Les Cahiers du GERAD ISSN: 0711–2440

Improving regularity in the crew pairing: A hybrid bonus-based and MIP optimizer approach

M. F. Benammour, F. Quesnel, B. Rochefort, F. Soumis

G–2025–81

December 2025

La publication de ces rapports de recherche est rendue possible grâce au soutien de HEC Montréal, Polytechnique Montréal, Université McGill, Université du Québec à Montréal, ainsi que du Fonds de recherche du Québec – Nature et technologies.

Dépôt légal – Bibliothèque et Archives nationales du Québec, 2025
– Bibliothèque et Archives Canada, 2025

The publication of these research reports is made possible thanks to the support of HEC Montréal, Polytechnique Montréal, McGill University, Université du Québec à Montréal, as well as the Fonds de recherche du Québec – Nature et technologies.

Legal deposit – Bibliothèque et Archives nationales du Québec, 2025
– Library and Archives Canada, 2025

Improving regularity in the crew pairing: A hybrid bonus-based and MIP optimizer approach

Mohamed Faouzi Benammour [a, c]
Frédéric Quesnel [b, c]
Benoît Rochefort [a, c]
François Soumis [a, c]

[a] *Bibliothèque, HEC Montréal, Montréal (Qc), Canada, H3T 2A7*

[b] *HEC Montréal, Montréal (Qc), Canada, H3T 2A7*

[c] *GERAD, Montréal (Qc), Canada, H3T 1J4*

benammour.m.f@gmail.com
quesnel.frederic@uqam.ca
benoit.rochefort@gerad.ca

December 2025
Les Cahiers du GERAD
G–2025–81

Abstract : The Crew Pairing Problem (CPP) involves constructing feasible pairings (sequences of flights, connections, and rest periods) for airline crew while minimizing operational costs and ensuring full flight coverage. However, beyond cost minimization, regularity has become a crucial objective for many airlines, as it enhances operational stability and may reduce indirect logistical costs. Traditional CPP optimization methods primarily focus on cost reduction but often fail to effectively incorporate regularity, as they lack a structured mechanism to balance both objectives.

This study proposes several approaches to enhance regularity in the CPP. The first approach is bonus-based, identifying pairings with high repetition potential in the initial solution. The CPP is then re-solved using a branch-and-price algorithm to encourage the selection of these repeatable pairings. The second approach reoptimizes the initial solution by solving an *MIP* that explicitly maximizes regularity. Finally, we show that both approaches are complementary and can be applied sequentially to further enhance regularity while maintaining cost efficiency, forming a hybrid method.

Computational experiments conducted on datasets from a major Asian airline demonstrate the effectiveness of the proposed methods. The results show that they generate highly regular solutions with minimal cost deviations, achieving up to an 88% improvement in regularity ratio and a 58% reduction in pairing patterns. This significant improvement highlights the practical applicability of the methods in optimizing airline crew pairing.

1 Introduction

Airlines continuously strive to ensure that each flight is equipped with the appropriate aircraft and crew, and that these resources are positioned at the right location and time to execute their flight schedules. However, this process is inherently complex and costly, involving multiple interdependent planning stages. Airline operations are typically planned in five steps. The first is *flight scheduling*, where flight schedules are determined with the objective of maximizing profit. This is followed by *fleet assignment*, which allocates aircraft types to scheduled flights, taking into account passenger demand, aircraft availability by type, and fleet conservation requirements. Next is *aircraft routing*, which constructs feasible itineraries for aircraft to cover each flight exactly once while adhering to maintenance requirements. The fourth step is *crew pairing*, which seeks to create minimum-cost crew pairings (sequences of flights, connections, and rest periods that comply with regulatory and contractual constraints) that cover all flights for a roster period (typically a month). Finally, *crew assignment* ensures that rosters are allocated to crew members in a manner that effectively covers all pairings. These planning steps collectively form the backbone of efficient airline operations. Among these steps, crew pairing optimization is particularly critical, as it directly impacts operational costs. A major challenge for airlines lies in minimizing crew-related expenses, which represent one of the largest operational cost components.

The objective of the *Crew Pairing Problem (CPP)* is to assign each flight exactly once to a pairing, ensuring feasibility under operational and regulatory constraints while minimizing operational costs. Pairing feasibility constraints include safety regulations, such as maximum duty and flight hours, and minimum rest periods. Contractual agreements impose additional regulations and a complex cost structure, adding significant complexity to the problem.

While minimizing costs remains a primary goal, achieving regularity in the CPP has gained increasing attention due to its significant operational and financial benefits. Regularity refers to the consistent repetition of identical pairings over multiple days, ensuring stability and predictability throughout a planning period, such as a month. Regular crew pairings provide numerous advantages, particularly in reducing indirect and administrative costs. By maintaining consistency, airlines can negotiate long-term contracts for accommodations and transportation services, such as hotels and limousines, leading to substantial savings compared to irregular, ad hoc arrangements. Additionally, predictable pairings improve employee satisfaction by offering more stable work schedules and better work-life balance. From an operational perspective, regular pairings improve efficiency by streamlining transitions between terminals and optimizing crew movement across consistent routes, especially during tight connections. Moreover, they enhance resilience during disruptions such as flight delays. By leveraging repeated pairings, airlines can apply proven solutions to recurring issues, avoiding the need to devise new responses each time. This combination of financial, operational, and employee-related benefits makes regularity an essential consideration in modern crew pairing strategies.

The CPP is usually solved using a branch-and-price algorithm (column generation embedded in a branch-and-bound algorithm). Such algorithms face challenges in incorporating regularity due to several factors. First, column generation methods typically treat flights as independent, whereas regularity relies on the presence of identical flights across different dates. In this context, flights that share the same operational characteristics (same origin, destination, and scheduled times) but occur on different days are grouped under the same *flight number*. Secondly, regularity is a global objective that involves the set of selected pairings. Including such an objective in the column generation subproblems would therefore be challenging. Additionally, the dependence of regularity on the structure of the entire set of pairings makes it challenging to evaluate the impact of a single pairing on overall regularity in the pairing subproblem. One way to accelerate column generation while enhancing regularity is through the three-phase method (Barnhart, Cohn, et al. 2003; Saddoune, Desaulniers, et al. Mar. 2013). This approach sequentially solves a daily, weekly, and monthly version of the CPP, using each solution as the initial input for the subsequent phase. While it generally achieves good regularity, it often struggles with cost efficiency. Conversely, the rolling horizon approach Saddoune, Desaulniers, et al. July 2009, which divides the planning period into overlapping time windows and solves each sequentially, typically minimizes costs but frequently results in irregular pairings. To overcome these limitations, this study proposes methods that aim to balance regularity and cost efficiency, providing airlines with practical and adaptable tools to optimize both objectives.

The contributions of this paper are as follows:

- We propose two solution methods that solve the CPP while explicitly taking regularity into account. Each algorithm takes as input an initial CPP solution and attempts to improve its regularity while keeping costs low.
- The first solution method identifies pairings with high repetition potential in the initial solution. The CPP is then re-solved using a branch-and-price algorithm with these repeatable pairings given as suggestions.
- The second solution method formulates the CPP as a bi-objective *MIP* whose variables are copies of the pairings in the initial solution. The *MIP* aims to find a solution that achieves a good balance between cost and regularity. This model is relatively small and can be solved quickly.
- We show that both algorithms are complementary in terms of solution improvement. We thus propose a hybrid approach that combines the two.
- We assess the regularity of a CPP solution using two metrics. The first, drawn from industry practice, is intuitive but tends to overemphasize rarely repeated pairing patterns. To address this limitation, we propose an alternative metric that corrects this bias and justifies its use both theoretically and empirically. Both metrics can be integrated into the optimization models developed.

The proposed methods are tested on a real-world dataset from a large Asian airline, achieving an average improvement of 40% to 60% in regularity with minimal cost deviations from the initial solution.

The remainder of this paper is organized as follows: Section 2 reviews the relevant literature, highlighting the key contributions and gaps in existing approaches. Section 3 introduces the problem definition, focusing on optimizing regularity in the CPP. We present the mathematical model and the associated constraints. Section 4 discusses the solution approaches, including methodologies designed to incorporate regularity. Section 5 presents computational results to evaluate the effectiveness of the proposed methods. Finally, Section 6 concludes the research, summarizing the findings and suggesting directions for future work.

2 Literature review

The Crew Pairing Problem (CPP) has been widely studied in operations research, with a focus on optimizing crew pairings to minimize costs while ensuring full flight coverage. This section reviews key contributions and limitations in CPP research, highlighting the evolution of solution approaches and their impact on cost efficiency and operational regularity.

2.1 Crew Pairing Problem CPP

The Crew Pairing Problem (CPP) is generally formulated as a *set partitioning problem (SPP)* or a *set covering problem (SCP)*, depending on the specific constraints for each flight, along with other secondary constraints. Each variable in the problem represents a feasible pairing. Considering a set of flights $\mathcal{F}$ over a given period, usually a month, the objective is to determine a set of feasible pairings for the crew that minimizes costs while ensuring that each flight $f \in \mathcal{F}$ is covered exactly once. Using the following notations:

- $\mathcal{P}$: Set of all feasible pairings.
- $\mathcal{F}$: Set of flights to be covered.
- $\mathcal{Q}$: Set of indices for secondary constraints.
- c_p: Cost of pairing $p \in \mathcal{P}$.
- a_{fp}: Binary constant equal to 1 if flight $f \in \mathcal{F}$ is covered in pairing $p \in \mathcal{P}$, and 0 otherwise.
- b_{qp}: Contribution of pairing $p \in \mathcal{P}$ to secondary constraint $q \in \mathcal{Q}$.
- e_q: Right-hand side of the secondary constraint $q \in \mathcal{Q}$.
- y_p: Binary variable equal to 1 if pairing p is selected and 0 otherwise.

The CPP is formulated as follows (see Saddoune, Desaulniers, et al. Mar. 2013):

$$\begin{aligned}
&\text{Minimize} && \sum_{p\in\mathcal{P}} c_p y_p && && (1)\\
&\text{subject to} && \sum_{p\in\mathcal{P}} a_{fp} y_p = 1, && \forall f \in \mathcal{F} && (2)\\
& && \sum_{p\in\mathcal{P}} b_{qp} y_p \le e_q, && \forall q \in \mathcal{Q} && (3)\\
& && y_p \in \{0,1\}, && \forall p \in \mathcal{P} && (4)
\end{aligned}$$

Objective function (1) minimizes the total cost of the selected pairings. Constraint (2) ensures that each flight $f \in \mathcal{F}$ is covered exactly once by one of the selected pairings. Constraint (3) enforces additional secondary requirements, such as hotel accommodations, crew availability at each base, and duty day limitations (see Sandhu et al. 2007; Quesnel, Desaulniers, et al. 2017; Quesnel, Desaulniers, et al. 2020). Finally, Constraint (4) ensures that y_p is binary.

2.2 CPP solution methods

Solving formulation (1)–(4) poses significant challenges due to the large number of feasible pairings. For small instances, Hu et al. 1999 proposes enumerating all feasible pairings, but this method becomes impractical for larger instances typical of the industry. To address this limitation, Klabjan et al. 2001 suggests focusing on a subset of promising pairings. However, this technique may occasionally omit pairings that appear of low quality but are essential for high-quality solutions (more details on this work are provided in 2.3). As the size and complexity of instances increase, more scalable techniques are required. *Column generation* has become the dominant approach, offering a structured framework for tackling large-scale CPPs efficiently (see Quesnel, Desaulniers, et al. 2017; Saddoune, Desaulniers, et al. Mar. 2013; Tahir et al. 2021).

2.2.1 Column generation

Column generation is an iterative method introduced by Dantzig et al. 1960 and first applied by Gilmore et al. 1961, illustrated by Lavoie et al. 1988, Desaulniers, Desrosiers, Ioachim, et al. 1998, Desaulniers, Desrosiers, and Solomon 2005 and Saddoune, Desaulniers, et al. Mar. 2013. Its goal is to solve the linear relaxation of optimization problems with a large number of variables.

Column generation alternates between solving a Restricted Master Problem (RMP) and one or more subproblems (see Figure 1). The RMP is the linear relaxation of (1)–(4), restricted to a limited number of variables (also called columns). The goal of the subproblems is to identify columns with negative reduced costs, which are then added to the RMP. This iterative process continues until the subproblems can no longer generate columns with negative reduced costs, indicating that the RMP solution is optimal for the linear relaxation of the global problem. The key advantage of this method

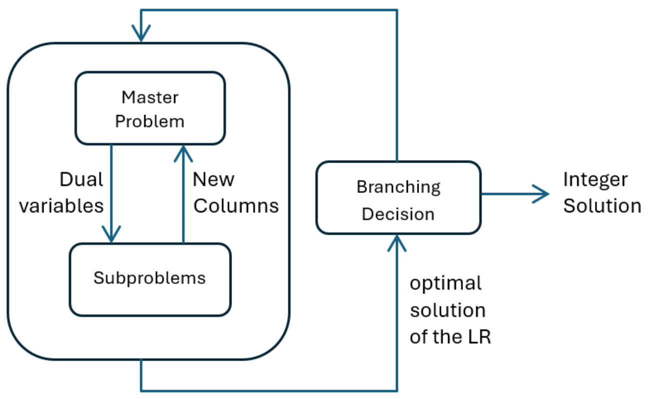

Figure 1: Column Generation Algorithm Saddoune 2010

lies in its ability to address all necessary pairings while focusing efficiently and explicitly on a significant subset of these pairings.

Desaulniers, Desrosiers, Dumas, et al. 1997 proposes a column-generation-based solution method for the CPP. Several other papers that use column generation to solve the CPP (see Desrosiers, Dumas, Desrochers, et al. 1991, Barnhart, Johnson, Anbil, et al. 1994, Desaulniers, Desrosiers, Dumas, et al. 1997, and Desaulniers, Desrosiers, Ioachim, et al. 1998) have been published. Barnhart, Johnson, Nemhauser, et al. 1998 highlighted the method's efficiency in large-scale instances. Column generation is still the most commonly used algorithm to solve the CPP linear relaxation (see, for instance, Zeren et al. 2016; Quesnel, Desaulniers, et al. 2020; Parmentier et al. 2020; Tahir et al. 2021).

For the CPP, the subproblems are usually formulated as *shortest path problems with resource constraints*, as detailed by Irnich et al. 2005. These subproblems are solved using dynamic programming and are represented by complex acyclic networks. Each subproblem is defined for a crew base and a specific day within the planning horizon, with the objective of generating feasible pairings for crew members based at a particular location. The network configuration depends on the cost structure of each pairing and the applied validity rules. Two distinct modeling methods emerge: the first relies on flight-based networks, as applied by Sandhu et al. 2007 and Saddoune, Desaulniers, et al. July 2009, where nodes represent flight beginnings and ends, connected by arcs linking sequentially feasible flights; the second involves duty-based networks, as proposed by Lavoie et al. 1988, where nodes represent duties (sequence of flight forming a day of work) and arcs connect operationally feasible duties. In duty-based networks, the set of all duties is generally very large. As such, a large subset of promising duties is generated in preliminary steps, and only those duties are considered in the subproblems.

When column generation yields fractional solutions, different strategies can be employed to obtain integer solutions. Muter et al. 2013 limit column generation to the root node, relying on integer programming for column selection, though this may lead to higher costs or infeasibility. A more robust method is the branch-and-price (B&P) algorithm, which integrates column generation into a branch-and-bound tree. This method generates new columns at each branching node, improving the quality of the integer solution. The algorithm was originally introduced in Desrosiers, Dumas, Solomon, et al. Oct. 1995, and the term *"branch-and-price"* was later coined by Barnhart, Johnson, Nemhauser, et al. 1998. B&P was successfully implemented by Desaulniers, Desrosiers, Dumas, et al. 1997, who solved instances with up to 1,200 flights in under four hours. Further enhancements and branching strategies were later developed by Desaulniers, Desrosiers, Ioachim, et al. 1998. Typically, fully exploring the branch-and-bound tree would be too computationally demanding, so heuristic branching, such as column fixing, is used (Saddoune, Desaulniers, et al. July 2009).

Several alternative approaches have also been explored to improve the efficiency and scalability of CPP solution methods. Some studies combine column generation with other techniques, such as

Benders decomposition (Zeighami et al. 2019) or dynamic aggregation methods (Elhallaoui et al. 2005; Desaulniers, Lessard, et al. 2020). Notably, Desaulniers, Lessard, et al. 2020 applied dynamic constraint aggregation to solve monthly problems involving over 46,000 flights. Yaakoubi et al. 2020 proposes improvements to the latter by incorporating machine learning, applied to monthly CPP instances involving up to 50,000 flights. Other hybrid methods combine integer programming with metaheuristics, such as particle swarm optimization PSO (Mohamed et al. 2020) or supervised machine learning with mixed-integer optimization (Hizir 2024). Metaheuristics have also been used independently—for instance, variable neighborhood search VNS (Xu et al. 2021) and genetic algorithms GA (Kornilakis et al. 2002).

Finally, acceleration methods have been developed for both regular and irregular flight schedules. Barnhart, Cohn, et al. 2003 introduced a three-phase approach for regular flight schedules, which solves the CPP sequentially over three levels: daily, weekly, and monthly. It begins by solving a representative daily problem based on typical flights that operate at least three times per week. The resulting cyclic pairings are then expanded into a weekly schedule, followed by a re-optimization phase that accounts for operational constraints and irregularities. A final monthly phase further refines the plan to ensure feasibility over the entire planning horizon. This approach is effective for multi-day pairings and regular schedules, but is less suitable when the flight schedules are irregular.

For irregular schedules, Saddoune, Desaulniers, et al. July 2009 proposed a *rolling horizon* method, which divides the planning horizon into overlapping time windows of length L, with overlap O, and solves each subproblem sequentially using column generation. Initial conditions are propagated from one window to the next to ensure continuity. By reducing the size of the problem and focusing on smaller sub-instances, this method achieves better results in terms of computation time, particularly for large-scale monthly problems.

For comprehensive literature reviews, see Barnhart, Cohn, et al. 2003; Desaulniers, Desrosiers, and Solomon 2005; Kasirzadeh 2015; Quesnel 2019.

2.3 Regularity

Although few studies have directly addressed the concept of regularity in crew pairing, several works have explored related notions under different definitions and modeling approaches.

First, some studies define regularity as the repetition of the same sequence of activities. In the context of the CPP, this would translate to how frequently the same pairings are repeated across the planning horizon. This is the interpretation considered in this paper. Tajima et al. 1997 describes an airline crew scheduling problem involving many irregular flights. The authors propose a heuristic that systematically merges irregular flights into pairings consisting exclusively of regular flights. Although the primary objective was to minimize the number of man-days in the solution, the approach also manages to produce regular crew schedules. However, the authors do not report the impact on operational costs. Regularity is measured as the percentage of pairings that are repeated every day of the planning horizon. This measure was not considered in this study because it has several flaws. For instance, it ignores pairings that repeat almost every day of the horizon. However, such pairings can greatly contribute to regularity. By contrast, this paper introduces a *regularity ratio measure* which accounts for the level of repetition of each pairing.

Klabjan et al. 2001 proposes a weekly model that simultaneously considers cost and regularity in a weekly schedule. Their method generates a vast array of random pairings (in the millions), which are then incorporated into a linear programming problem. The problem is solved in several phases, each focusing on g-regular pairings (i.e., pairings that can be repeated on g days per week), starting from the highest regularity (7-regular) down to 4-regular. After these stages, irregular pairings were generated, and the complete flight schedule was partitioned accordingly. All flights not assigned to a g-regular group, where $4 \leq g \leq 7$, were classified as irregular flights and had to be covered by irregular pairings. In their model, penalty costs were assigned to irregular pairings, decreasing as regularity increased. In their study, computing time reached 40 hours for small-scale problems (492 flights per week). Regularity was measured by the number of flight segments assigned to the 1-regular group—the fewer such segments, the higher the regularity. Since the method relies on a subset of enumerated pairings, it may overlook certain pairings that appear suboptimal but are essential for constructing a high-quality solution. For the same reason, applying this method to larger datasets could become prohibitively time-consuming as the number of enumerated pairings increases.

Wu et al. 2016 develops a column generation approach for CPP to minimize the total person-days. They propose an algorithm for finding good pairings and consider regularity based on the fact that most flight legs are regularly scheduled. Regularity is exploited to quickly generate many columns just by repeating itineraries. Such repeated itineraries are preferable so that crew members will have a low risk of making mistakes in duty time. The authors define two pairings as twins if for every corresponding pair of flight legs, the departure and arrival airports are the same, and the departure and arrival times are the same but not on the same day. The idea is that when the column generation finds a valid pairing, all its twin pairings are added. They confirm that this idea improves the speed of the column generation process.

Other works interpret regularity as temporal consistency in the crew's work patterns, regardless of task content. Dillon et al. 1999 presents an approach for pilot reserve crew scheduling that generates reserve duty patterns (reserve pairings), which are then allocated using preferential bidding. The study emphasizes quality-of-life considerations, particularly the regularity of monthly schedules (crew like to work the same days of the week on duty across all weeks in the month). They generate call-out day pairings of varying lengths, which also exhibit regularity. Note that Dillon et al. 1999 considers a different aspect of regularity than this paper. Whereas they focus on schedule regularity (days on, days off patterns), we focus on pairing regularity (the repetition of the same pairing patterns). Although this was not studied in this paper, it seems intuitive that more regular pairings would allow for more regular schedules.

Gopalakrishnan et al. 2005 discusses regularity in the context of crew fatigue, noting that while many rules exist to address sleep loss in aircrew, there are no specific regulations aimed at minimizing irregular sleep patterns. The authors suggest that it may be possible to optimize over legal pairings that exhibit more regularity in sleep schedules without significantly increasing crew costs. They also report preliminary computational experiments, which indicate promising results in reducing crew fatigue through improved regularity at the crew pairing optimization stage. The basic idea behind their approach is to use the intersection of rest periods in a pairing as a measure of regularity of sleep for the air crew.

Steinzen et al. 2009, in the context of public transport, propose a solution method to improve regularity in ex-urban bus networks, particularly addressing irregularities that are undesirable from the drivers' perspective. Their method also partially integrates the vehicle and crew scheduling problems. A computational study showed that the proposed method improves solutions in both cost and regularity compared to traditional approaches. However, a current limitation of this approach is that it does not consider a full integration of vehicle and crew scheduling. To evaluate the improvement in regularity, the measures used include the percentage of preserved duties, which indicates how many duties from the original schedule are exactly retained in the new schedule. The percentage of preserved regular pairs reflects how many regular pairings are maintained. Another measure is the average regular chain length, which represents the mean number of regular tasks per duty. The percentage of average chain length compares this value in the new solution to that in the original schedule.

In their computational study, the authors report that the proposed method improves both cost and regularity compared to traditional approaches. However, they also acknowledge that the current version of their method does not yet support fully integrated vehicle and crew scheduling, which remains a limitation.

Amberg et al. 2011 proposes a set of methods to enhance the similarity of resource schedules in public transport, addressing the limitations of traditional approaches that often produce highly irregular schedules by considering only one planning day at a time. The authors introduce two main strategies to improve similarity: the first solves each day's scheduling problem independently while enforcing proximity to a common reference schedule; the second solves the scheduling problems for multiple days simultaneously by leveraging regular patterns to ensure similarity. They also propose a *MIP* formulation in which patterns are used as variables, solved using a column generation approach. Computational results show that the proposed methods can significantly improve schedule similarity, with only a minor increase in cost compared to fully cost-optimal solutions.

3 Problem definition

This section outlines the problem addressed in this study. It begins by defining regularity within the Crew Pairing Problem (CPP) and emphasizing its operational significance. The measures used to quantify and evaluate regularity in crew pairing solutions are also discussed. Finally, the mathematical model developed to integrate regularity into the CPP is presented.

3.1 Terminology

Understanding the terminology related to crew pairing and operations management is essential for grasping the complexities of the *CPP*. *Crew members* are responsible for operating flights and ensuring passenger safety and comfort. Depending on their roles, crew members may include pilots (e.g., captain, first officer) and cabin staff (e.g., flight attendants). Each crew member is assigned to a specific *crew base*, serving as their home airport.

A *flight* is defined by its flight number, origin and destination airports, and scheduled times. It is typically operated by the same aircraft and crew.

A *flight number* corresponds to a set of flights that share the same operational characteristics (same origin, destination, and scheduled times) but occur on different days.

A *pairing* is defined as a sequence of flights, connections (time on ground) separated by mandatory rest periods. All pairings must start and end at the same crew base and are subject to numerous validity rules. Additionally, crew members may travel as passengers for repositioning purposes, which is referred to as *deadheading*. Between consecutive flights/deadheads, a *sit connection* serves as a waiting period during which crew members prepare for their next segment. A *duty* represents a typical workday for a crew member, comprising flights separated by connections, potentially followed by a *layover* or rest period. The goal of the crew pairing problem is to select a set of valid pairings that covers each flight exactly once, at minimum cost.

Finally, a *pairing pattern* is a pairing with flights replaced by corresponding flight numbers. Thus, several pairings can belong to the same pairing pattern, and two such pairings are identical except for their start date.

3.2 Regularity in the Crew Pairing Problem

Pairing regularity is characterized by the consistent repetition of identical pairings across multiple days. Let $\mathcal{F}$ be the set of flights to cover, and $\mathcal{N}$ the associated set of flight numbers. Let $\bar{\mathcal{P}}$ be a set of pairings forming a valid CPP solution, and let $\bar{\mathcal{T}}$ denote the set of pairing patterns used in that solution. This paper uses two different metrics to evaluate the regularity of a solution.

The first measure, called *Total Number of Pairing Patterns ($M_{pattern}$)*, corresponds to the number of unique pairing patterns used in a solution. We thus have:

$$M_{\text{pattern}} = |\bar{\mathcal{T}}| \tag{5}$$

A smaller M_{pattern} value indicates greater regularity, as fewer distinct patterns suggest a more consistent schedule. This measure is widely used by airlines as it is simple to understand.

While simple and intuitive, $M_{pattern}$ inherently tends to favor longer pairings when used as an optimization objective, as fewer patterns are required when pairings span more flights. Another drawback of this measure is that it emphasizes patterns that are less frequently repeated. For example, consider two solutions, each composed of 34 pairings. In the first solution, the pairings are distributed across five pairing patterns with frequencies (30–1–1–1–1), while in the second, they are distributed as (16–16–2) across three pairing patterns. Although the first solution appears more regular (i.e, most pairings follow the same pattern), the Total Number of Patterns measure would assign it a higher score (five patterns versus three), incorrectly suggesting lower regularity. This highlights the limitation of this measure, which treats all patterns equally, regardless of their usage frequency.

To address these shortcomings, we introduce a new metric to measure the regularity of a CPP solution, named the *ratio-based measure (M_{ratio})*. This measure evaluates how consistently a flight is repeated through the same pattern over the planning horizon, on average. For flight $f \in \mathcal{F}$, let $\mathcal{T}_f$ denote the set of all pairings in the solution that use the same pattern as f, in Solution $\bar{P}$, and let $\mathcal{G}_f$ denote the set of all flights with the same flight number as f (or the set of all flight associated with flight number $f \in \mathcal{N}$, depending on context). The *regularity ratio* of flight f is then defined as:

$$R_f = \frac{|\mathcal{T}_f|}{|\mathcal{G}_f|}. \tag{6}$$

R_f thus takes the value 1 if all flights with the same flight number as f belong to pairings following the same pairing pattern. Note that R_f is defined for flights rather than flight numbers. Two different flights with the same flight number can have different R_f values if they belong to pairings following different pairing patterns.

Building on this concept, we define M_{ratio} as the average regularity ratio across all flights:

$$M_{\text{ratio}} = \frac{1}{|\mathcal{F}|} \sum_{f \in \mathcal{F}} R_f. \tag{7}$$

This measure can be equivalently reformulated as a summation over the pairing patterns in a solution. Let $\mathcal{F}_t \subset \mathcal{F}$ and $\mathcal{N}_t$ be the set of flights and flight numbers associated with pattern $t \in \bar{\mathcal{T}}$, respectively. The regularity ratio of pattern t, denoted R_t, is defined as the sum of the regularity ratios of its flights:

$$R_t = \sum_{f \in \mathcal{F}_t} R_f. \tag{8}$$

Then, the overall regularity measure can be rewritten as:

$$M_{\text{ratio}} = \frac{1}{|\mathcal{F}|} \sum_{t \in \bar{\mathcal{T}}} R_t. \tag{9}$$

M_{ratio} is bounded between 0 and 1, and a higher value indicates that flights are more consistently assigned to the same pairing patterns, which reflects greater regularity. It can be viewed as a percentage, making it easier to interpret and compare across different cases.

We now reformulate M_{ratio} in a way that is more suitable for integration into a mathematical model. Let z_t^* denote the number of times pairing pattern $t \in \bar{\mathcal{T}}$ is used in solution $\bar{\mathcal{P}}$. For each flight $f \in \mathcal{F}_t$, we therefore have $|\mathcal{T}_f| = z_t^*$. R_f can then be reformulated as :

$$R_t = \sum_{f \in \mathcal{F}_t} R_f = \sum_{f \in \mathcal{F}_t} \frac{|\mathcal{T}_f|}{|\mathcal{G}_f|} = z_t^* \cdot \sum_{f \in \mathcal{F}_t} \frac{1}{|\mathcal{G}_f|} = z_t^{*2} \cdot \sum_{n \in \mathcal{N}_t} \frac{1}{|\mathcal{G}_n|} \tag{10}$$

Where the last equality is obtained by noticing that the flight of $\mathcal{F}_t$ can be grouped by flight number, each flight number being represented z_t^* times. For the sake of simplicity, this derivation assumes that each flight number can be at most once per pairing, but the results are easy to generalize if that were not the case. Let $\Gamma(t) = \sum_{n \in \mathcal{N}_t} \frac{1}{|\mathcal{G}_n|}$. Let $\mathcal{T}$ denote all possible pairing pattern, and suppose $z_t^* = 0$ if and only if $t \notin \bar{\mathcal{T}}$. The regularity ratio can then be expressed as

$$M_{\text{ratio}} = \frac{1}{|\mathcal{F}|} \sum_{t \in \mathcal{T}} z_t^{*2} \Gamma(t) \tag{11}$$

Finally, the factor $\frac{1}{|F|}$ is constant for a given instance and can be ignored for optimization purposes.

Note that both metrics play a critical role in evaluating the regularity of optimized solutions. $M_{pattern}$ offers a simple and intuitive measure of regularity, favoring solutions with fewer and longer pairings. In contrast, M_{ratio} captures a more nuanced understanding of how patterns align with flights, ensuring that regularity reflects both operational and structural aspects of the solution. As shown in Section 5.2, M_{ratio} can also be used as a regularity objective inside optimization algorithms. Together, these metrics offer comprehensive insights into crew pairing regularity, enabling airlines to choose the evaluation method that best aligns with their strategic goals and operational needs.

To further motivate the usefulness of M_{ratio}, Figure 2 compares two solutions for the same monthly instances. Even though both solutions use the same number of patterns (191), solution S2 is clearly more regular, as evidenced by the higher number of patterns that are repeated almost 30 times. Correspondingly, the regularity ratio of S2 (92.14) is much higher than that of solution S1 (79.13). This reinforces the idea that M_{pattern} can fail to capture important elements of regularity.

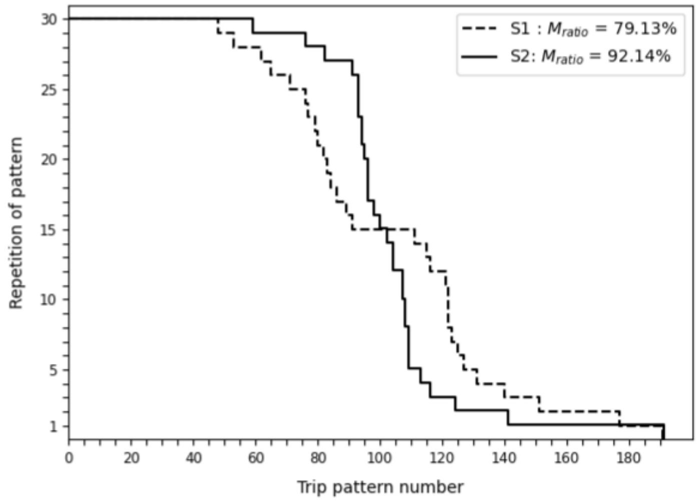

Figure 2: Pattern repetition for two solutions

3.3 Mathematical model

A naive method of achieving regularity would be to directly add a regularity component to the CPP objective function. Here, we present such a model and explain why solving it would be impractical. The proposed model explicitly incorporates pairing patterns, aiming to enhance the regularity metric by encouraging the repetition of these patterns. The following notation is used :

– $\mathcal{P}(t)$: The subset of pairings belonging to pattern $t \in \mathcal{T}$.
– c_p: Cost associated with pairing $p \in \mathcal{P}$.
– y_p: Binary variable taking value 1 if the pairing $p \in \mathcal{P}$ is selected, and 0 otherwise.

- z_t: Integer variable representing the number of repetitions of pattern $t \in \mathcal{T}$.
- $F(z_t)$: is a general bonus function dependent on z_t.

The proposed model is an integer program designed to solve the CCP with a focus on regularity. It is expressed as follows:

$$
\begin{aligned}
\text{Minimize} \quad & \sum_{p \in \mathcal{P}} c_p y_p + \sum_{t \in \mathcal{T}} F(z_t) && && (12)\\
\text{subject to} \quad & \sum_{p \in \mathcal{P}} a_{fp} y_p = 1, && \forall f \in \mathcal{F} && (13)\\
& \sum_{p \in \mathcal{P}} b_{qp} y_p \le e_q, && \forall q \in \mathcal{Q} && (14)\\
& \sum_{p \in \mathcal{P}(t)} y_p = z_t, && \forall t \in \mathcal{T} && (15)\\
& y_p \in \{0,1\}, && \forall p \in \mathcal{P} && (16)\\
& z_t \in \mathbb{N}, && \forall t \in \mathcal{T} && (17)
\end{aligned}
$$

Objective function (12) minimizes the total cost, which includes the cost of selected pairings and a regularity objective, with $F(z_t)$ designed to encourage regularity by rewarding patterns with high regularity. For instance, to consider the $M_{pattern}$ metric, we would have $F(z_t) = 0$ if $z_t = 0$, and M otherwise. To implement the M_{ratio} measure, we would set $F(z_t) = z_t^2 \Gamma(t)$. Of course, other functions could be considered, even if not directly linked to a regularity measure.

To illustrate how different forms of $F(z_t)$ can affect the solution, consider the case where we prefer selecting a single pattern repeated 20 times over two patterns repeated 10 times each. This preference reflects a desire for deeper repetition of fewer patterns, which promotes greater regularity. To enforce such behavior, a concave function can be used, such as $F(z_t) = -\alpha z_t^\beta$ with $\beta > 1$ and $\alpha > 0$. This functional form assigns increasing marginal bonuses as pattern usage increases, encouraging the selection of fewer, more frequently used patterns rather than many infrequently repeated ones. Note that a linear bonus function (i.e., $F(z_t) = -\alpha z_t$) would not incentivize regularity, as it would make the regularity component of the objective function proportional to the number of pairings.

Constraint (13), known as the flight coverage constraint, ensures that each flight $f \in \mathcal{F}$ is covered exactly once by the selected pairings. Constraint (14) enforces additional secondary requirements. Constraint (15) establishes the linkage between patterns and pairings by ensuring that the total number of pairings belonging to a pattern $t \in \mathcal{T}$ equals the variable z_t, which represents the number of times pattern t is repeated. Constraint (16) enforces the binary nature of the decision variable y_p. Finally, Constraint (17) requires z_t, the number of repetitions of pattern t, to be a non-negative integer, reflecting the discrete nature of pattern repetitions.

Solving (12)–(17) would be challenging for several reasons. The first challenge lies in the complexity of the potential non-linearity of the objective function (12). This nonlinearity leads to weak linear relaxations, making it difficult to derive tight bounds during optimization. Consequently, the branch-and-bound (B&B) process becomes more complex, requiring deeper exploration of the search tree and significantly increasing computational time. Depending on the functional form of $F(.)$, existing mathematical programming algorithms may not be directly applicable or may perform poorly, further complicating the resolution of the model.

The second challenge arises from linking constraints (15). There is one such constraint per pairing pattern, which can be millions in large-scale problems. If column generation were used to solve formulation (12)–(17), constraints (15) would need to be dynamically added to the problem. However, this would bring the additional difficulty of determining whether a given pattern is being generated in the pricing subproblems, to properly compute column reduced costs. This would also greatly complicate the dominance procedure that is key to efficient column generation (Irnich et al. 2005).

In summary, these challenges highlight the need for advanced solution techniques to address the complexities of both formulations. Strategies to avoid the nonlinearity of the objective function and efficiently manage linking constraints are key to ensuring practical and computationally efficient resolutions of the CPP with regularity.

4 Solution methods

The general approach proposed in this paper is to consider an initial CPP solution (referred to as the input solution) and improve its regularity while maintaining a low solution cost. This initial solution is obtained by solving the CPP using traditional techniques (e.g., branch-and-price). In practice, such a solution is generally available since the CPP is solved several times during a month to adjust the solution.

We propose two methods of improving the regularity of a given solution. The first approach identifies pairings with high repetition potential in the initial solution. The CPP is then re-solved using a branch-and-price algorithm, incorporating bonuses to encourage the selection of these highly repeatable pairings. This method is presented in Section 4.1. The second approach re-optimizes the initial solution by solving a Mixed-Integer Programming (MIP) model that explicitly targets regularity as an objective. This method is detailed in Section 4.2. Finally, both methods can be applied sequentially to further improve regularity while maintaining cost efficiency, forming a hybrid approach described in Section 4.3.

4.1 Bonus-based improvement

Model (12)–(17) poses challenges, particularly due to the nonlinearity introduced by the bonus function $F(z_t)$. To address this, we leverage the input solution to identify highly repeatable patterns and re-optimize the CPP by initializing it with pairings corresponding to these patterns, augmented with appropriate bonuses.

Let $\hat{\mathcal{P}}$ be the set of pairings in the input solution, and let z_t^* denote the number of times pattern $t \in \mathcal{T}$ is used in the input solution. These values provide useful indicators for identifying patterns of interest in the optimization process. By leveraging the z_t^* values and a bonus function F, the method assigns a bonuses $F(z_t^*, t)$ to pairings. The objective function is then reformulated by adding these bonuses to the cost of each pairing.

The updated optimization model is formulated as follows:

$$\text{Minimize} \quad \sum_{p\in\mathcal{P}} (c_p + F(z_t^*, t))\, y_p \tag{18}$$

$$\text{subject to} \quad \sum_{p\in\mathcal{P}} a_{fp} y_p = 1, \qquad \forall f \in \mathcal{F} \tag{19}$$

$$\sum_{p\in\mathcal{P}} b_{qp} y_p \le e_q, \qquad \forall q \in \mathcal{Q} \tag{20}$$

$$y_p \in \{0, 1\}, \qquad \forall p \in \mathcal{P} \tag{21}$$

This reformulation encourages the repetition of patterns selected in the input solution and promotes their selection in the new solution. In practice, this is achieved by only assigning bonuses to pairings identified as *regular*. A pairing is regular if its corresponding pattern is used at least $\underline{z}$ times in the input solution. The CPP is then re-optimized by solving the reformulated model using a *branch-and-price* algorithm. This approach sidesteps the non-linearity of Model (12)–(17) by considering static bonuses based on the input solution. We propose two bonus functions to enhance regularity, presented in the following subsections.

4.1.1 Exponential bonus function

This function prioritizes patterns based on their frequency of use, encouraging higher repetition of the same pattern. We define:

- α: A positive scaling parameter ($\alpha > 0$) that adjusts the magnitude of the bonus.
- β: An exponent parameter ($\beta > 1$) that controls the steepness of the function.

The bonus function is formulated as:

$$F(z_t^*, t) = -\alpha \cdot (z_t^*)^\beta \tag{22}$$

This bonus function is both negative and concave, ensuring that patterns used more frequently are given higher priority. The concavity of the function plays a critical role in its behavior, as it increases the marginal benefit of additional repetitions for frequently used patterns. For instance, a pattern repeated 20 times will receive a significantly higher bonus than two identical patterns repeated 10 times each, emphasizing the operational advantage of concentrating repetitions within fewer patterns.

4.1.2 Ratio-based bonus function

This bonus function prioritizes patterns by assigning a value based on their regularity ratio in the input solution. Let R_t^* represents the regularity ratio (R_t, Equation (8)) for pattern t from the input solution (see 3.2 for details). The proposed *ratio-based bonus* function is given by:

$$F(z_t^*, t) = -\alpha \cdot R_t^* = -\alpha z_t^{*2}\Gamma(t) \tag{23}$$

Here, α is a positive scaling parameter ($\alpha > 0$) that adjusts the bonus magnitude.

This approach rewards patterns by summing the regularity ratios of all their associated flights, naturally aligning with the concept of Ratio-Based measure (M_{ratio}, Equation (9)) as it emphasizes uniformity across flights within a pattern. It strongly incentivizes patterns with consistently high repetition rates across their flights.

4.1.3 Advantages and drawbacks

The Bonus-Based Improvement approach effectively enhances regularity in the CPP by using negative bonus functions to promote pattern repetition while maintaining cost efficiency. This method improves regularity without the complexity of solving nonlinear models. However, its effectiveness depends on the chosen parameters, such as the concavity factor and scaling coefficients. Poor parameter selection can result in either insufficient regularity improvements or non-productive solutions.

Another drawback is the computational time required. Re-optimizing the solution takes almost as much time as a new optimization. In actuality, re-optimization is slightly faster because the suggested pairings act as a warm start. To mitigate this, we propose several enhancements aimed at streamlining the process. In our implementation, some overhead steps, such as duty generation, can be performed only once, resulting in substantial time savings during re-optimization.

Finally, another limitation is that the method does not restructure solutions: It may encourage the repetition of similar conflicting patterns separately instead of consolidating them into a single, more frequent pattern. For instance, consider a scenario with four flights: A, B, C, and D, where flights A and C have nearly the same departure/arrival time, and similarly with flights B and C. Suppose the possible pairings are (AB, AD, CB, CD), each appearing 15 times. The bonus-based method would encourage all four pairings equally to increase their repetition. As a result, the optimization process has no incentive to prefer consolidating pairings into fewer, more dominant patterns. This limitation arises because the solution becomes a local minimum, with no mechanism to escape it. However, regularity could be enhanced by only reinforcing AB and CD, allowing them to each repeat 30 times instead. Addressing this issue would require additional post-processing techniques beyond the scope of this approach. Figure 3 illustrates this situation.

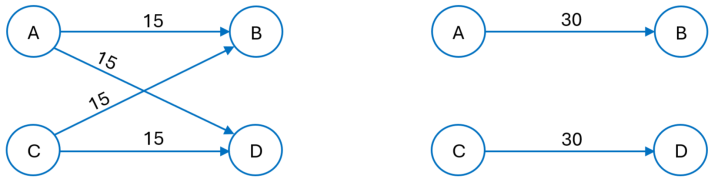

Figure 3: Illustration of alternative grouping strategies for pattern repetition

4.2 Integer Programming model

In this section, we present two Integer Programming (IP) models aimed at improving the regularity of crew pairing solutions while maintaining low costs. Both models rely on an input solution to provide a set of feasible pairings as well as a reference solution cost. Let $\tilde{\mathcal{T}} = \{t \in \mathcal{T} | z_t^* > 0\}$ be the set of all pairing pattern in the input solution. Each model considers a fixed set of pairings, denoted $\tilde{\mathcal{P}}$, which consists of all feasible pairings that correspond to a pattern in $\tilde{\mathcal{T}}$ (pairings of the initial solution plus all feasible copies matching their patterns). This has the effect of limiting the number of variables while allowing for increased regularity.

Each model is formulated as a bi-objective optimization problem, simultaneously maximizing regularity and minimizing total cost. They also contain a soft constraint to keep pairing costs at an acceptable level. In particular, a parameter r is introduced to control the maximum allowable deviation from the cost of the input solution.

The first and second models consider M_{pattern} and M_{ratio} as the regularity metric. In the second model, the nature of M_{ratio} (Equation (11)) makes the objective function quadratic. It is then linearized.

The following notations are used in both models:

– λ: a parameter weighted $0 \leq \lambda \leq 1$ that balances the trade-off between the two objectives, minimizing the total cost and maximizing the regularity,
– r: the authorized percentage increase in cost.

4.2.1 Minimizing the total number of patterns (MIPp)

The first model focuses on minimizing the total number of patterns used in the solution. To achieve this, the model uses z_t as a binary decision variable to indicate whether pattern $t \in \tilde{\mathcal{T}}$ is selected. The complete bi-objective model is formulated as follows:

$$\text{Minimize} \quad \lambda \sum_{p \in \tilde{\mathcal{P}}} c_p y_p + (1-\lambda) \sum_{t \in \tilde{\mathcal{T}}} z_t + M\gamma \tag{24}$$

$$\text{subject to} \quad \sum_{p \in \tilde{\mathcal{P}}} c_p y_p \leq (1+r) \sum_{p \in \tilde{\mathcal{P}}} c_p y_p^* + \gamma \tag{25}$$

$$\sum_{p \in \tilde{\mathcal{P}}} a_{fp} y_p = 1, \qquad \forall f \in \mathcal{F} \tag{26}$$

$$\sum_{p \in \tilde{\mathcal{P}}} b_{qp} y_p \leq e_q, \qquad \forall q \in \mathcal{Q} \tag{27}$$

$$y_p \leq z_t, \qquad \forall t \in \tilde{\mathcal{T}}, \forall p \in \tilde{\mathcal{P}}(t) \tag{28}$$

$$y_p \in \{0,1\}, \qquad \forall p \in \tilde{\mathcal{P}} \tag{29}$$
$$z_t \in \{0,1\}, \qquad \forall t \in \tilde{\mathcal{T}} \tag{30}$$

Cost constraint (25) ensures that the total cost of the improved solution does not exceed the cost of the initial solution by more than a specified margin r. It is implemented as a soft constraint, since it has been observed that in practice, exceeding the limit slightly can have a great impact on regularity. However, the unit penalty M is chosen so that it doesn't happen often in practice. The flight coverage constraint (26) guarantees that every flight in the set $\mathcal{F}$ is covered exactly once by a selected pairing. In practice, this constraint is also implemented as a soft constraint, which is not reflected in (24)–(30) for the sake of simplicity. Constraint (27) enforces the secondary requirements. The pattern-flight linkage constraint (28) establishes the dependency between pairings and patterns, enforcing that a pairing y_p can only be selected if the corresponding pattern z_t it belongs to is also included. Constraints (29) and (30) ensure the y_p and z_t variables are binary respectively. It is worth noting that since the variable set is fixed, the model can be solved using standard integer programming tools without relying on column generation. For this reason, linking constraints (28) do not pose the same issues as constraints (15).

4.2.2 Maximizing the cumulative regularity ratio (MIPr)

The second model incorporates the regularity ratio M_{ratio} (Equation (11)) in the objective function. It is formulated as :

$$\text{Maximize} \qquad -\lambda \sum_{p\in\tilde{\mathcal{P}}} c_p y_p + (1-\lambda)\sum_{t\in\tilde{\mathcal{T}}} z_t^2 \cdot \Gamma(t) - M\gamma \tag{31}$$
$$\text{subject to} \qquad \sum_{p\in\tilde{\mathcal{P}}} c_p y_p \leq (1+r)\sum_{p\in\tilde{\mathcal{P}}} c_p y_p^* + \gamma \tag{32}$$
$$\sum_{p\in\tilde{\mathcal{P}}} a_{fp} y_p = 1, \qquad \forall f \in \mathcal{F} \tag{33}$$
$$\sum_{p\in\tilde{\mathcal{P}}} b_{qp} y_p \leq e_q, \qquad \forall q \in \mathcal{Q} \tag{34}$$
$$\sum_{p\in\tilde{\mathcal{P}}(t)} y_p = z_t, \qquad \forall t \in \tilde{\mathcal{T}} \tag{35}$$
$$y_p \in \{0,1\}, \qquad \forall p \in \tilde{\mathcal{P}} \tag{36}$$
$$z_t \in \mathbb{N}, \qquad \forall t \in \tilde{\mathcal{T}} \tag{37}$$

Cost constraint (32) ensures that the total cost of the selected pairings does not exceed the cost of the initial solution by more than a specified margin r. The flight coverage constraint (33) guarantees that every flight $f \in \mathcal{F}$ is covered exactly once by one of the selected pairings. Constraints 32 and (33) are again implemented as soft constraints (only the former is depicted in the above formulation for the sake of simplicity). Constraint (34) enforces the secondary requirements. The pattern-pairing linkage constraint (35) establishes the dependency between pairings and patterns, enforcing that the total number of pairings y_p associated with a pattern $t \in \tilde{\mathcal{T}}$ must match the number of repetitions z_t of that pattern. Constraints (36) and (37) ensure that the y and z variables are binary and integer, respectively.

Objective function (31) is nonlinear due to the quadratic term z_t^2. To address this complexity, we introduce an alternative linearized model. Let $w_{t,i}$ be a decision variable that equals 1 if pattern t is used at least i times ($i \leq z_t$), and 0 otherwise. Let δ denote the total number of days in the planning period, which also corresponds to the maximum possible number of repetitions for a pattern. Using the well-known result:

$$k^2 = \sum_{i=1}^{k} (2i-1) \tag{38}$$

We reformulate the regularity objective as:

$$\sum_{t \in \tilde{\mathcal{T}}} z_t^2 \cdot \Gamma(t) = \sum_{t \in \tilde{\mathcal{T}}} \sum_{i=1}^{z_t} (2i-1) \cdot \Gamma(t) = \sum_{t \in \tilde{\mathcal{T}}} \sum_{i=1}^{\delta} (2i-1) \cdot w_{t,i} \cdot \Gamma(t) \tag{39}$$

Recall that $\mathcal{P}(t)$ is the subset of pairings associated with pattern $t \in \tilde{\mathcal{T}}$. Model (31)–(37) can be reformulated as :

$$\begin{aligned}
\text{Maximize} \quad & -\lambda \sum_{p \in \tilde{\mathcal{P}}} c_p y_p + (1-\lambda) \sum_{t \in \tilde{\mathcal{T}}} \sum_{i=1}^{\delta} (2i-1) \cdot \Gamma(t) \cdot w_{t,i} - M\gamma && (40)\\
\text{subject to} \quad & \sum_{p \in \tilde{\mathcal{P}}} c_p y_p \le (1+r) \sum_{p \in \tilde{\mathcal{P}}} c_p y_p^* + \gamma && (41)\\
& \sum_{p \in \tilde{\mathcal{P}}} a_{fp} y_p = 1, && \forall f \in \mathcal{F}, \quad (42)\\
& \sum_{p \in \tilde{\mathcal{P}}} b_{qp} y_p \le e_q, && \forall q \in \mathcal{Q}, \quad (43)\\
& \sum_{i=1}^{\delta} w_{t,i} = \sum_{p \in \tilde{\mathcal{P}}(t)} y_p, && \forall t \in \tilde{\mathcal{T}}, \quad (44)\\
& w_{t,i-1} \ge w_{t,i}, && \forall t \in \tilde{\mathcal{T}},\ i = 2, \ldots, D, \quad (45)\\
& y_p \in \{0,1\}, && \forall p \in \tilde{\mathcal{P}}, \quad (46)\\
& w_{t,i} \in \{0,1\}, && \forall t \in \tilde{\mathcal{T}},\ i = 1, \ldots, D. \quad (47)
\end{aligned}$$

Constraint (41) ensures that the total cost of the selected pairings does not exceed the cost of the initial solution by more than a specified margin r. Flight coverage constraints (42) ensure that each flight $f \in \mathcal{F}$ is covered exactly once by the selected pairings y_p (again, implemented as soft constraints). Constraints (43) enforce the secondary requirements. Pattern linkage constraints (44) enforces the relationship between the pattern repetitions $w_{t,i}$ and the total pairings y_p associated with pattern $t \in \tilde{\mathcal{T}}$. Ordering constraint (45) maintains sequential consistency for the repetitions $w_{t,i}$. If repetition i of a pattern is selected ($w_{t,i} = 1$), then repetition $i-1$ must also be selected ($w_{t,i-1} = 1$). Finally, constraints (46) and (47) ensure that y and w decision variables are binary.

4.2.3 Advantages and drawbacks

The integer programming method is designed to enhance the regularity of existing solutions while adhering to strict cost constraints. Unlike other methods, it directly refines a feasible solution by adjusting the given set of pairings without exceeding the initial cost by more than a fixed amount. Due to its limited size, it is able to quickly improve the regularity of an initial solution by selecting desirable pairings and their copies.

By simultaneously reoptimizing the whole schedule, the integer programming approach avoids the main drawback of the bonus-based approach. In cases where two different patterns are "competing" for flights, the approach is able to select one and discard the other, in a way that enhances the overall regularity of the solution.

However, this method relies on the quality of the initial patterns, as it does not generate new ones. The model also introduces numerous linking constraints, which can increase computational complexity, especially in large-scale instances. This can be mitigated by providing the optimizer with the initial solution and by using a heuristic stopping criterion.

4.3 Hybrid Bonus-Based and integer programming approach

The *Bonus-Based Improvement* method encourages the repetition of patterns through a single optimization step. However, the solution obtained from this method can still be further improved in terms of regularity. To address this, we propose an iterative version of this method, referred to as the *Iterative Bonus-Based Improvement Approach*. In this extension, several iterations of the bonus-based method are performed. Bonuses $F(z_t^*)$ are recomputed based on the solution obtained in the previous iteration, and the updated model (18)–(21) is solved again. By repeating this process iteratively, regularity can be improved while keeping costs close to that of the input solution. In practice, convergence is typically achieved within 3–4 iterations, leading to a local optimum. Pseudo-algorithm 1 provides a structured overview of this iterative extension:

Algorithm 1 Iterative Bonus-Based Improvement Approach

```
1: Input: $z^* :=$ CPP solution.
2: while true do
3:     Calculate bonus $F(z_t^*, t)$ for each pattern $t$ in the previous solution.
4:     $z^* =$ Solve the problem (18)–(21) with the pre-calculated bonuses.
5:     if the solution shows no further improvement or meets the desired criteria then
6:         break
7:     end if
8: end while
9: Output: Return current solution.
```

Both the *Bonus-Based Improvement* Approach and the *Integer Programming Optimizer* have unique strengths and limitations. The *Bonus-Based Improvement* can generate new patterns during re-optimization and improving regularity through bonus functions, but tends to converge to the first local optimum and requires significant execution time. Conversely, the *Integer Programming Optimizer* is faster, and avoids stagnation at the first local optimum, but cannot generate new patterns. To overcome these challenges, we propose a *Hybrid Method*, which sequentially integrates both approaches (*MIP–Bonus*).

Starting from an initial solution, the method first applies the *Integer Programming Optimizer* to refine and re-optimize the solution while maintaining cost efficiency. Then, the *Bonus-Based Improvement* approach is used to encourage the generation of new patterns and construct an improved solution. By combining both methods in a structured sequence, the hybrid approach leverages the strengths of each technique, effectively balancing regularity enhancement and computational efficiency. The main steps of this approach are outlined in Algorithm 2.

Algorithm 2 Iterative Combined Bonus-Based and Integer Programming Approach

```
1: Input: Initial feasible solution from the pairing optimizer.
2: Step 1: Refine the input solution using the MIP model.
3: while stopping criterion not met do
4:     Step 2: Apply the bonus-based approach with predefined bonus functions.
5:     Step 3: Re-optimize the updated solution using the MIP model.
6: end while
7: Output: Final solution with improved regularity and controlled cost.
```

This combined approach leverages the complementary strengths of the two methods. By alternating between the two methodologies, the combined approach avoids stagnation in the first local optima and ensures that new patterns are generated where necessary. This paper proposes two variants of the bonus approach and two variants of the *MIP* approach. These variants can be combined in several ways within the iterative combined framework. We present a few possibilities in Section 5.4.2

5 Computational results

This section presents the results of different computational experiments conducted to evaluate the proposed solution approaches. The CPP is solved using a commercial airline planning software (developed by our industrial partner), which implements a branch-and-price algorithm. The software operates heuristically to find good-quality solutions in a relatively short time. This software was used to obtain initial solutions and to implement the bonus-based improvement method. The integer programming models were solved using the CPLEX solver.

All experiments were performed on a Linux computer running AlmaLinux 9.6 (Sage Margay), equipped with an Intel(R) Core(TM) i9-14900K processor featuring 24 cores and 32 threads. The system had 62 GiB of RAM, and computations were configured to optimize performance: the solver utilized up to 4 threads to parallelize its computations, the Integer programming solver used up to 20 threads to accelerate solution times, and all other computations were performed using a single core and a single thread.

In the following subsections, we first describe the *test instances* (Section 5.1) used in the experiments, including the *Input* and the *airline solution AS* solutions. We then present the results for each optimization approach: Section 5.2: the *Bonus-Based Improvement* approach, Section 5.3: the *Integer Programming Optimizer*, and Section 5.4: the proposed *Hybrid Methods*

5.1 Test instances

For the computational experiments, we used data derived from one-month flight schedules operated by a major Asian airline. Two datasets were considered. The first, corresponding to April 2019, includes 12,256 flights operated by four aircraft types on short and medium-haul routes. The second, for April 2025, comprises 11,642 flights covering a similar operational scope. These real-world datasets provide a comprehensive basis for evaluating the performance of the proposed methods in optimizing both regularity and cost under realistic conditions. Note that one dataset is pre–COVID-19 while the other is post–COVID-19, resulting in markedly different operational contexts.

The total cost of a CPP solution can be separated into two components: the *hard costs* and the *soft costs*. While hard costs represent direct operational expenses such as hotel costs and per diems, soft costs account for penalties related to operational considerations, including aircraft changes and short connections. Additionally, certain global constraints impose extra costs to ensure feasible rosters, such as productivity constraints.

To assess the effectiveness of each approach, we compare the improved solutions against two benchmarks: the initial feasible solution *Input* provided by our industrial partner's industrial software, and the Airline Solution *AS*, a manually created solution developed over multiple iterations by the airline. We consider the Input solution as the main benchmark since it is the only one obtained through an algorithmic process. The AS solution is used to estimate how close our approaches come to an "ideal" solution. However, it is important to note that AS includes certain pairings that are manually imposed and not always feasible under the rules enforced in the optimization models.

For each dataset, we consider a set of 50 initial solutions generated by the commercial pairing optimizer as our *Input*. To obtain these solutions, randomness was introduced in the solver, which was run several times. This was sufficient to generate solutions that significantly differ across runs.

To illustrate the variability of the input solutions, we plot them across key metrics, as shown in Figure 4 for the 2019 dataset. The number of pairing patterns ranges from 383 to 472 (mean: 423.4), and the regularity ratio varies from 44.18% to 50.70% (mean: 47.57%) with a standard deviation of 1.69, corresponding to a coefficient of variation of 3.56%. The number of pairings also shows a relative standard deviation of 3.9%. This variability highlights the optimizer's stochastic behavior and supports using these solutions as a consistent benchmark for regularity enhancement techniques.

Regarding costs, the average *hard cost* is 18,773, the *soft cost* averages 9,216, and the *total cost* averages 27,988. While the hard cost remains nearly constant, the soft and total costs show moderate variability over a range of 484.

Further analysis of the relationship between the number of pairing patterns and the regularity ratio in the 2019 dataset reveals a clear inverse trend, illustrated in Figure 5: solutions with fewer pairing patterns tend to exhibit higher regularity.

Of particular interest, we note that a single solution dominates the rest in terms of both regularity measures. This further indicates that maximizing the regularity ratio is compatible with minimizing the number of pairing patterns. However, the correlation between the two metrics is not perfect, illustrating that they are not equivalent.

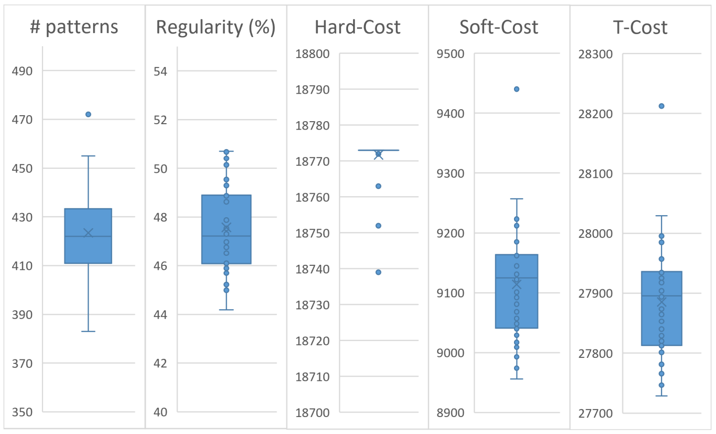

Figure 4: Distribution of 50 solutions generated by the initial pairing optimizer across key indicators (2019 Dataset)

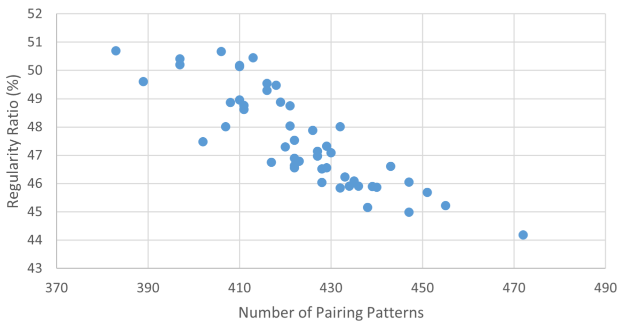

Figure 5: Dispersion of regularity ratio vs. number of pairing patterns across 50 initial solutions (2019 Dataset)

We evaluate each approach using four key metrics. The first two are the *Regularity Ratio* and the number of *Pairing Patterns* (as defined in Section 3.2), which quantify the regularity of the solution. For these metrics, we report the *minimum*, *average*, and *maximum* values across 50 runs, each starting from a different input solution. The other two metrics are *Cost* (broken down into *hard*, *soft*, and *total* components) and *Execution Time*, measured in seconds. For both, only the average values are reported to assess economic and computational performance. These indicators provide a comprehensive basis for evaluating the trade-offs of each method.

In each results table, the row labeled *Input* summarizes the 50 baseline solutions and their key characteristics. The *AS* row presents a highly regular expert-designed solution. The 2019 dataset includes 162 pairing patterns, a regularity ratio of 88.80%, and cost values: 18,773 (hard), 9,216 (soft), and 27,988 (total). For the 2025 dataset, the *AS* solution contains 200 pairings, a regularity ratio of 81.16%, and costs of 24,384 (hard), 136,419 (soft), and 145,635 (total). Each remaining row in the tables represents a step of the algorithm, showing the outcome after applying a specific iteration.

5.2 Bonus-Based improvement

This subsection presents the results of the *Bonus-Based* approach, using two bonus functions: the *Exponential Bonus Function (Expo)* and the *Ratio-Based Bonus Function (Ratio)*. The parameters α and β (for *Expo*) and α (for *Ratio*) are tuned to maintain productivity and avoid overly aggressive regularity adjustments. This tuning has to be performed independently for each dataset because of the scale difference of the cost functions involved. For instance, while the soft costs are around 9 000 for the 2019 dataset, they are at around 145 000 for the 2025 dataset. This is a result of the COVID-19 pandemic, which changed the operational context between the two datasets. It is thus necessary to scale the bonus functions for each dataset. Bonus function parameters used in our tests are presented in Table 1.

Table 1: Parameters used for each method and each dataset

Method	2019 dataset	2025 dataset
Expo	$\alpha = 2\,;\beta = 2$	$\alpha = 4\,;\beta = 2$
Ratio	$\alpha = 20$	$\alpha = 40$

These gains are achieved with no hard cost impact, while the soft cost and total cost even show slight reductions. Execution times are relatively low compared to the time required to generate the initial *Input* solution. This is mainly due to the elimination of overhead tasks (such as duty generation) that can be skipped in subsequent iterations, making the optimization process approximately three times faster. This efficiency makes the method both practical and well-suited for operational use.

Tables 2 and 3 summarize the results for the Bonus-Based Improvement methods on the 2019 and 2025 datasets, respectively. For each regularity metric (number of pairing patterns and regularity ratio), we report the minimum, maximum, and average value across 50 runs (each with a different input solution). We also report the average costs (hard, soft, and total) and the average CPU time of each method. The same values for the input solutions and the airline solution (AS) are reported for convenience. For AS, computing times are not reported since the solutions were created by hand.

Table 2: Results for Bonus based (Ratio and Expo) Bonus Approaches (2019 Dataset)

	Pairing Pattern			Regularity (%)			Cost			Time
	min	avg	max	min	avg	max	hard	soft	total	(s)
Input	383	423.4	472	44.18	47.57	50.70	18771.7	9114.1	27885.2	1181.4
Expo	211	226.9	246	75.60	79.06	83.59	18773.0	8875.5	27647.9	297.7
Ratio	204	220.6	234	76.73	81.32	85.04	18773.0	8905.9	27678.2	272.4
AS	–	–	162	–	–	88.80	18773.0	9216.0	27988.0	

Table 3: Results for Bonus based (Ratio and Expo) Bonus Approaches (2025 Dataset)

	Pairing Pattern			Regularity (%)			Cost			Time
	min	avg	max	min	avg	max	hard	soft	total	(s)
Input	353	396.1	415	54.02	55.73	61.90	24339.0	119177.4	143515.5	1010.4
Expo	236	253,5	272	76,31	79,89	83,18	24337,8	119726,9	144064,0	395.3
Ratio	220	232.84	248	80.76	85.05	87.73	24335.8	120162.4	144497.7	258.2
AS	–	–	200	–	–	81.16	24384.0	121251.0	145635.0	–

For both datasets, the two bonus strategies (*Expo* and *Ratio*) yield substantial improvements over the initial *Input* solutions in terms of regularity and number of pairings, with similar or even slightly reduced total costs.

For the 2019 dataset, the *Expo* approach improves the average regularity from 47.57% to 79.06% (a relative gain of 66.2%), while reducing the number of pairing patterns from 423.4 to 226.9 (a 46.4% reduction). The *Ratio* method achieves even higher regularity (81.32%, or a 70.9% improvement) with 220.6 pairings (-47.9%). Both variants achieve these gains without increasing the hard cost, and the total cost is even slightly reduced.

Similarly, for the 2025 dataset, both bonus strategies remain effective. The *Expo* method improves regularity from 55.73% to 79.89% (a 43.3% improvement), while reducing the average number of pairing patterns from 396.1 to 253.5 (-36%). The *Ratio* variant increases regularity further to 85.05% (a 52.6% gain), with 232.8 pairings on average (-41.2%). These improvements come with minimal changes in total cost.

Finally, execution times are significantly lower than those required to generate the original input solutions. This efficiency is mainly due to the reuse of previously generated duties, which eliminates overhead tasks and makes the bonus-based optimization much faster—often up to three times faster than a fresh run.

Although both methods significantly increase the regularity of the input solution, they still fall short of the AS solution in terms of pairing pattern. However, in terms of regularity ratio, both methods are very close, and sometimes exceed AS. In particular, the Ratio method achieves an average of 87.73% regularity compared to 81.16% in AS. This is because the M_{pattern} highly penalizes solutions that have several rarely used patterns. Such solutions are more likely to be produced by optimization software to cover problematic flights. In opposition, a human planner is more likely to reuse patterns for convenience's sake, even if it means a more costly solution.

5.3 Integer programming Optimizer

This subsection presents the results for the two variants of the *Integer Programming Optimizer* with different optimization objectives: minimizing the number of pairing patterns (*MIPp*) and maximizing regularity (*MIPr*). In both casestionResults for In, the model is not solved to optimality; instead, a CPU time limit of 5000 seconds is imposed per run, as further improvements beyond this limit were observed to be negligible in practice. Note that CPU time refers to the cumulative time used by all threads across all cores, whereas the actual runtime (i.e., wall-clock time) can be much shorter due to parallel processing.

A maximum total cost increase of $r = 1\%$ over the input solution is allowed when necessary, ensuring that improvements in regularity or pairing efficiency do not compromise operational feasibility. Results for 50 runs of each model are summarized in Tables 4 and 5 for the 2019 and 2025 datasets, respectively.

Both approaches yield substantial improvements over the *Input* solutions in terms of regularity, while preserving cost feasibility and demonstrating short runtimes. As expected, *MIPp* produces solutions with a lower number of patterns, whereas *MIPr* produces solutions with a higher regularity ratio.

For the *MIPp* model, which minimizes the number of pairing patterns, the 2019 dataset shows a 55% reduction in the average number of pairings (from 423.4 to 192.9), along with a 67% improvement in regularity (from 47.57% to 79.47%). The total cost remains nearly unchanged (99.79% of the input), and the average runtime is relatively low (115.3 seconds). On the 2025 dataset, *MIPp* reduces the average number of pairings from 396.1 to 201.5 (-49.2%) and increases regularity to 82.61% (+48.3%), with minimal cost increase and an average runtime of 133.4 seconds.

The *MIPr* model, which directly maximizes the regularity ratio, achieves even stronger results on both datasets. In the 2019 case, the regularity ratio rises to 85.07% (+79%) with, on average, 204.9 pairing patterns (-51.6%), and a runtime of 292.2 seconds. In the 2025 case, the regularity increases from 55.73% to 87.31% (+56.7%), with a reduced number of pairings (213.8, -46.0%). The longer runtime (303.1 seconds) is due to the higher complexity of the regularity-based formulation.

Across both datasets, the hard costs remain virtually identical to the input solution, and the total cost remains tightly controlled. These results confirm that *MIPp* is highly effective at reducing the number of pairings quickly, while *MIPr* excels in significantly enhancing regularity, even under tight cost constraints.

Table 4: Results for MIPp and MIPr Approaches (2019 Dataset)

	Pairing Pattern			**Regularity (%)**			**Cost**			**Time**
	min	avg	max	min	avg	max	hard	soft	total	(s)
Input	383	423.4	472	44.18	47.57	50.70	18771.7	9114.1	27885.2	1181.4
MIPp	181	192.9	209	73.12	79.47	84.70	18773.0	9053.4	27825.9	115.3
MIPr	189	204.9	234	79.39	85.07	87.31	18773.0	9105.4	27878.0	292.2
AS	–	–	162	–	–	88.80	18773.0	9216.0	27988.0	–

Table 5: Results for MIPp and MIPr Approaches (2025 Dataset)

	Pairing Pattern			**Regularity (%)**			**Cost**			**Time**
	min	avg	max	min	avg	max	hard	soft	total	(s)
Input	353	396,1	415	54,02	55,73	61,90	24339,0	119177,4	143515,5	1010.4
MIPp	192	201,5	213	75,35	82,61	86,55	24550,6	120212,1	144762,6	133,4
MIPr	201	213,8	230	83,82	87,31	89,41	24355,7	120562,9	144918,0	303,1
AS	–	–	200	–	–	81.16	24384.0	121251.0	145635.0	–

5.4 Iterative approaches

In this subsection, we present results for iterative optimization approaches. Two strategies are evaluated: (1) the *Iterative Bonus-Based Improvement*, which applies the bonus-based method iteratively; and (2) The *Hybrid Bonus-Based and Integer Programming Approach*, which combines the bonus-based and integer programming methods.

5.4.1 Iterative Bonus-Based improvement

The iterative bonus-based method was tested with both the exponential bonus function (Equation 22) and the bonus function based on the regularity ratio (Equation 23). In each setting, we perform three iterations of the bonus-based method, each output serving as the input for the next iteration.

Tables 6 and 7 present results over three iterations of the *Iterative Bonus-Based Improvement* method using both exponential and ratio-based bonus functions. For both datasets, we observe clear gains in regularity and pairing pattern reduction, with diminishing returns across iterations. Indeed, while a significant improvement in both regularity metrics is observed after the second iteration, the third iteration only manages to marginally improve regularity.

Importantly, these gains are achieved with minimal or no increase in cost. In the 2019 case, soft and total costs are slightly reduced compared to the *Input* solution. In 2025, the total cost increases remain within acceptable operational margins: +0.57% for *Expo* and +0.86% for *Ratio*.

It is worth noting that while the regularity ratio approaches that of the expert-designed *AS* solution (88.80% for 2019 and 81.16% for 2025), the number of pairing patterns remains higher (around 195 and 220 vs. 162 and 200 in *AS*, for 2019 and 2025, respectively). This confirms that regularity cannot be assessed solely by counting patterns, as high regularity can result from frequently repeated key patterns even when the total number is higher.

Table 6: Results for iterative Bonus based (Ratio and Expo) Bonus Approaches (2019 Dataset)

	Pairing Pattern			Regularity (%)			Cost			Time
	min	avg	max	min	avg	max	hard	soft	total	(s)
Input	383	423.4	472	44.18	47.57	50.70	18771.7	9114.1	27885.2	1181.4
Itr1: Expo	211	226.9	246	75.60	79.06	83.59	18773.0	8875.5	27647.9	297.5
Itr2: Expo	190	199.5	212	81.87	85.36	87.07	18773.0	8944.6	27716.8	336.7
Itr3: Expo	190	195.5	203	82.49	86.19	88.09	18773.0	8944.5	27716.7	274.3
Itr1: Ratio	204	220.6	234	76.73	81.32	85.04	18773.0	8905.9	27678.2	272.4
Itr2: Ratio	186	198.8	211	83.54	86.09	87.53	18773.0	8984.9	27757.3	271.2
Itr3: Ratio	186	195.7	205	83.82	86.56	87.74	18773.0	8984.1	27756.4	245.3
AS	–	–	162	–	–	88.80	18773.0	9216.0	27988.0	

Table 7: Results for iterative Bonus based (Ratio and Expo) Bonus Approaches (2025 Dataset)

	Pairing Pattern			Regularity (%)			Cost			Time
	min	avg	max	min	avg	max	hard	soft	total	(s)
Input	353	396.1	415	54.02	55.73	61.90	24339.0	119177.4	143515.5	1010.4
Itr1: Expo	236	253,5	272	76,31	79,89	83,18	24337,8	119726,9	144064,0	395.4
Itr2: Expo	214	224,2	236	80,05	84,71	86,33	24335,6	119967,4	144302,6	407.6
Itr3: Expo	211	218,3	230	83,43	85,81	87,14	24335,1	119992,5	144327,4	252.2
Itr1: Ratio	220	232.8	248	80.76	85.05	87.73	24335.8	120162.4	144497.7	358.2
Itr2: Ratio	208	215.6	228	87.43	88.79	90.08	24332.6	120386.3	144718.3	230.5
Itr3: Ratio	207	213.0	221	87.74	89.29	90.21	24332.6	120417.2	144749.3	218.8
AS	–	–	200	–	–	81.16	24384.0	121251.0	145635.0	–

5.4.2 Hybrid Bonus-Based and Integer programming approach

This subsection presents the results of hybrid optimization strategies that combine the Bonus-Based Improvement approach with the Integer Programming Optimizer. Given that there are several possible settings for each approach, several combinations can be envisioned. We present two such combinations. In the first (*MIPp-Ratio-MIPp*), the input solution is first solved using *MIPp*. The produced solution is then reoptimized using the *Ratio*, method, which is in turn reoptimized by *MIPp* . The second method (*MIPr-Ratio-MIPr*) is similar, except that each *MIPp* is replaced by a *MIPr*.

To ensure that total costs remain within the authorized increase threshold of $(1 + r)\%$ over the input solution, we allocate specific cost margins to each *MIP* optimization phase. In our experiments, the first *MIP* is allowed to reach up to 0.5% over the input solution cost, while the second *MIP* is permitted a total increase of up to 1% over the input solution cost. This allocation strategy guarantees that the full hybrid approach respects the global cost constraint. More generally, for a hybrid process involving m *MIP* steps, the i-th *MIP* iteration can be allowed an increase of ir/m over the input solution. This setup allows for controlled cost escalation while still enabling iterative improvements in regularity and pattern structure. Note that the bonus method does not control the cost increase, so it is possible that an *MIP* has no feasible solution within the cost limit. This is why we implement the cost constraint as a soft constraint.

Tables 8 and 9 summarize the results for both the 2019 and 2025 datasets, each over three iterations. The *better* columns indicate the number of solutions (out of 50) that outperform the expert-designed *AS* solution in terms of each regularity metric.

For the 2019 dataset, the *MIPp-Ratio-MIPp* strategy reduces the average number of pairing patterns to 172.6 (a 59.2% reduction) and increases regularity to 86.65% (an 82.2% improvement), all within 0.37% of the baseline total cost. The method also produces solutions with as few as 157 pairings, even outperforming the *AS* solution in several instances. The *MIPr-Ratio-MIPr* variant achieves the highest regularity observed, reaching 89.19% on average (an 87.5% improvement) with an average of 179.9 pairing patterns (a 57.5% reduction). It exceeds the *AS* regularity ratio in 34 out of 50 test cases. In both approaches, the hard cost remains unchanged, and total cost increases remain under 0.21%.

For the 2025 dataset, similar trends are observed. The *MIPp-Ratio-MIPp* approach reduces the number of pairing patterns to an average of 186.1 (a 53% reduction). It also increases the average regularity ratio to 89.60%, well above the *AS* value of 81.16%. It outperforms the *AS* solution in all 50 cases in terms of number of patterns and in 49 out of 50 cases in terms of regularity ratio. The *MIPr-Ratio-MIPr* variant pushes regularity even further, reaching an average ratio of 91.06% (a 63.4% improvement over the baseline), with 194.8 pairing patterns on average. It outperforms the *AS* solution in all 50 cases in terms of number of patterns and in 46 out of 50 cases in terms of regularity ratio. Its average total cost remains comparable to that of the *MIPp-Ratio-MIPp* approach.

Remarkably, for both approaches, these improvements are achieved within the authorized cost limit relative to the input solution, while the total cost remains approximately 0.5% lower than that of the *AS* solution. This demonstrates that the proposed methods can produce higher-quality solutions at a lower cost. In other words, for the same cost level as *AS*, the method could potentially further enhance both regularity metrics.

Table 8: Hybrid Approaches Results (2019 Dataset)

	Pairing Pattern				Regularity (%)				Cost			Time
	min	avg	max	better	min	avg	max	better	hard	soft	total	(s)
Input	383	423.4	472	–	44.18	47.57	50.70	–	18771.7	9114.1	27885.2	1181.4
Itr1: MIPp	181	192.9	209	0	73.12	79.43	84.70	0	18773.0	9053.4	27825.9	144.6
Itr2: Ratio	165	182.1	201	0	82.91	87.26	89.82	3	18773.0	9073.3	27845.8	266.8
Itr3: MIPp	157	172.6	184	3	80.97	86.65	89.73	5	18773.0	9214.6	27987.1	8.8
Itr1: MIPr	186	202,8	229	0	81,29	85,39	87,23	0	18773,0	9027,4	27799,9	143.7
Itr2: Ratio	172	183,2	196	0	87,17	88,77	90,18	23	18773,0	9033,4	27805,9	257.1
Itr3: MIPr	171	179,9	190	0	87,93	89,19	90,40	41	18772,2	9127,4	27899,2	8.7
AS	–	–	162	–	–	–	88.80		18773.0	9216.0	27988.0	

Table 9: Hybrid Approaches Results (2025 Dataset)

	Pairing Pattern				Regularity (%)				Cost			Time
	min	avg	max	better	min	avg	max	better	hard	soft	total	(s)
Input	353	396.1	415	–	54.02	55.73	61.90	–	24339.0	119177.4	143515.5	1010.4
Itr1: MIPp	195	205.0	216	8	74.08	81.87	85.84	37	24508.0	119730.0	144237.9	134.7
Itr2: Ratio	185	195.6	209	38	82.10	89.78	92.46	50	24429.5	120158.6	144587.9	285.4
Itr3: MIPp	176	186.1	198	50	80.45	89.57	92.47	49	24559.0	120213.9	144772.5	7.8
Itr1: MIPr	208	223.5	243	0	81.72	84.95	87.34	50	24358.0	119939.0	144296.3	149.4
Itr2: Ratio	189	199.7	209	25	85.20	90.81	92.06	50	24337.2	120421.2	144758.0	318.5
Itr3: MIPr	187	194.8	207	46	86.16	91.06	92.46	50	24352.1	120491.6	144843.3	8.9
AS	–	–	200	–	–	–	81.16	–	24384.0	121251.0	145635.0	–

6 Conclusion

This study addressed the Crew Pairing Problem (CPP) by incorporating *regularity* as a key optimization objective alongside cost. Regularity (defined as the consistent repetition of pairing patterns) enhances operational robustness, simplifies crew planning, and makes it an essential criterion for modern airline scheduling.

To achieve this, we proposed and tested three families of methods: (1) bonus-based iterative refinements, (2) integer programming (MIP) optimizers, and (3) hybrid approaches that combine the strengths of both. Each strategy was evaluated on 50 baseline solutions provided by a commercial pairing optimizer, which displayed notable variability in regularity and pattern counts. Bonus-based approaches, particularly when using ratio-based functions, achieved up to 47% reduction in pairing patterns and a relative improvement of over 71% in regularity ratio. *MIP* optimizers, though limited to refining existing solutions, reached good improvements in a single iteration, which demonstrates their efficiency. The *MIPp* objective reduced the number of patterns by 55%, while the *MIPr* model increased regularity by over 79%.

The most notable results were achieved by the *hybrid approaches*. The best-performing configuration, *MIPr* combined with ratio-based bonuses, reduced pairing patterns by 57%, and improved the regularity ratio up to 88%. Importantly, these improvements were achieved with total cost increases of less than the authorized augmentation 1%, and without any rise in hard costs. Overall, our findings highlight the value of tailoring optimization methods to specific goals. If the priority is to reduce the number of pairing patterns, *MIPp*-based strategies are most effective. If regularity ratio is paramount, *MIPr* paired with bonus functions is recommended.

These results provide a strong foundation for implementing robust and regular crew pairings in real-world airline operations. Future extensions could include more precise regularity metrics, integration with the crew rostering problem, and personalized crew preferences towards regularity.

References

Amberg, Boris, Amberg, Bastian, and Kliewer, Natalia (2011). "Approaches for increasing the similarity of resource schedules in public transport". In: *Procedia-Social and Behavioral Sciences* 20, pp. 836–845.

Barnhart, C., Cohn, A. M., et al. (2003). "Airline Crew Scheduling". In: *Handbook of Transportation Science, International Series in Operations Research & Management Science.* Vol. 56. Springer, US, pp. 517–560.

Barnhart, C., Johnson, E. L., Anbil, R., et al. (1994). "A column-generation technique for the long-haul crew-assignment problem". In: *Optimization in Industry 2.* John Wiley & Sons, Inc., pp. 7–24.

Barnhart, C., Johnson, E. L., Nemhauser, G. L., et al. (1998). "Branch-and-Price: Column Generation for Solving Huge Integer Programs". In: *Operations Research* 46.3, pp. 316–329.

Dantzig, G. B. and Wolfe, P. (1960). "Decomposition Principle for Linear Programs". In: *Operations Research* 8.1, pp. 101–111.

Desaulniers, G., Desrosiers, J., Dumas, Y., et al. (1997). "Crew Pairing at Air France". In: *European Journal of Operational Research* 97.2, pp. 245–259.

Desaulniers, G., Desrosiers, J., Ioachim, I., et al. (1998). "A Unified Framework for Deterministic Time Constrained Vehicle Routing and Crew Scheduling Problems". In: *Fleet Management and Logistics*, pp. 57–93.

Desaulniers, G., Desrosiers, J., and Solomon, M. (2005a). *Column Generation.* New York, NY: Springer.

— (2005b). *Large-Scale Models in the Airline Industry.* Springer, US.

Desaulniers, G., Lessard, F., et al. (2020a). "Dynamic Constraint Aggregation for Solving Very Large-Scale Airline Crew Pairing Problems". In: *SN Operations Research Forum* 1.3, p. 19.

Desaulniers, Guy, Lessard, François, et al. (2020b). "Dynamic constraint aggregation for solving very large-scale airline crew pairing problems". In: *SN Operations Research Forum.* Vol. 1. Springer, pp. 1–23.

Desrosiers, J., Dumas, Y., Desrochers, M., et al. (1991). *A breakthrough in airline crew scheduling.* Technical Report. Springer.

Desrosiers, J., Dumas, Y., Solomon, M. M., et al. (Oct. 1995). "Time constrained routing and scheduling". In: *Handbooks in Operations Research and Management Science, Vol. 8 : Network Routing.* Ed. by M. Ball et al. Amsterdam: Elsevier. Chap. 2, pp. 35–139.

Dillon, J. E. and Kontogiorgis, Spyros (1999). "US Airways optimizes the scheduling of reserve flight crews". In: *Interfaces* 29.5, pp. 123–131.

Elhallaoui, I. et al. (2005). "Dynamic Aggregation of Set-Partitioning Constraints in Column Generation". In: *Operations Research* 53.4, pp. 632–645.

Gilmore, P. C. and Gomory, R. E. (1961). "A linear programming approach to the cutting-stock problem". In: *Operations Research* 9.6, pp. 849–859.

Gopalakrishnan, Balaji and Johnson, Ellis L (2005). "Airline crew scheduling: State-of-the-art". In: *Annals of Operations Research* 140, pp. 305–337.

Hizir, Ahmet Esat (2024). "Large-Scale Airline Recovery using Mixed-Integer Optimization and Supervised Learning". PhD thesis. Massachusetts Institute of Technology.

Hu, J. and Johnson, E. L. (1999). "Computational Results with a Primal–dual Subproblem Simplex Method". In: *Operations Research Letters* 25.4, pp. 149–157.

Irnich, S. and Desaulniers, G. (2005). "Shortest path problems with resource constraints". In: *Column Generation*. Ed. by G. Desaulniers, J. Desrosiers, and M. M. Solomon. Boston, MA: Springer US. Chap. 2, pp. 33–65.

Kasirzadeh, A. (2015). *Optimisation intégrée des rotations et des blocs mensuels personnalisés des équipages en transport aérien*. Ecole Polytechnique, Montreal (Canada).

Klabjan, D. et al. (2001a). "Airline crew scheduling with regularity". In: *Transportation science* 35.4, pp. 359–374.

Klabjan, D. et al. (2001b). "Solving Large Airline Crew Scheduling Problems: Random Pairing Generation and Strong Branching". In: *Computational Optimization and Applications* 20.1, pp. 73–91.

Kornilakis, Harry and Stamatopoulos, Panagiotis (2002). "Crew pairing optimization with genetic algorithms". In: *Hellenic conference on artificial intelligence*. Springer, pp. 109–120.

Lavoie, S., Minoux, M., and Odier, E. (1988). "A New Approach for Crew Pairing Problems by Column Generation with an Application to Air Transportation". In: *European Journal of Operational Research* 35.1, pp. 45–58.

Mohamed, NF et al. (2020). "Comparison of two hybrid algorithms on incorporated aircraft routing and crew pairing problems". In: *Indonesian Journal of Electrical Engineering and Computer Science* 18.3, pp. 1665–1672.

Muter, I. et al. (2013). "Solving a Robust Airline Crew Pairing Problem with Column Generation". In: *Computers & Operations Research* 40.3, pp. 815–830.

Parmentier, Axel and Meunier, Frédéric (2020). "Aircraft routing and crew pairing: Updated algorithms at Air France". In: *Omega* 93, p. 102073.

Quesnel, F. (2019). "Trois Variantes Du Problème De Rotations Pour Une Approche Semi-intégrée De La Planification D'horaires De Personnel Aérien". PhD thesis. Ecole Polytechnique, Montreal (Canada).

Quesnel, F., Desaulniers, G., and Soumis, F. (2017). "A new heuristic branching scheme for the crew pairing problem with base constraints". In: *Computers & Operations Research* 80, pp. 159–172.

— (2020). "Improving air crew rostering by considering crew preferences in the crew pairing problem". In: *Transportation Science* 54.1, pp. 97–114.

Saddoune, M (2010). "Optimisation simultanée des rotations et des blocs mensuels des équipages aériens". PhD thesis. École Polytechnique de Montréal.

Saddoune, M., Desaulniers, G., and Soumis, F. (July 2009). "A Rolling Horizon Solution Approach for the Airline Crew Pairing Problem". In: *Proceedings of the 2009 International Conference on Computers & Industrial Engineering*. Troyes, France, pp. 344–347.

— (Mar. 2013). "Aircrew Pairings with Possible Repetitions of the Same Flight Number". In: *Computers and Operations Research* 40.3, pp. 805–814.

Sandhu, R. and Klabjan, D. (2007). "Integrated Airline Fleeting and Crew-Pairing Decisions". In: *Operations Research* 55.3, pp. 439–456.

Steinzen, Ingmar, Suhl, Leena, and Kliewer, Natalia (2009). "Branching strategies to improve regularity of crew schedules in ex-urban public transit". In: *OR spectrum* 31.4, pp. 727–743.

Tahir, Adil et al. (2021). "An improved integral column generation algorithm using machine learning for aircrew pairing". In: *Transportation Science* 55.6, pp. 1411–1429.

Tajima, Akira and Misono, Shinji (1997). "Airline crew-scheduling problem with many irregular flights". In: *International Symposium on Algorithms and Computation*. Springer, pp. 2–11.

Wu, Wei et al. (2016). "A column generation approach to the airline crew pairing problem to minimize the total person-days". In: *Journal of Advanced Mechanical Design, Systems, and Manufacturing* 10.3, JAMDSM0040–JAMDSM0040.

Xu, Yifan, Wandelt, Sebastian, and Sun, Xiaoqian (2021). “Airline integrated robust scheduling with a variable neighborhood search based heuristic”. In: *Transportation Research Part B: Methodological* 149, pp. 181–203.

Yaakoubi, Yassine, Soumis, François, and Lacoste-Julien, Simon (2020). “Machine learning in airline crew pairing to construct initial clusters for dynamic constraint aggregation”. In: *EURO Journal on Transportation and Logistics* 9.4, p. 100020.

Zeighami, Vahid and Soumis, François (2019). “Combining Benders’ decomposition and column generation for integrated crew pairing and personalized crew assignment problems”. In: *Transportation Science* 53.5, pp. 1479–1499.

Zeren, Bahadır and Özkol, Ibrahim (2016). “A novel column generation strategy for large scale airline crew pairing problems”. In: *Expert Systems with Applications* 55, pp. 133–144.