoek, Jasper, Hugo Larochelle, and Ryan P Adams (2012). “Practical bayesian optimization of machine learning algorithms”. In: *Advances in neural information processing systems* 25.
- Souza, L. (2024). *Behind The Scenes: We Explored Air Canada’s Operations Center*. <https://simpleflying.com/inside-air-canada-operations-center/>. Accessed: 2025-02-23.
- Starita, Stefano et al. (2020). “Air traffic control capacity planning under demand and capacity provision uncertainty”. In: *Transportation Science* 54.4, pp. 882–896.
- Stecke, Kathryn E (1983). “Formulation and solution of nonlinear integer production planning problems for flexible manufacturing systems”. In: *Management Science* 29.3, pp. 273–288.
- Steuer, Ralph E and Eng-Ung Choo (1983). “An interactive weighted Tchebycheff procedure for multiple objective programming”. In: *Mathematical programming* 26, pp. 326–344.

- T4 (2022). *U.S. airline market share*. Accessed: 2023-09-02 ([Archived](#)). URL: <https://www.t4.ai/industry/us-airline-market-share>.
- Tan, Jay Francis P et al. (2024a). “The Role of Flight Dispatchers: Analysis of Workload Balancing of Scheduled Domestic Flights”. In: *International Journal of Multidisciplinary Research and Growth Evaluation* 05.03, pp. 366–379. DOI: [10.54660/IJMRGE.2024.5.3.366-379](https://doi.org/10.54660/IJMRGE.2024.5.3.366-379). URL: <https://doi.org/10.54660/IJMRGE.2024.5.3.366-379>.
- (2024b). “The role of flight dispatchers: Analysis of workload balancing of scheduled domestic flights at the airport in Pasay city”. In: *International Journal of Multidisciplinary Research and Growth Evaluation*.
- Todoric, Marija and Toni Mastelic (2023). “Comparison of Similarity Measures for Trajectory Clustering-Aviation Use Case”. In: *Journal of Communications Software and Systems* 19.2, pp. 178–187.
- Trapote-Barreira, César, Andreas Deutschmann, and Francesc Robusté (2016). “Managing airlines: the cost of complexity”. In: *Transportation research procedia* 18, pp. 297–304.
- Turhan, Serkan, Samir Elhedhli, and Fatma Gzara (2025). “Workload Balancing for Flight Dispatchers”. In: *Computers & Operations Research*. Under review.
- U.S. Code of Federal Regulations Title 14 (2022a). *Certification: Airmen Other Than Flight Crewmembers: Aircraft Dispatcher Courses, 14 C.F.R. §Appendix A to Part 65*. Accessed: 2023-09-24. URL: <https://www.ecfr.gov/current/title-14/chapter-I/subchapter-D/part-65/appendix-Appendix%20A%20to%20Part%2065>.
- (2022b). *Operating Requirements: Domestic, Flag, and Supplemental Operations, Dispatch release: Flag and domestic operations, 14 C.F.R. §121.687*. Accessed: 2023-09-24. URL: <https://www.ecfr.gov/current/title-14/chapter-I/subchapter-G/part-121/subpart-V/section-121.687>.
- Vanheusden, Sarah et al. (2022). “Analysing the effectiveness of workload balancing measures in order picking operations”. In: *International Journal of Production Research* 60.7, pp. 2126–2150.
- Wang, Haoyu Peter et al. (2022). “Unsupervised learning for combinatorial optimization with principled objective relaxation”. In: *Advances in Neural Information Processing Systems* 35, pp. 31444–31458.
- Wickens, Christopher D et al. (2021). *Engineering psychology and human performance*. Routledge.
- Wilder, Bryan, Bistra Dilkina, and Milind Tambe (2019). “Melding the data-decisions pipeline: Decision-focused learning for combinatorial optimization”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 01, pp. 1658–1665.
- Xu, Hongjie et al. (2024). “Machine Learning-Enhanced Ant Colony Optimization for Column Generation”. In: *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 1073–1081.

- Xu, Yifan (2024). “Perspectives on Modelling Airline Integrated Scheduling Problem: a Review on State-of-the-Art Methodologies”. In: *Journal of the Air Transport Research Society*, p. 100023.
- Xu, Yifan, Sebastian Wandelt, and Xiaoqian Sun (2021). “Airline integrated robust scheduling with a variable neighborhood search based heuristic”. In: *Transportation Research Part B: Methodological* 149, pp. 181–203.
- Yildiz, Burak C, Fatma Gzara, and Samir Elhedhli (2017). “Airline crew pairing with fatigue: Modeling and analysis”. In: *Transportation Research Part C: Emerging Technologies* 74, pp. 99–112.
- Ying, Kuo-Ching, Pourya Pourhejazy, and Xin-Yi Huang (2024). “Revisiting the development trajectory of parallel machine scheduling”. In: *Computers & Operations Research*, p. 106709.
- Young, Mark S and Neville A Stanton (2002). “Attention and automation: New perspectives on mental underload and performance”. In: *Theoretical issues in ergonomics science* 3.2, pp. 178–194.
- Yuan, Haofeng, Lichang Fang, and Shiji Song (2024). “A reinforcement-learning-based multiple-column selection strategy for column generation”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 38. 8, pp. 8209–8216.
- Zeighami, Vahid and François Soumis (2019). “Combining Benders’ decomposition and column generation for integrated crew pairing and personalized crew assignment problems”. In: *Transportation Science* 53.5, pp. 1479–1499.
- Zhang, Qingfu and Hui Li (2007). “MOEA/D: A multiobjective evolutionary algorithm based on decomposition”. In: *IEEE Transactions on evolutionary computation* 11.6, pp. 712–731.
- Zhang, Wei, Ming Li, and Jianhui Wang (2023). “Graph Neural Network Based Column Generation for Energy Management in Networked Microgrid”. In: *Energy Reports* 9, pp. 88–102.
- (2024). “A Neural Column Generation Approach to the Vehicle Routing Problem with Two-Dimensional Loading and Last-In-First-Out Constraints”. In: *arXiv preprint arXiv:2406.12454*.

APPENDICES

Appendix A

Chapter 2 Appendix

A.1 Focus-Search-Improve Genetic Algorithm

Input: Instance Data: I, K, T, h_{it}
Time Limit
Extension Size for Partial Problem
Initial time-point set: T'
Output: Heuristic Solution

- 1 **Step 1:** Calculate Lagrangian lower bound (*Lower Bound*).
- 2 **Step 2:** Initialize best solution as all flights assigned to the first workstation.
- 3 **Step 3:** Initialize the genetic algorithm with T' .
- 4 **Step 4:** Iterate with the Focus-Search-Improve procedure until either the lower bound or time limit has been reached.
- 5 **while** Best objective on $T > \text{Lower Bound}$ **and** Elapsed Time $<$ Time Limit **do**
- 6 **Step 4.1:** Iterate the genetic algorithm.
- 7 **Step 4.2:** Update the best known solution with the genetic algorithm's last solution pool, evaluated on the full time-point set T .
- 8 **Step 4.3:** If the genetic algorithm's best objective value on T' is less than the best known objective calculated on full set T , extend the time-point set T' according to Algorithm A.2 and extension size.
- 9 **Step 5:** Return the best known solution for the objective calculated on full set T .
- 10 **return** Best Solution on T .

A.2 Focus Procedure of the Algorithm

Input: Instance Data: T , T'

Extension Size

Solutions from Algorithm A.3

Output: Updated set of times T'

- 1 **Step 1:** Select the solutions with best (minimum) objective values in the solutions from Algorithm A.3
 - 2 **Step 2:** For each solution, calculate the workload balance error at each time-point not included in T' as the difference between the current objective value and the objective value when that time-point is included.
 - 3 **Step 3:** Calculate the total workload balance error at each time-point as the sum of balance errors of individual solutions.
 - 4 **Step 4:** If all time-points have zero workload balance errors, end the procedure and return the current T'
 - 5 **Step 5:** Select a time-point with the highest total error, breaking ties arbitrarily.
 - 6 **Step 6:** Add the selected time-point to the set T' .
 - 7 **Step 7:** Reduce the Extension Size by 1. If becomes zero, return T' and end the procedure.
 - 8 **Step 8:** Run the Search procedure in Algorithm A.3 for 2 iterations to tune the current population.
 - 9 **Step 9:** Reapply the procedure from the beginning.
-

A.3 Search Procedure of the FSI-GA

Input: Instance Data: I, K, T'
Algorithm Parameters: Iteration Limit, Population Size, C^{elite} ,
Best known objective on T
Output: Feasible Solutions improved for T'

- 1 **Step 1:** Initialize the population randomly.
- 2 **Step 2:** Iterate until the limit has been reached.
- 3 **for** $\text{Iteration} = 1$ **to** Iteration Limit **do**
- 4 **Step 2.1:** Calculate the population objective values on T' .
- 5 **Step 2.2:** Update the best solution found so far (on T'). If the best objective
 on T' is lower than best known objective on T , return the current population.
- 6 **Step 2.3:** Initialize the next population as empty.
- 7 **Step 2.4:** Select C^{elite} -many elites from the population and add them to the
 next population.
- 8 **Step 2.5:** Select parent pairs, then perform crossover and mutation on each
 pair to obtain the offspring.
- 9 **Step 2.6:** Apply the Improvement procedure in Algorithm [A.4](#) to each
 offspring.
- 10 **Step 2.7:** Add the offspring to the next population. Replace the current
 population with the next population.
- 11 **Step 3:** Return the last population.
- 12 **return** Population.

A.4 Improvement Procedure of the FSI-GA

Input: Feasible Solution: *Solution*
Output: Improved Solution: *Solution*

- 1 **Step 1:** Iterate over all flight pairs.
- 2 **foreach** *Flight i* $\in$ *Solution* **do**
- 3 **foreach** *Flight j* $\in$ *Solution* **do**
- 4 **Step 2.1:** Evaluate the balance measure over time-points where either
 Flight *i* or *j* is active, store the balance value as *Current Imbalance*.
- 5 **Step 2.2:** Swap the workstation assignments of Flight *i* and *j*, evaluate the
 balance measure over the same set of time-points, store the balance value
 as *Swapped Imbalance*.
- 6 **Step 2.3:** If *Current Imbalance* is greater (worse) than *Swapped Imbalance*,
 accept the swap and apply the improvement procedure from the beginning
 with the new solution.
- 7 **Step 2.4:** If *Current Imbalance* is less than or equal to *Swapped Imbalance*,
 revert the swap and continue with the iterations.
- 8 **Step 3:** Return the solution only when no improvements are identified.
- 9 **return** *Solution*.
