Journal of Air Transport Management 114 (2024) 102492

Contents lists available at ScienceDirect

Journal of Air Transport Management

journal homepage: www.elsevier.com/locate/jairtraman

An adaptive greedy heuristic for large scale airline crew pairing problems

Bahadır Zeren [a,b], Ender Özcan [a], Muhammet Deveci [c,d,e,*]

[a] School of Computer Science, University of Nottingham, Wollaton Road, Nottingham, NG8 1BB, UK
[b] Turkish Airlines, Directorate of Corporate Development and Information Technologies Department, Istanbul, Turkey
[c] Department of Industrial Engineering, Turkish Naval Academy, National Defence University, 34940 Tuzla, Istanbul, Turkey
[d] The Bartlett School of Sustainable Construction, University College London, London, WC1E 7HB, UK
[e] Department of Electrical and Computer Engineering, Lebanese American University, Byblos, Lebanon

ARTICLE INFO

Keywords:
Airline crew pairing
Set covering
Heuristic
Genetic algorithm

ABSTRACT

A crew pairing represents a sequence of flight legs that constitute a crew work allocation, starting and ending at the same crew base. A complete set of crew pairings covers all flight legs in the timetable of an airline for a given planning horizon. That determines the rosters for the crew and their quality, since those pairings would potentially include layovers, deadheads and connection times which are the key factors which directly contribute to the operational crew costs. Considering that crew costs form the second largest bit in the overall operational cost, generating optimized crew pairings is a vital process for the airline companies. In this study, a score-based adaptive greedy heuristic and a genetic algorithm are presented for solving large scale instances of airline crew pairing problems. Both solution methods are applied to a set of real-world problem instances from Turkish Airlines which is one of the largest carriers in the world. The empirical results show that the proposed approaches are indeed capable of generating high quality solutions for crew pairing, even for the large scale problem instances.

1. Introduction

The airline industry and its planning problems have been the main focus of Operation Researchers (ORs) since 1950s (Kasirzadeh et al., 2017). Many airlines can improve their profitability by employing advanced optimization models & algorithms and computer hardware & software (Klabjan, 2005; Herekoglu and Kabak, 2023) There is a range of optimization problems in the competitive airline industry (Agustin et al., 2017) to deal with, for example, flight scheduling, fleet assignment, aircraft maintenance & routing, and crew scheduling (crew pairing and crew rostering) (Ahmed et al., 2022; Deveci and Demirel, 2018a). Those problems are typically solved sequentially (Wen et al., 2021) and often in a stage-based manner (Dunbar et al., 2014) due to the complex nature of the real-life instances, such as, their scale, number of constraints & variables, non-linear costs and more (Muter et al., 2013; Klabjan et al., 2001b). Crew scheduling problem (CSP) is one of the biggest airline planning problems and also the most sophisticated one. Because of its high level of complexity, it is generally decomposed into two problems: (i) crew pairing problem (CPP), and (ii) crew rostering (or crew assignment) problem (CRP), respectively (Shafipour-Omrani et al., 2021; Ahmed et al., 2022). Since both problems carry similar mathematical structures, they are often tackled by employing similar methods (Desrosiers et al., 2000), including heuristic or mathematical programming techniques (Deng and Lin, 2011).

This study focuses on the airline crew pairing problem which is an NP-hard optimization problem (Azadeh et al., 2013; Aydemir-Karadag et al., 2013; Saemi et al., 2022), inherently complex, and computationally intensive part of airline crew scheduling (ACS) (Zeighami et al., 2020). A crew paring is a sequence of one ore more duties/legs in a time period which starts and ends at a home base. The lengths of pairings for short-haul generally may range from minimum of 1 to maximum 4 of days including overnight rest, but this rule may be flexible according to airline company or medium-haul/long-haul flight. It is actually a sequence of flight legs representing a work pattern for crew assignment (Zeren and Özkol, 2016). The aim of crew pairing optimization is to cover all flight legs and minimizing crew costs and increasing efficiency by generating set of feasible pairings (Cohn and Barnhart, 2003). The quality of crew pairings depend on some important factors, such as, the number of deadheads, number of overnights and total man day as key performance indicators. An effective and efficient solution to CPP is crucial, since the crew costs are the second highest component of direct operating cost after fuel (Deveci and Demirel, 2018a) For a major airline, the salary costs of the crew members can reach hundreds of millions of dollars per year. The main objective of the pairing optimization helps to minimize operational crew costs and maximize crew utilization (Zeren and Özkol, 2016) while respecting various complex

* Corresponding author at: Department of Industrial Engineering, Turkish Naval Academy, National Defence University, 34940 Tuzla, Istanbul, Turkey.
E-mail addresses: bzeren@gmail.com (B. Zeren), ender.ozcan@nottingham.ac.uk (E. Özcan), muhammetdeveci@gmail.com (M. Deveci).

https://doi.org/10.1016/j.jairtraman.2023.102492
Received 30 May 2022; Received in revised form 25 July 2023; Accepted 22 September 2023
Available online 29 September 2023

constraints, such as, airline company policies, union or Federal Aviation Administration (FAA) (Emden-Weinert and Proksch, 1999). Even small improvements in CPP can provide significant financial benefits (million of dollars saving) (Gershkoff, 1989; Graves et al., 1993; Hoffman and Padberg, 1993; Barnhart et al., 1995; Kohl and Karisch, 2004; Deveci and Demirel, 2018a; Wen et al., 2021). Because of all of these reasons including its economic importance, the CPP has received the great interest from the operations research community (Zeighami et al., 2020). It is also one of the most commonly studied problems in both industrial and academic environment.

Airline crew pairing problems are classified as daily, weekly or monthly crew pairing problems. In daily problem, every flight leg is supposed to be flown every day (Haouari et al., 2019) If the each flight is repeated each week, then this problem can be solved as weekly problem. Although the monthly problem leading to large scale instances is the most difficult class to solve, airline companies prefer monthly schedules the most (Deveci and Demirel, 2018b).

Selection hyper-heuristics are a class of general-purpose high-level optimizers managing and mixing a set of low level heuristics whether constructive or perturbative. This study proposes a novel squeaky-wheel (Joslin and Clements, 1999) inspired adaptive greedy heuristic, controlling a set of constructive heuristics, for large-scale crew paring. This approach can be considered as a non-stochastic selection hyper-heuristic as none of its components have any randomness. Its performance is compared to a novel genetic algorithm (GA) used as a hyper-heuristic mixing constructive low level heuristics. Additionally, in this study, we provide a set of new benchmark instances obtained from a major airline company, Turkish airlines using their publically available schedule for future research. We have tested our approaches across those instances. The rest of the paper is organized as follows: Section 2 mentions literature review. Fundamental definitions are introduced in Section 2.2. Section 4 gives some notation and mathematical formulation.

2. Background

2.1. Related work

The crew scheduling problem (CSP) has been studied extensively for many years by the OR community and various techniques ranging from exact methods to heuristics have been proposed. Although significant results are achieved and optimal algorithms do exist for various instances of this problem, nevertheless not to the satisfaction of all airlines (Arabeyre et al., 1969; Baker and Fisher, 1981; Deveci and Demirel, 2018b). CSP plays an important role in the OR literature because of not only its potential economic impact but also its challenging nature as a real-world problem influencing the development and improvement of important optimization solution methodologies including branch-and-price (Quesnel et al., 2020), branch-and-cut (Hoffman and Padberg, 1993), lagrangian relaxation (Beasley and Cao, 1996; Sandhu and Klabjan, 2007), stabilization (du Merle et al., 1999), shortest path algorithms (Lavoie et al., 1988; Hjorring and Hansen, 1999; Makri and Klabjan, 2004; Borndörfer et al., 2006), benders decomposition (Mercier et al., 2005; Zeighami and Soumis, 2019) and heuristics (Salazar-González, 2014; Quesnel et al., 2017; Saemi et al., 2022).

Focusing on crew pairing problem (CPP), mathematical programming based exact methods including branch&cut, branch&bound, column generation based heuristics, branch&price (Levine, 1996; Klabjan et al., 2001a; Barnhart et al., 2003; Tahir et al., 2022); have been proposed as solution methods. However, when the problem size increases to large-scale, it becomes extremely difficult to find a high quality solution in reasonable amount of time using mathematical programming and sometimes even a solution might not be found. Klabjan et al. (2001a) proposed LP relaxation of the set partitioning problem for solving large-scale airline crew scheduling problem. They have generated roughly half a billion pairings and selected nearly 10 million pairings with low reduced cost from within. Branch-and-bound heuristic is used for column selection. Yaakoubi et al. (2020) integrated a machine learning and mathematical programming permits to solve larger crew pairing problems.

Meta-heuristic algorithms have also been applied to CPP. The performance of meta-heuristics are usually dependent on the initial algorithmic parameter settings and cannot guarantee that the solutions obtained are optimal (Doi et al., 2018). Azadeh et al. (2013) presented various meta-heuristics, including ant colony optimization (ACO), genetic algorithm (GA), particle swarm optimization (PSO), and hybrid algorithms based on GA and ACO for solving the crew pairing problems and applied to synthetic small scale flight data. The hybrid PSO outperformed the other approaches across 20 instances. Saemi et al. (2022) presented an integrated mathematical model to solve crew pairing and rostering problems using an ant colony optimization algorithm.

A hyper-heuristic is a search method or learning mechanism that selects or generates low level heuristics to solve computationally hard problems (Drake et al., 2020). The main characteristic of hyper-heuristics is that they search the space of heuristics instead of solutions directly (Burke et al., 2019). The iterative search process carried out at each step by a selection hyper-heuristic can usually be decomposed into two stages: heuristic selection and move acceptance (Drake et al., 2020). Selection hyper-heuristics have been successfully applied to many real-world problems, including timetabling (Burke et al., 2003; Kendall and Hussin, 2004; Burke et al., 2007; Pillay and Banzhaf, 2009), bin-packing (Ross et al., 2002; López-Camacho et al., 2014), vehicle routing (Walker et al., 2012), personnel scheduling (Cowling et al., 2002), and cutting stock (Terashima-Marín et al., 2005). In none of the previous studies, a hyper-heuristic is used for crew pairing and we introduce two novel methods in this study.

2.2. Airline crew pairing terminology

Crew pairings generated during crew pairing optimization phase must be legal and satisfy all the constraints of collective agreements, union's specific work rules, governmental regulations, company policies and FAA, which often differ from one airline to another. Some of the types of constraints can be defined as follows: Minimum connection times must be within a certain range which can be between 30, 60 or 90 min depending on the connection airport. Total duty time must not exceed 12–14 h depending on the starting time of the duty and/or crew composition. Minimum rest time can be between 8–36 h depending on duration of the flight duty. A crew pairing can touch maximum 4 days (Zeren and Özkol, 2016). A sample crew pairing can be seen in Table 1. Some of the basic airline terminology is covered in Table 2.

3. Proposed approach

In the literature and in the industry, mathematical based methods have been heavily studied and used to solve CSPs. Even though there are plenty of papers on solving CSPs using meta-heuristics such as genetic algorithms (GA), particle-swarm optimization (PSO) etc., none of them are tested using the large scale instances of CSP (Deveci and Demirel, 2018b). Hence, this situation offers an opportunity for researchers who work in the field of heuristic optimization, including meta-heuristics, and hyper-heuristics to design and test new approaches. In previous studies, each solution methods is often tested on either synthetic problems or particular problem instances obtained from particular airline companies which are often not accessible. Hence, the lack of public benchmark datasets in this field is an other issue for the researchers. Even for the small and medium size CSP instances, this makes it almost impossible to compare the performance of different heuristic optimization approaches. Contributing to the literature and to further developments by making the datasets used public would be an other contribution of this study to the field. Therefore

Table 1
A sample crew pairing.

Flight duty nr	Duty start	Duty end	Departure time	Arrival time	Dep.	Arr.
1	25.03.15 10:00		25.03.15 11:15	25.03.15 13:20	IST	TLV
			25.03.15 14:20	25.03.15 16:25	TLV	IST
		25.03.15 23:05	25.03.15 17:25	25.03.15 22:35	IST	ALA
Rest period	–	–	–	–	–	–
2	26.03.15 19:45	27.03.15 03:25	26.03.15 20:45	27.03.15 02:55	ALA	IST
Rest period	–	–	–	–	–	–

Table 2
The definitions of some basic concepts of crew scheduling.

Basic concepts	Definitions
Flight/Flight leg	Flight leg is the most basic input for crew scheduling. Each flight leg has the following characteristics: flight no, place of departure, place of arrival, departure time, arrival time, flight (block) time and fleet type.
Duty/Flight Duty	A duty is constituted by one ore more consecutive flight legs and means a working day for a crew (Klabjan and Schwan, 2001).
Pairing	A pairing is composed of flight legs of aircraft rotations, starts and finally ends at the same home base. It is a sequence of duties and rest periods. In short and medium-haul problems, pairings usually take 1–4 days. Longer pairings are allowed in long-haul problems (Kasirzadeh et al., 2017).
Home/crew base	A home base is a city or an airport where all or part of the flight crew of airline resides. Major airline companies usually have more than one home bases. Therefore all the pairings must start and end at the same home base and crew pairings generated for each home base must be proportional to the numbers of the crew who resides on that home base.
Connection time	Time period between two consecutive flight legs.
Rest period	Time period between two flight duties is named rest period. During this period crew is free of any duty and must have rest for the next duty at the hotel they are taken to.
Deadhead	Deadhead crew is passive (not working) members of the flight. They are just transported to an other location like passengers to perform an other duty (Andersson et al., 1998).
Total duty time	Time period that starts with briefing period that takes place just before the first leg of the duty and ends at the end of debriefing period that takes place at the end of last leg of the duty (Demirel and Deveci, 2017).
Briefing period	Time period that is mainly reserved for preparation purposes for the crew just before the first leg of the duty. It is generally applied as 30 or 60 min depending on whether the duty is the first duty of the pairing.
Debriefing period	Time period that is reserved for doing necessary arrangements, data collections etc. just before ending the flight duty.

the main motivation of this study can be summarized as developing effective heuristic optimization approaches for solving large scale crew pairing problems and encourage further developments in the field by making the used datasets public.

Two different heuristic optimization approaches that can be considered as hyper-heuristics were developed and compared in this study:

- A score based ordering heuristic (SBH) which is a novel adaptive greedy heuristic that is inspired by squeaky-wheel optimization method (Joslin and Clements, 1999). It uses difficulty scores calculated for each leg after each pairing generation using values obtained from some sub-heuristics such as *CoverabilityDegree (CD)*, *DeadheadingHeuristicModifier (DHM)* and *ActivityHeuristicModifier (AHM)* which are discussed in detail in Section 4.1. Ordering (priority) of the legs is dynamically changed according to their difficulty scores and crew pairings are generated according to this order by solving a sub optimization problem that was modeled as a resource constrained shortest path problem (RCSP). This approach can also be called an adaptive hyper-heuristic that uses combination of heuristic orderings.
- A novel genetic algorithm (GA) model used as a hyper-heuristic mixing constructive low level heuristics that is used for finding optimum ordering (priority) of the legs and generates crew pairings by solving the same RCSP problem.

3.1. Main problem model

CSP can generally be represented as a set partitioning (SP) (Anbil et al., 1991; Desaulniers et al., 1997; ReVelle and McGarity, 1997; Dawid et al., 2001; Doi et al., 2018) or a set covering (SC) problem (Baker and Fisher, 1981; Aydemir-Karadag et al., 2013; Díaz-Ramírez et al., 2014; Demirel and Deveci, 2017) in the literature. Each column represents a generated pairing and each row represents a different flight leg. If a flight leg was covered more than once, this model should be considered as SC problem. Over-covered flights represents deadhead flights.

The mathematical formulation of this study was formulated as set covering model because it perfectly satisfies all the representation requirements of CPP. The mathematical formulation of SC for CPP can be defined as follows:

$$\min \quad \sum_{p \in P} c_p x_p \tag{1}$$

$$\text{s.t.} \quad \sum_{p:\, f \in p} x_p \geq 1, \quad \forall f \in F \tag{2}$$

$$x_p \in \{0,1\}, \quad \forall p \in P \tag{3}$$

In the Eq. (1), c_p indicates cost value for each crew pairing in P and x_p is the decision variable which indicates whether the crew pairing is in the solution set. Hence, Eq. (1) gives the total cost of the solution. Eq. (2) is an inequality constraint which guarantees the full covering of all flights in F. If Eq. (2) gives a value greater than 1 for a flight in F, that flight is a dead-head flight. And Eq. (3) represents standard constraints that designate ranges for decision variables of the problem.

3.2. Outline

The ultimate aim of the proposed approaches is to find optimized leg orderings (priorities) that would eventually generate solutions of good quality when pairings were generated in that order. Both of the developed approaches use some common codes for their initialization and solution generation phases. As can be seen from Fig. 1 both of them start with the same main initialization routine and run almost the same solution generation procedure. They only differ in the internal mechanism they use for leg orderings.

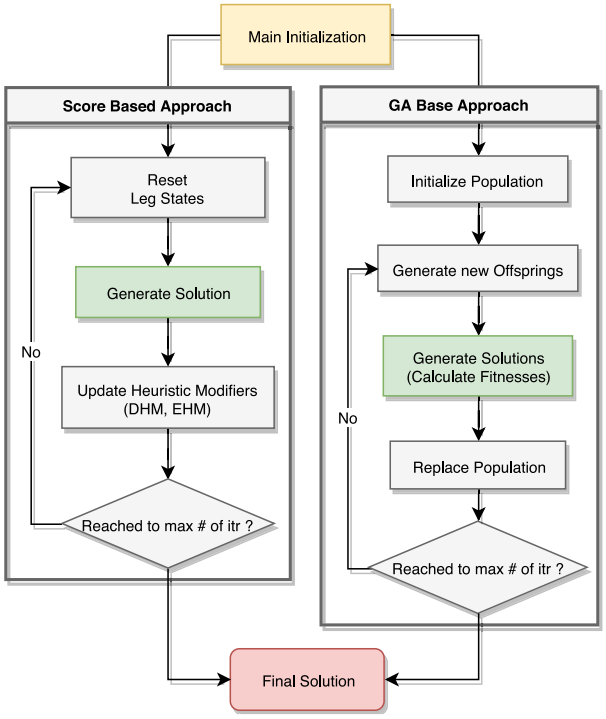


Fig. 1. Main schema of the proposed approaches.

The score based approach (SBH) uses a score based dynamic ordering that was achieved by calculating difficulty score for each leg after each pairing addition to the solution. Difficulty score used in this study was calculated using a combination of some sub-heuristics such as *CoverabilityDegree (CD)*, *DeadheadingHeuristicModifier (DHM)* and *ActivityHeuristicModifier (AHM)*.

The GA based approach obtains leg orderings directly from chromosomes and manipulates ordering information coded on the chromosomes using GA operators during the optimization.

3.3. Initialization

In initialization, both approaches run the same phases as depicted in Fig. 2. Initialization retrieves all the legs as input for the optimization period, first.

Then the optimization context is formed to store all the information generated from input data; including leg lists, duty lists, duty network and rule implementations. This process is followed by the initialization of the rule engine which is responsible for all sort of validation checks. After that, all the airports are extracted from the leg data and all the legs are validated and classified. Then all the valid duties are generated using a depth-first-search implementation. Every duty instance is validated using the rule engine during the duty generation. After generating all the possible duties, duty network which was intensively used during the pairing generation (shortest path) step of the approaches is built. For the details of the duty network used in this study, readers can refer to Zeren and Özkol (2016). After the duty network is built, pairing generation check for all duties and legs is performed in order to know whether a duty and/or leg could be covered by a valid pairing before the optimization phase starts. At the last step of the initialization phase, default values of associated state variables of all legs are calculated and set. The default values calculated in this phase are used at the beginning of each iteration to reset/assign leg states. Therefore, instead of calculating these values in each iteration, using already calculated default leg state values significantly improves the performance of the approach.

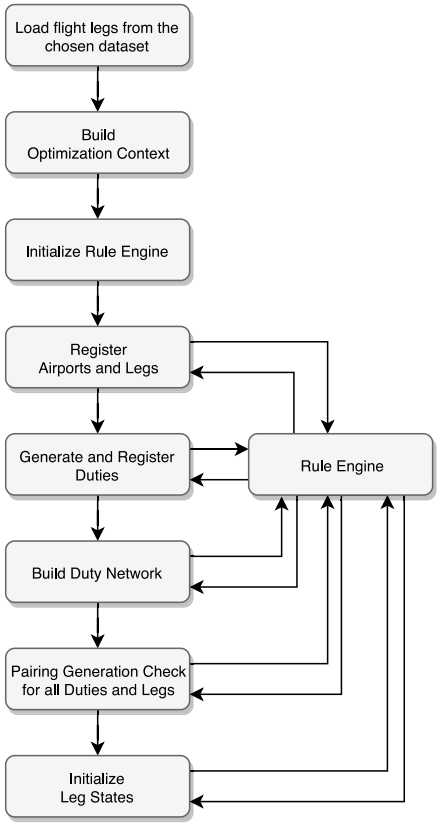


Fig. 2. Initialization of the system.

3.4. Rule engine

The rule engine is designed as a separate component within the approach for improved manageability and maintainability. To control and validate each step of the airport registration, leg registration, duty generation, and pairing generation, following standard interfaces are developed.

- $Introducer$ is used to initialize and classify instances of airports and legs before they are registered. Some attributes of airports such as location (international or domestic), status (home base or not) are determined by calling all registered implementations of this interface.
- $ConnectivityChecker$: A leg to leg connection while building a duty or, a duty to duty connection while building a crew pairing can be established by calling all registered implementations of this interface.
- $AppendabilityChecker$: Addition of a leg to a duty or, addition of a duty to a pairing can be possible after calling all registered implementations of this interface.

3.5. LegState

$LegState$ is a special data structure that was used to keep the status of legs at any time and at any step of the solution generation procedure. It is always maintained during the optimization and some critical values related to legs are stored. The most important $LegState$s are as follows:

- $NumOfCoverings$ (NoC): Indicates how many times a leg was covered. At the beginning of each iteration, this state is reset to zero "0" for all the legs. During the solution generation phase, it is updated regularly after each pairing generation. When a crew pairing was generated, NoC values of all the legs of that pairing

are increased by one "1". If NoC value of a leg was greater than one "1", this means that leg was covered by more than one pairing and except one of those pairings, all the other pairings would include that leg as deadhead.

- $NumOfInclPairsWoDh$ (NoP): Indicates how many crew pairings left that would cover a particular leg in the search space, without causing any deadheads. Like NoC, an instance of NoP variable is stored and updated for each leg in the timetable.

3.5.1. LegState initialization

During the system initialization default values of all leg state variables are calculated and set. Default values of NoC variables of all legs in the system is zero "0". But calculation of default values of NoP variables is a more computationally expensive task. As it is illustrated in Alg. 1, all the possible combinations of pairings are enumerated to calculate NoP values of each leg.

As it is discussed in the experimental results section, enumeration of all possible pairings requires high computational power and significantly reduces performance of the algorithm. With the help of performance benchmarks performed, only crew pairings that include one or two duties were enumerated. Therefore enumeration for crew pairings with three or more duties are not carried out.

As illustrated in Alg. 1, initialization process to calculate default values of NoP variables is started with generating an empty $Pair$ on line 2. On line 3–5, all the deadhead free, home base departed duties are evaluated in a loop. On line 6, the duty found is added to the $Pair$. On line 7–8, if $Pair$ was a one duty pairing, NoP values of the legs that are included in $Pair$ is increased by calling $incLegState()$ procedure. If the $Pair$ was not complete, on line 10, recursive procedure $findPairings()$ is called to find connection duties that would compose new complete pairings. In $findPairings()$ procedure, algorithm searches for connection duties that will eventually generate complete $Pair$s and increase NoP variables if any new pairing was found.

In all steps of this enumeration procedure, rule engine was intensively called to validate pairings enumerated.

Algorithm 1 *LegState* initialization

```
1: procedure initLegStates(Duties)
2:     Pair ← {}
3:     for all Duty in Duties do
4:         if Duty has no deadheads then
5:             if Duty starts from the home base then
6:                 Pair ← Pair + Duty
7:                 if Duty ends at the home base then
8:                     incLegState(Pair)
9:                 else
10:                    findPairings(Pair, Duties)
11:                Pair ← Pair - Duty
12: procedure findPairings(Pair, Duties)
13:    for all Duty in Duties that are addable to Pair do
14:        if Duty has no deadheads then
15:            Pair ← Pair + Duty
16:            if Duty ends at the home base then
17:                incLegState(Pair)
18:            else
19:                findPairings(Pair, Duties)
20:            Pair ← Pair - Duty
21: procedure incLegState(Pair)
22:    for all Leg in Pair do
23:        nop[Leg] ← nop[Leg] + 1
```

3.5.2. LegState update

After each pairing generation during the optimization, values of all leg state variables are updated. NoC variables are increased by one "1" for all legs that were included in new pairings generated and added to the solution. On the contrary, NoP variables of the affected legs are decreased from the default values they obtained in the beginning of the solution generation.

Similarly, in Alg. 2, update process to decrease values of NoP variables is started with generating an empty $Pair$ on line 2. On line 3–5 all the deadhead free duties that include any of the legs from $NewPair$ are evaluated in a loop. The rest of the procedure is very similar to the initialization procedure described above. The most important difference between initialization and update procedures is that in the initialization procedure, duties that do not start from home base does not trigger the rest of the procedure. Because the aim of the initialization is enumerating all the pairs possible and therefore duties that do not start from home base would be considered in pairings that were already started by home base departed duties. But in the update procedure, as can be seen in lines 13–14 home base arrival duties can trigger the procedure as well.

Since, only crew pairings that include one or two duties were enumerated as described above, for the sake of ease, Alg. 2 was depicted as not to expose the flow needed for duties that neither start nor end at the home base.

Algorithm 2 *LegState* update

```
1: procedure updateLegStates(NewPair, Duties)
2:     Pair ← {}
3:     for all Leg in NewPair do
4:         for all Duty in Duties that include Leg do
5:             if Duty has no deadheads then
6:                 Pair ← Pair + Duty
7:                 if Duty starts from the home base then
8:                     if Duty ends at the home base then
9:                         decLegState(Pair)
10:                    else
11:                        findPairings(Pair, Duties, fw)
12:                else
13:                    if Duty ends at the home base then
14:                        findPairings(Pair, Duties, bw)
15:                Pair ← Pair - Duty
16: procedure findPairings(Pair, Duties, Direction)
17:    for all Duty in Duties that are addable to Pair towards Direction do
18:        if Duty has no deadheads then
19:            Pair ← Pair + Duty
20:            if Direction = fw then
21:                if Duty ends at the home base then
22:                    decLegState(Pair)
23:                else
24:                    findPairings(Pair, Duties, fw)
25:            else
26:                if Direction = bw then
27:                    if Duty starts from the home base then
28:                        decLegState(Pair)
29:                    else
30:                        findPairings(Pair, Duties, bw)
31:            Pair ← Pair - Duty
32: procedure decLegState(Pair)
33:    for all Leg in Pair do
34:        nop[Leg] ← nop[Leg] - 1
```

3.6. Solution generation

Solution generation is a common phase for both of the approaches. At each iteration of the score based approach *(SBH)*, a complete set of crew pairings that constitutes a complete solution to the crew pairing problem is generated. Similarly in the GA based approach, complete solution instances for each newly generated chromosome is built and therefore fitness calculation is done.

As illustrated in Fig. 3, solution generation phase is started with the leg list provided and process is triggered with the selection of a flight leg.

In the score-based approach, this flight leg selection procedure was accomplished using dynamically calculated difficulty scores. An uncovered leg with the highest difficulty score is selected first for the pairing generation. After generation of a pairing for the selected flight leg, all the related leg states and therefore difficulty scores are updated. After leg states were updated, a new uncovered leg is selected using updated difficulty scores and same procedures are applied until no leg left remains uncovered.

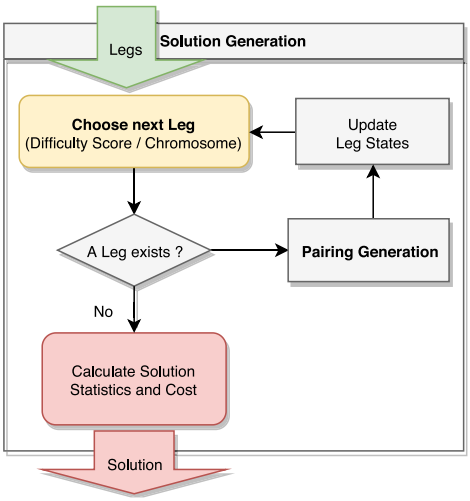


Fig. 3. Typical solution generation flow.

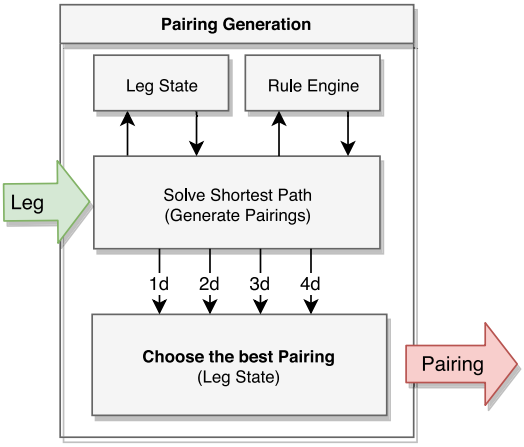


Fig. 4. Typical pairing generation flow.

In the GA based approach, leg ordering is determined by the chromosomes and there is no dynamic component that updates the order. The rest of the procedure is like the score based approach as described above.

3.7. Pairing generation

Pairing generation is the most important component of the solution generation phase. As can be seen from Fig. 4, pairing generation is triggered with a leg. The input leg is chosen according to the difficulty scores in the *(SBH)* approach or the leg ordering in chromosomes for the *GA* based approach. Then RCSP is solved over the sub network that was extracted by exploring the full duty network. And finally the best pairing that would give best NoP values for the rest of the solution is selected.

The first step of the pairing generation is the network explore to find best paths that eventually would generate best pairings for the selected leg. As can be seen from Fig. 5, first, all root duties that include selected leg are fetched. Then they are ordered according to their position to the home base. First, one day duties (SE) and home base departed duties (S) are ordered and placed on the first positions. Then connection duties (C) that neither start nor end at the home base are placed. And finally home base arrival duties (E) are placed to the last positions.

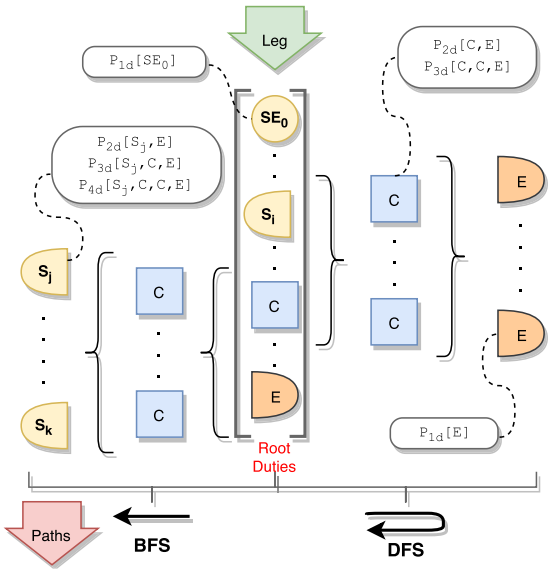


Fig. 5. Flow of network explore.

First, the duty network is explored towards forward direction in the time line starting from non home base arrival duties (S + C) that constitute the first part of the root duties. During this search, dept-first-search (DFS) algorithm was used and the best possible paths for all duty nodes were stored in a data structure. For each duty node, maximum three "3" (maximum pairing day limit - 1) paths were stored as to include best alternatives for all length combinations of pairings. These paths include all the connected duties starting from the node owner to home base arrival duties (E). At the end of DFS phase, many paths that would generate complete pairings (S→E) and incomplete paths (C→E, E) that would generate incomplete pairings those starting duties have not been found yet are found.

The last step of the network explore phase is completing the incomplete paths by directing the search procedure in the backward direction in the time line. During the backward search breath-first-search (BFS) algorithm was used and home base departed duties (S) were searched for the incomplete paths found. At the end of BFS phase, best possible paths of all combinations possible starting from all home base departed duties (S) are found. As shown in Fig. 5, $\{SE_0, \ldots, S_i, \ldots, S_j, \ldots, S_k\}$ are the starting nodes of paths that would yield possible pairings.

During the DFS and BFS phases, paths and their quality are checked and best ones are stored. Eq. (6) was used to compare qualities of two paths of the same length met on the same node. The first term Eq. (4) calculates the difference of number of already covered legs (NoC) of the path candidates. Higher numbers would mean more deadheads generated. Therefore the path with the small NoC value is preferred in first two lines of Eq. (6). If NoC values of the paths were equal, Eq. (5) is calculated. Eq. (5) gives the difference of total block time (bt) of active legs (l_{act} legs that are not deadheads) of the paths. Higher values of active block time (abt) is preferable because duties and pairings with higher (bt) values would cover more legs and eventually would help to decrease number of duties (total man day) of the solution. Total man day is considered as one of the most important cost factors in crew pairing generation. Therefore the path with the higher abt value is preferred in 3th and 4th lines of Eq. (6).

$$difference_{\text{dh}}(p_1, p_2) = \sum_{l \text{ in } p_1} NoC_l - \sum_{l \text{ in } p_2} NoC_l. \tag{4}$$

$$difference_{\text{act}}(p_1, p_2) = \sum_{l_{\text{act}} \text{ in } p_1} bt_l - \sum_{l_{\text{act}} \text{ in } p_2} bt_l. \tag{5}$$

$$getBest(p_1,p_2)=\begin{cases}p_1, & \text{if } (difference_{\text{dh}}(p_1,p_2)<0).\\ p_2, & \text{if } (difference_{\text{dh}}(p_1,p_2)>0).\\ p_1, & \text{if } (difference_{\text{act}}(p_1,p_2)>0).\\ p_2, & \text{if } (difference_{\text{act}}(p_1,p_2)<0).\\ p_1, & \text{otherwise.}\end{cases} \tag{6}$$

During the DFS and BFS phases, some tree pruning techniques were employed for performance improvement purposes. First, all the nodes visited during the DFS and BFS are marked and if search algorithm revisits that node later, all the sub tree calculations that has already been done starting from that node would be prevented. The second strategy is based on doing quality checks with the paths that already found. If a node would not yield a better path in terms of quality metrics, that node would be disregarded and sub tree search from that node would be prevented.

At the end of the pairing generation phase a list of best paths $SE_0,\dots,S_i,\dots,S_j,\dots,S_k$ (Fig. 5) are found. From this list, best paths that can generate valid crew pairings are selected using Eq. (6). And finally four "4" best valid crew pairings are generated with different lengths (1d, 2d, 3d, 4d). At the last step one of these four pairings is selected for adding to the generated solution.

To decide which pairing to add Eq. (7) was used in both approaches. In Eq. (7) algorithm calculates how NoP would change for all the uncovered legs (l_{ucvr}) in the system and chooses the pairing that would generate the maximum NoP value at the end.

$$\max_{p_{1\text{d},2\text{d},3\text{d},4\text{d}}}[\min_{l_{\text{ucvr}}}[NoP(l)]]. \tag{7}$$

In this strategy it was aimed to keep the amount of diversity high in search space and not stuck with less quality options and more deadheads in early stages during the optimization.

4. Proposed approaches

As already outlined before there are two high level heuristics were developed and compared in this study (Fig. 1).

4.1. Score-based adaptive greedy heuristic

The main difference of the score based heuristic is that it controls optimization's direction based on the difficulty scores calculated adaptively for each leg in the system as a solution is constructed. Difficulty score was calculated as described in Eq. (8).

$$\begin{aligned}diffScore_l(t)&=w_{\text{CD}}\times norm(CD_l)\\&=w_{\text{DHM}}\times norm(DHM_l(t))\\&=w_{\text{AHM}}\times norm(AHM_l(t)).\end{aligned} \tag{8}$$

where,

$$\begin{aligned}norm(CD_l)&=\frac{CD_l}{CD_{\max}}\\ norm(DHM_l(t))&=\frac{DHM_l(t)}{DHM_{\max}(t)}\\ norm(AHM_l(t))&=\frac{AHM_l(t)}{AHM_{\max}(t)}\end{aligned} \tag{9}$$

$$w_{\text{CD}}+w_{\text{DHM}}+w_{\text{AHM}}=1$$

As seen in Eq. (8), difficulty score used in this study has three components. All of the components contribute to the difficulty score with the associated weights. The first component is $CoverabilityDegree$ (CD) which is a dynamic component that is based on NoP values of the legs and updated after each pairing addition to the solution. However this component is not enough for managing the direction of the optimizer. Other two components are used to manage optimization direction and avoid generating same or similar pairings/solutions. $DeadheadingHeuristicModifier$ (DHM) and $ActivityHeuristicModifier$ (AHM) are the components that are updated at the end of each iteration based on deadheading and pairing quality metrics. All these terms are also normalized as seen in Eq. (9) with the maximum values of the values obtained at the current solution step.

- Coverabilty Degree (CD)
  $CoverabilityDegree$ (CD) of a leg and its maximum value to calculate the final normalized CD ($norm(CD_l)$) was calculated using Eq. (10). As can be seen from the formula, higher values of (CD) mean existence of less number of pairings and high difficulty score for an associated leg.

$$\begin{aligned}CD_l&=NoP_{\max}-NoP_l\\ CD_{\max}&=NoP_{\max}\end{aligned} \tag{10}$$

- Heuristic Modifiers
  Heuristic modifiers are the main factor that drives the optimization direction with the associated weights. Without heuristic modifiers optimization would generate the same solution at each iteration. They also can be considered as a learning mechanism where outputs of the important KPIs (Key Performance Indicators) (DHM, AHM) are updated in each iteration and used to form new solutions. There are two heuristic modifier terms in the main difficulty score equation (Eq. (8)). These terms are updated at the end of each iteration after solution generation phase was completed.
  - Deadheading Heuristic Modifier (DHM)
    In the beginning of the optimization this value equals to zero "0" for all legs. After each solution generation during the optimization, legs' DHM values are increased if any associated leg had deadhead repetitions in the solution. DHM was updated using Eq. (11).

$$DHM_l(t+1)=DHM_l(t)+NoC_l-1. \tag{11}$$

  - Activity Heuristic Modifier (AHM)
    In the beginning of the optimization this value equals to zero "0" for all legs. After each solution generation during the optimization, legs' AHM values are increased if any associated leg was covered by a duty that has less active block time (ABT_{d_l}) than a specified threshold (ABT_{th}). AHM was updated using Eq. (12). ABT_{th} value used in this study was 4 h.

$$AHM_l(t+1)=\begin{cases}AHM_l(t)+ABT_{\text{th}}-ABT_{d_l}, & \text{if } (ABT_{d_l}<ABT_{\text{th}})\\ AHM_l(t), & \text{otherwise.}\end{cases} \tag{12}$$

4.2. GA-based hyper-heuristic

GA is a widely studied technique used to solve wide range of optimization problems. In this study, GA was used to solve crew pairing problem which is a well known combinatorial optimization problem. As seen in Fig. 1, in this approach, GA is used as an ordering optimizer. It basically sequences the legs in the chromosomes to produce high quality solutions running the solution generation procedure explained earlier using those ordered legs. The operators employed in this approach were as follows:

- Row based chromosome design
  In this study row-based chromosome design was used. As can be seen from Fig. 6, in this design each gene in the chromosome represents a different leg. During the fitness calculation (solution generation) of each newly generated chromosome, leg orders provided by chromosomes were used. Naturally, lengths of all chromosomes generated were equal to number of legs in the system.

Fig. 6. Row based chromosome.

- Cycle crossover
  Cycle crossover was used as a crossover operator. It first identifies a number of cycles between two selected parent chromosomes and then to form child chromosomes these cycles are copied from parent chromosomes in turn. For more detail readers can consult to Oliver et al. (1987).
- Swap mutation
  In this approach, a straight-forward random gene swap operator was used as a mutation operator. In this implementation, all the genes are visited in a loop. For each gene, a random number is generated between [0–1). If the number generated was below the mutation rate chosen, swap operation is performed with another randomly selected gene from the same chromosome. In this study $mrate = 0.02$ was used as the mutation rate.
- Binary tournament selection
  Parent chromosomes to put into crossover operation to generate new chromosomes are selected using binary tournament selection. In this implementation two chromosomes are randomly selected and the best one are chosen as the first parent. The same selection is done to determine the second parent. And eventually two selected parents are put into the cycle crossover operation to generate new offsprings.
- Population and replacement
  Because of the large size of the problems solved in this study, the population size was limited to 20. Elitist strategy which always keeps certain amount of best chromosomes in the population was used. For doing that in each new generation two "2" of the best chromosomes were fixed at the top of the next generation and the rest of it was selected using the same binary tournament selection.

4.3. Solution evaluation

In both of the approaches Eq. (13) was used to compare qualities of two solutions. Two main terms are used in Eq. (13). First one is deadhead related term $dhDiff$ and the second one activity cost related term $actDiff$. $dhDiff$ term basically calculates differences of number of deadhead legs between two solutions. $actDiff$ term calculates total amount of difference in terms of active block time shortage in all duties between two solutions. The first term penalizes deadheads while the other one penalizes duties with less active block time than a specified threshold value (ABT_{th}) and helps to decrease number of duties needed and increases crew utilization. Both terms were multiplied by new weights that are calculated using w_{DHM} and w_{AHM} which were previously used in difficulty score calculations. Since Eq. (13) does not include the other weight w_{CD} which exists in Eq. (9), new weights were normalized with each other. Thus both, difficulty score calculations and solution cost evaluation were linked with the usage of the same weights.

$$getBest(s_1, s_2) = \begin{cases} s_1, & \text{if } \left(\dfrac{w_{\text{DHM}}}{w_{\text{DHM}} + w_{\text{AHM}}} \times dhDiff(s_1, s_2) + \dfrac{w_{\text{AHM}}}{w_{\text{DHM}} + w_{\text{AHM}}} \times actDiff(s_1, s_2) \right) \leq 0 \\ s_2, & \text{otherwise.} \end{cases} \tag{13}$$

where,

$$dhDiff(s_1, s_2) = \frac{\sum_{l \text{ in } s_1} (NoC_l - 1) - \sum_{l \text{ in } s_2} (NoC_l - 1)}{\max \left[\sum_{l \text{ in } s_1} (NoC_l - 1), \sum_{l \text{ in } s_2} (NoC_l - 1) \right]} \tag{14}$$

Table 3
Test data sets.

Month	Legs (fleet)	Leg connections (valid/total)	Duties
Jan	17 268	548 532/3 669 342	416 153
Feb	15 738	511 362/3 405 856	393 285
Mar	17 318	501 518/3 659 411	390 973
Apr	16 833	615 542/4 233 568	439 661
Jun	16 127	651 823/4 487 894	406 233
Jul	15 656	678 563/4 683 294	401 527

$$actDiff(s_1, s_2) = \frac{\sum_{d \text{ in } s_1} Act_d - \sum_{d \text{ in } s_2} Act_d}{\max \left[\sum_{d \text{ in } s_1} Act_d, \sum_{d \text{ in } s_2} Act_d \right]} \tag{15}$$

$$Act_d = \begin{cases} ABT_{\text{th}} - ABT_d, & \text{if } (ABT_d < ABT_{\text{th}}) \\ 0, & \text{otherwise.} \end{cases} \tag{16}$$

In Eqs. (14) and (15) calculation details including how they were normalized of both terms are shown. And Eq. (16) shows how Act_d is calculated using active block time threshold value ABT_{th} for each duty in solutions.

5. Experimental results

The timetable data used in this study obtained from Turkish Airlines and it was extensively studied in Zeren and Özkol (2016). The same rules were implemented and applied during this study.

5.1. Data

The timetable data used in this study covers period of first six months of 2014. Therefore data of various seasons were used. Details of the data used is presented in Table 3 where months, number of legs, number of valid and total leg connections that were checked during duty generation and total number of duties can be seen. Number of legs information covers all the legs from the planning month and first ten days of the following month that is added to the planning period to be able to generate pairings which can touch to the following month. Optimization runs performed for this study and the metrics given in Table 3 cover only cockpit 320 fleet which is a typical large scale fleet that is composed of narrow body aircrafts. The algorithm also incorporates flight legs from other fleets into account and produces mixed duties with deadhead legs from other fleets.

5.1.1. Home base

Mainly because of the rules and the network structure of the time period and the data given, existence of single home base was considered. Therefore there is no term added to the equations for providing workload distribution balance between different home bases. As can be seen from the anonymized data, there are two airports at the home base and pairings are allowed to start from one airport and to end at the other one.

5.1.2. Costs, KPIs and weight sets

As described in previous sections, there are two main cost factors (KPIs) considered. The first one is deadheads and the second one is poor quality duties (duties with less active block time). In this study, other non-significant cost factors were disregarded because of sake of ease and to increase measurability of the system response of applying different weights.

Indeed, "number of deadheads" and "total man day" are the most important KPIs that crew planners pay attention the most. Number of deadheads are critical especially in high seasons when existence of available seats at aircrafts became extremely important. Number of deadheads also decreases crew utilization by using crew in passive flights that are also counted as regular duties even though the deadhead

Table 4
Duty limits for NoP calculations and statistics.

Duty limit	Pairings	Deadheads	Man day	Avg. Time Per Itr.
1	26 073	192	15 702	3 min
2	781 483	87	15 354	4 min
3	26 843 846	81	15 311	8 min
4	352 554 711	72	15 306	13 min

Table 5
Scenarios and weights used.

Scenario	Difficulty score			Solution comparison	
	w_{CD}	w_{DHM}	w_{AHM}	$\frac{w_{DHM}}{w_{DHM}+w_{AHM}}$	$\frac{w_{AHM}}{w_{DHM}+w_{AHM}}$
S_{dh}	0.9	0.08	0.02	0.8	0.2
S_{act}	0.9	0.02	0.08	0.2	0.8

crew were not active. Total man day is an other KPI that determines how many crew is needed to be able to operate properly one month period of timetable. Its importance increases towards end of the year and can be become extremely vital in last two months. The main reason is decrease in total number of available crew because of annual flying time limits.

In the score based approach, these two KPI values were optimized using DHM, AHM and their associated weights w_{DHM} and w_{AHM}. CD and its weight w_{CD} were used to manage search space's diversity and size. In the GA based approach, just w_{DHM} and w_{AHM} were used like described in Eq. (13). To manage search space's diversity and size NoP values were used.

As described before, enumeration of all possible pairings to calculate NoP values for each leg requires too much computational power. To decide optimum limit a number of experimental runs performed as seen in Table 4. These runs were performed using January data and 20 iterations completed for each setting. With the help of tests runs performed in Table 4, number of duties limit per pairing for NoP calculations was set to 2 by checking the trade-off between performance and solution quality.

There were two scenarios determined: S_{dh} was composed to see the effect of increased w_{DHM} and S_{act} was composed to see the effect of increased w_{AHM} like shown in Eq. (13). Same values for w_{DHM} and w_{AHM} were used in both approaches since these weights are normalized. After extensive tuning runs the value for w_{CD} was determined as 0.9 for score based approach runs as can be seen in Table 5. It basically shows how important keeping the search space diverse and giving high priority to the legs that has less alternative pairs during the optimization.

5.2. Experiments

A Java library was developed for this study and a typical iteration (solution generation procedure) takes around 4 min for all data sets used. Proposed approaches were run for 100 iterations and each session took around 6 h for each data and each scenario used.

Table 6 includes some of the important statistical and KPI values obtained from the runs performed using score based approach. Columns in Table 6 represent each month's datasets and the first group of rows that belong to scenario S_{dh} runs represent number of pairings, number of duties, number of active legs, number of man days and number of deadheads respectively. Similarly the other group of rows represent the same metrics that belong to scenario S_{act} runs.

Unlike score based approach, because of the random nature of the GA based approach, runs belong to that approach were performed 30 times for each scenario and data set separately. Similarly, Tables 7, 8 and 9 includes maximum, average and minimum values of the same metrics belong to each scenarios and months sequentially.

Results shown in Tables 6, 7, 8 and 9 were comprehensively investigated by planning specialists from Turkish Airlines and it was reported that solution quality provided is very encouraging for further developments and comparable to current practices that are being carried out by Turkish Airlines using state of the art planning systems. It is also observed that especially score based approach is capable of generating comparable but less number of deadheads than the heuristic approach proposed by Erdogan et al. (2015) that used similar but smaller data sets.

Fig. 7a, b, c and d shows converge of KPI values for both scenarios. Fig. 7a, b, c and d were generated using outputs of Jan runs and the

Table 6
Statistics and KPI values of score based approach runs.

Scenario		Jan	Feb	Mar	Apr	May	Jun
S_{dh}	Pairs	5 254	4 700	5 263	4 893	4 793	4 626
	Duties	6 079	5 371	6 069	5 941	5 865	5 645
	LegsActive	13 780	12 314	13 851	13 308	12 962	12 414
	ManDays	15 034	13 300	14 942	14 650	14 584	14 096
	DH	66	54	78	79	93	99
S_{act}	Pairs	5 158	4 672	5 245	4 851	4 898	4 673
	Duties	6 057	5 357	6 048	5 926	5 848	5 601
	LegsActive	13 820	12 322	13 833	13 446	13 107	12 567
	ManDays	14 990	13 272	14 904	14 632	14 538	14 018
	DH	163	86	142	168	200	164

Table 7
Statistics and KPI values of GA based approach (Jan–Feb).

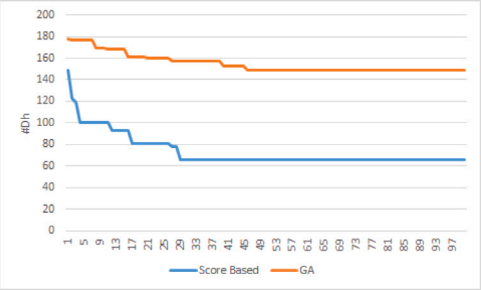
Scenario		Jan			Feb		
		Max	Avg	Min	Max	Avg	Min
S_{dh}	Pairs	5 383	5 370	5 358	4 849	4 833	4 820
	Duties	6 115	6 104	6 087	5 418	5 406	5 392
	LegsActive	13 843	13 773	13 746	12 316	12 316	12 316
	ManDays	15 580	15 558	15 522	13 812	13 787	13 760
	DH	157	145	122	125	117	111
S_{act}	Pairs	5 356	5 340	5 316	4 822	4 809	4 799
	Duties	6 085	6 075	6 063	5 393	5 377	5 365
	LegsActive	13 792	13 778	13 764	12 254	12 254	12 254
	ManDays	15 504	15 485	15 460	13 746	13 717	13 694
	DH	252	238	208	196	193	184

Table 8
Statistics and KPI values of GA based approach (Mar–Apr).

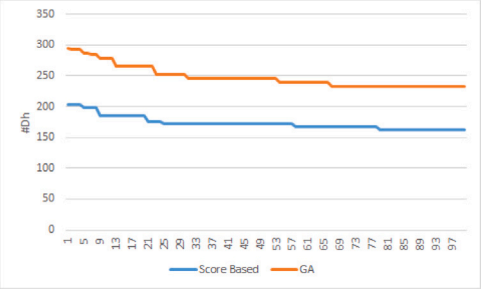
Scenario		Mar			Apr		
		Max	Avg	Min	Max	Avg	Min
S_{dh}	Pairs	5 414	5 394	5 374	5 088	5 074	5 062
	Duties	6 100	6 088	6 072	6 124	6 102	6 085
	LegsActive	13 874	13 873	13 864	13 433	13 423	13 412
	ManDays	15 490	15 468	15 432	15 508	15 459	15 416
	DH	145	138	130	134	124	113
S_{act}	Pairs	5 378	5 363	5 347	5 086	5 070	5 057
	Duties	6 072	6 057	6 045	6 114	6 095	6 076
	LegsActive	13 814	13 806	13 805	13 419	13 410	13 396
	ManDays	15 422	15 390	15 370	15 506	15 443	15 396
	DH	240	227	215	221	206	194

Table 9
Statistics and KPI values of GA based approach (May–Jun).

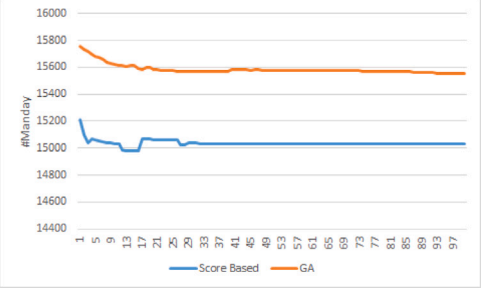
Scenario		May			Jun		
		Max	Avg	Min	Max	Avg	Min
S_{dh}	Pairs	5 030	5 016	5 005	4 800	4 781	4 755
	Duties	6 063	6 046	6 026	5 830	5 811	5 782
	LegsActive	13 124	13 120	13 114	12 586	12 575	12 556
	ManDays	15 468	15 431	15 366	14 950	14 899	14 840
	DH	150	138	128	156	148	140
S_{act}	Pairs	5 025	5 011	4 997	4 792	4 775	4 763
	Duties	6 057	6 041	6 023	5 833	5 803	5 785
	LegsActive	13 110	13 107	13 101	12 573	12 563	12 547
	ManDays	15 454	15 420	15 376	14 948	14 880	14 818
	DH	246	228	213	253	242	221



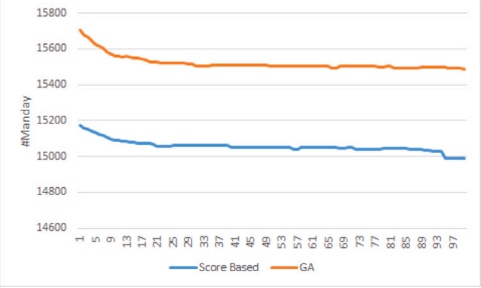
(a) Dh converge of S_{dh} runs.



(b) Dh converge of S_{Act} runs.



(c) ManDay converge of S_{dh} runs.



(d) ManDay converge of S_{Act} runs.

Fig. 7. Convergence of KPIs.

other test periods show very similar behavior. As can be seen from Fig. 7a, b, c and d; score based approach is far more successful on reaching better KPI values. For most of the cases GA approach could not produce results even at the final iteration better than the first iterations of the score based approach. This situation shows that *Coverability-Degree(CD)* which is the main metric used to decide which leg to be covered first is a considerably important metric that helps to converge better positions much earlier. Another important factor is score based approach produced better results even though GA approach has four times more number of evaluations because of four child chromosome evaluation in each iteration.

Fig. 8a shows visual comparison of number of deadhead differences between two approaches for the scenario S_{dh} which gives higher priority to minimizing number of deadheads. It is obviously seen that score based approach performs around two times better especially for the data that belongs to the first three months and it is significantly outperforms on the rest of the test period. According to results GA based approach generated around 60 more deadhead legs than score based approach and this situation could cause profit loss especially in high seasons when empty seats could be very valuable.

Similarly, Fig. 8b shows visual comparison of number of deadhead differences between two approaches for the scenario S_{act} which gives higher priority to minimizing number of mandays. Score based approach again outperforms for all months in the test period.

Fig. 8c and d show visual comparison of number of manday differences between two approaches for the scenario S_{dh} and scenario S_{act} respectively. Score based approach produced solutions of higher quality in terms of number of mandays as well. The values of these runs could be thought close to each other but the average difference 650 man day means significant amount of over utilization of crew and can lead to lack of available crew if the GA based schedules are taken into practice.

Obviously, the score based approach is capable of generating solutions of higher quality than the GA approach for all test cases. With the usage of additional KPI based sub-heuristics DHM and AHM, it much more efficiently explores the problem search space than the GA approach.

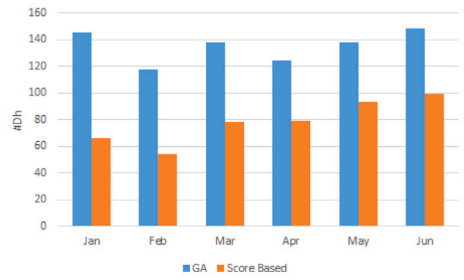
Nevertheless, the new GA approach proposed in this study is capable of generating high quality solutions for large instances of the crew pairing problems compared to classical models like studied in Zeren and Özkol (2012) which was successful only on small and mid size fleets.

6. Conclusion

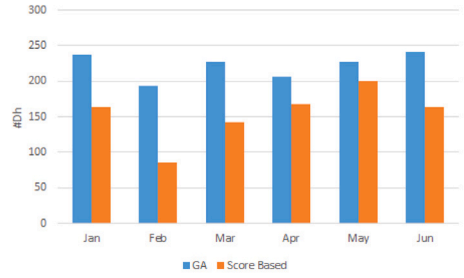
In this research study, widely known airline crew pairing problem was revisited and a comprehensive research was carried out on alternative solution approaches that can handle large scale instances of the problem. Two different solution approach based on leg orderings (priorities) were developed. The first approach developed is a score based adaptive greedy method which is inspired from squeaky-wheel optimization technique. It calculates difficulty scores for each leg and updates the leg orderings based on these difficulty scores during optimization. A resource constrained shortest path implementation was also developed to generate high quality crew pairings. This procedure searches for high quality pairings with high active block time and less deadheads in a greedy way. Besides, among the best alternative crew pairings with different lengths found, it selects the crew pairing that would keep rest of the search space as wide as possible. Score based approach produced solutions of considerably high quality for all six monthly timetables used. The second approach developed is a GA based heuristics that manages and optimizes leg orderings through genetic operators. This approach also produced comparably high quality solutions with the row based chromosome and same resource constrained shortest path procedure.

This study also shows importance of the metric called *Coverability-Degree(CD)* that constitutes the main component of the difficulty score calculation and the decision mechanism that is carried out at the end of the shortest path procedure developed. Besides shortest path procedure, score based approach that additionally uses $CoverabilityDegree$ (CD) metric in difficulty score calculations compared to GA approach is capable of generating solutions of considerably higher quality.

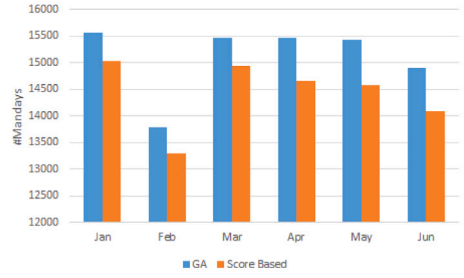
Promising solution qualities obtained in this study show that a well designed heuristics/meta-heuristics could generate high quality solutions even for large scale problem instances. Besides, proposed approaches open new doors for further developments in the field of heuristics for the large scale problems. They could be hybridized with current practices like column generation heuristics to improve their performance, solution quality and stabilization characteristics further.



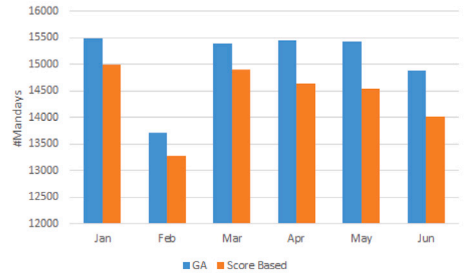
(a) Dh comparison of S_{dh} runs.



(b) Dh comparison of S_{Act} runs.



(c) ManDay comparison of S_{dh} runs.



(d) ManDay comparison of S_{Act} runs.

Fig. 8. Key performance indicators.

They can also be further developed by introducing new well designed sub-heuristics, KPIs and also significant performance improvements can be achieved with parallelization of the algorithm.

Contributing to the literature and to further developments by making the timetable data used public is an other desired output of this study. This situation could cause more publications and more opportunities for healthy comparison of different studies done in the field.

CRediT authorship contribution statement

Bahadır Zeren: Conceptualization, Data curation, Methodology, Software, Validation, Formal analysis, Writing – original draft, Writing – review & editing, Visualization. **Ender Özcan:** Supervision, Methodology, Validation, Writing – original draft, Writing – review & editing. **Muhammet Deveci:** Investigation, Writing – original draft, Writing – review & editing, Visualization.

Acknowledgments

This project has been supported by The Scientific and Technological Research Council of Turkey (TUBİTAK) BIDEB-2219 under the Postdoctoral Research Program grant number 1059B191701169.

References

Agustin, A., Juan, A., Pardo, E.G., 2017. A variable neighborhood search approach for the crew pairing problem. Electron. Notes Discrete Math. 58, 87–94.
Ahmed, M.B., Hryhoryeva, M., Hvattum, L.M., Haouari, M., 2022. A matheuristic for the robust integrated airline fleet assignment, aircraft routing, and crew pairing problem. Comput. Oper. Res. 137, 105551.
Anbil, R., Gelman, E., Patty, B., Tanga, R., 1991. Recent advances in crew-pairing optimization at American airlines. Interfaces 21 (1), 62–74.
Andersson, E., Housos, E., Kohl, N., Wedelin, D., 1998. Crew pairing optimization. In: Operations Research in the Airline Industry. Springer, pp. 228–258.
Arabeyre, J., Fearnley, J., Steiger, F., Teather, W., 1969. The airline crew scheduling problem: A survey. Transp. Sci. 3 (2), 140–163.
Aydemir-Karadag, A., Dengiz, B., Bolat, A., 2013. Crew pairing optimization based on hybrid approaches. Comput. Ind. Eng. 65 (1), 87–96.
Azadeh, A., Farahani, M.H., Eivazy, H., Nazari-Shirkouhi, S., Asadipour, G., 2013. A hybrid meta-heuristic algorithm for optimization of crew scheduling. Appl. Soft Comput. 13 (1), 158–164.
Baker, E., Fisher, M., 1981. Computational results for very large air crew scheduling problems. Omega 9 (6), 613–618.
Barnhart, C., Cohn, A.M., Johnson, E.L., Klabjan, D., Nemhauser, G.L., Vance, P.H., 2003. Airline crew scheduling. In: Handbook of Transportation Science. Springer, pp. 517–560.
Barnhart, C., Hatay, L., Johnson, E.L., 1995. Deadhead selection for the long-haul crew pairing problem. Oper. Res. 43 (3), 491–499.
Beasley, J.E., Cao, B., 1996. A tree search algorithm for the crew scheduling problem. European J. Oper. Res. 94 (3), 517–526.
Borndörfer, R., Schelten, U., Schlechte, T., Weider, S., 2006. A column generation approach to airline crew scheduling. In: Operations Research Proceedings 2005. Springer, pp. 343–348.
Burke, E.K., Hyde, M.R., Kendall, G., Ochoa, G., Özcan, E., Woodward, J.R., 2019. A classification of hyper-heuristic approaches: Revisited. In: Gendreau, M., Potvin, J.-Y. (Eds.), Handbook of Metaheuristics. Springer International Publishing, Cham, pp. 453–477.
Burke, E.K., Kendall, G., Soubeiga, E., 2003. A tabu-search hyperheuristic for timetabling and rostering. J. Heurist. 9 (6), 451–470.
Burke, E.K., McCollum, B., Meisels, A., Petrovic, S., Qu, R., 2007. A graph-based hyper-heuristic for educational timetabling problems. European J. Oper. Res. 176 (1), 177–192.
Cohn, A.M., Barnhart, C., 2003. Improving crew scheduling by incorporating key maintenance routing decisions. Oper. Res. 51 (3), 387–396.
Cowling, P., Kendall, G., Han, L., 2002. An investigation of a hyperheuristic genetic algorithm applied to a trainer scheduling problem. In: Evolutionary Computation, 2002. CEC'02. Proceedings of the 2002 Congress on. vol. 2, IEEE, pp. 1185–1190.
Dawid, H., König, J., Strauss, C., 2001. An enhanced rostering model for airline crews. Comput. Oper. Res. 28 (7), 671–688.
Demirel, N.C., Deveci, M., 2017. Novel search space updating heuristics-based genetic algorithm for optimizing medium-scale airline crew pairing problems. Int. J. Comput. Intell. Syst. 10 (1), 1082–1101.
Deng, G.-F., Lin, W.-T., 2011. Ant colony optimization-based algorithm for airline crew scheduling problem. Expert Syst. Appl. 38 (5), 5787–5793.
Desaulniers, G., Desrosiers, J., Dumas, Y., Marc, S., Rioux, B., Solomon, M.M., Soumis, F., 1997. Crew pairing at air france. European J. Oper. Res. 97 (2), 245–259.
Desrosiers, J., Lasry, A., McInnis, D., Solomon, M.M., Soumis, F., 2000. Air transat uses ALTITUDE to manage its aircraft routing, crew pairing, and work assignment. Interfaces 30 (2), 41–53.
Deveci, M., Demirel, N.Ç., 2018a. Evolutionary algorithms for solving the airline crew pairing problem. Comput. Ind. Eng. 115, 389–406.
Deveci, M., Demirel, N.Ç., 2018b. A survey of the literature on airline crew scheduling. Eng. Appl. Artif. Intell. 74, 54–69.

Díaz-Ramírez, J., Huertas, J.I., Trigos, F., 2014. Aircraft maintenance, routing, and crew scheduling planning for airlines with a single fleet and a single maintenance and crew base. Comput. Ind. Eng. 75, 68–78.

Doi, T., Nishi, T., Voß, S., 2018. Two-level decomposition-based matheuristic for airline crew rostering problems with fair working time. European J. Oper. Res. 267 (2), 428–438.

Drake, J.H., Kheiri, A., Özcan, E., Burke, E.K., 2020. Recent advances in selection hyper-heuristics. European J. Oper. Res. 285 (2), 405–428.

Dunbar, M., Froyland, G., Wu, C.-L., 2014. An integrated scenario-based approach for robust aircraft routing, crew pairing and re-timing. Comput. Oper. Res. 45, 68–86.

Emden-Weinert, T., Proksch, M., 1999. Best practice simulated annealing for the airline crew scheduling problem. J. Heuristics 5 (4), 419–436.

Erdogan, G., Haouari, M., Ormeci Matoglu, M., Ozener, O., 2015. Solving a large-scale crew pairing problem. J. Oper. Res. Soc. 66, http://dx.doi.org/10.1057/jors.2015.2.

Gershkoff, I., 1989. Optimizing flight crew schedules. Interfaces 19 (4), 29–43.

Graves, G.W., McBride, R.D., Gershkoff, I., Anderson, D., Mahidhara, D., 1993. Flight crew scheduling. Manage. Sci. 39 (6), 736–745.

Haouari, M., Zeghal Mansour, F., Sherali, H.D., 2019. A new compact formulation for the daily crew pairing problem. Transp. Sci. 53 (3), 811–828.

Herekoglu, A., Kabak, Ö., 2023. Spectral clustering approximation for large scale crew disruption data of an airline company for intelligent crew recovery. J. Soft Comput. Decis. Anal. 1 (1), 139–160. http://dx.doi.org/10.31181/jscda11202315.

Hjorring, C.A., Hansen, J., 1999. Column generation with a rule modelling language for airline crew pairing. In: Proceedings of the 34th Annual Conference of the Operational Research Society of New Zealand. pp. 133–142.

Hoffman, K.L., Padberg, M., 1993. Solving airline crew scheduling problems by branch-and-cut. Manage. Sci. 39 (6), 657–682.

Joslin, D.E., Clements, D.P., 1999. Squeaky wheel optimization. J. Artificial Intelligence Res. 10, 353–373. http://dx.doi.org/10.1613/jair.561.

Kasirzadeh, A., Saddoune, M., Soumis, F., 2017. Airline crew scheduling: models, algorithms, and data sets. EURO J. Transp. Logist. 6 (2), 111–137.

Kendall, G., Hussin, N.M., 2004. A tabu search hyper-heuristic approach to the examination timetabling problem at the MARA university of technology. In: International Conference on the Practice and Theory of Automated Timetabling. Springer, pp. 270–293.

Klabjan, D., 2005. Large-scale models in the airline industry. In: Column Generation. Springer, pp. 163–195.

Klabjan, D., Johnson, E.L., Nemhauser, G.L., Gelman, E., Ramaswamy, S., 2001a. Solving large airline crew scheduling problems: Random pairing generation and strong branching. Comput. Optim. Appl. 20 (1), 73–91.

Klabjan, D., Schaefer, A.J., Johnson, E.L., Kleywegt, A.J., Nemhauser, G.L., 2001b. Robust airline crew scheduling. Proc. Tristan IV 275–280.

Klabjan, D., Schwan, K., 2001. Airline crew pairing generation in parallel.. In: PPSC.

Kohl, N., Karisch, S.E., 2004. Airline crew rostering: Problem types, modeling, and optimization. Ann. Oper. Res. 127 (1–4), 223–257.

Lavoie, S., Minoux, M., Odier, E., 1988. A new approach for crew pairing problems by column generation with an application to air transportation. European J. Oper. Res. 35 (1), 45–58.

Levine, D., 1996. Application of a hybrid genetic algorithm to airline crew scheduling. Comput. Oper. Res. 23 (6), 547–558.

López-Camacho, E., Terashima-Marin, H., Ross, P., Ochoa, G., 2014. A unified hyper-heuristic framework for solving bin packing problems. Expert Syst. Appl. 41 (15), 6876–6889.

Makri, A., Klabjan, D., 2004. A new pricing scheme for airline crew scheduling. INFORMS J. Comput. 16 (1), 56–67.

Mercier, A., Cordeau, J.-F., Soumis, F., 2005. A computational study of benders decomposition for the integrated aircraft routing and crew scheduling problem. Comput. Oper. Res. 32 (6), 1451–1476.

du Merle, O., Villeneuve, D., Desrosiers, J., Hansen, P., 1999. Stabilized column generation. Discrete Math. 194 (1), 229–237. http://dx.doi.org/10.1016/S0012-365X(98)00213-1, URL http://www.sciencedirect.com/science/article/pii/S0012365X98002131.

Muter, İ., Birbil, Ş.İ., Bülbül, K., Şahin, G., Yenigün, H., Taş, D., Tüzün, D., 2013. Solving a robust airline crew pairing problem with column generation. Comput. Oper. Res. 40 (3), 815–830.

Oliver, I.M., Smith, D.J., Holland, J.R.C., 1987. A study of permutation crossover operators on the traveling salesman problem. In: Proceedings of the Second International Conference on Genetic Algorithms on Genetic Algorithms and their Application. L. Erlbaum Associates Inc., Hillsdale, NJ, USA, pp. 224–230, URL http://dl.acm.org/citation.cfm?id=42512.42542.

Pillay, N., Banzhaf, W., 2009. A study of heuristic combinations for hyper-heuristic systems for the uncapacitated examination timetabling problem. European J. Oper. Res. 197 (2), 482–491.

Quesnel, F., Desaulniers, G., Soumis, F., 2017. A new heuristic branching scheme for the crew pairing problem with base constraints. Comput. Oper. Res. 80, 159–172.

Quesnel, F., Desaulniers, G., Soumis, F., 2020. A branch-and-price heuristic for the crew pairing problem with language constraints. European J. Oper. Res. 283 (3), 1040–1054.

ReVelle, C., McGarity, A.E., 1997. Design and Operation of Civil and Environmental Engineering Systems. John Wiley & Sons.

Ross, P., Schulenburg, S., Marín-Bläzquez, J.G., Hart, E., 2002. Hyper-heuristics: learning to combine simple heuristics in bin-packing problems. In: Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation. Morgan Kaufmann Publishers Inc., pp. 942–948.

Saemi, S., Komijan, A.R., Tavakkoli-Moghaddam, R., Fallah, M., 2022. Solving an integrated mathematical model for crew pairing and rostering problems by an ant colony optimisation algorithm. Eur. J. Ind. Eng. 16 (2), 215–240.

Salazar-González, J.-J., 2014. Approaches to solve the fleet-assignment, aircraft-routing, crew-pairing and crew-rostering problems of a regional carrier. Omega 43, 71–82.

Sandhu, R., Klabjan, D., 2007. Integrated airline fleeting and crew-pairing decisions. Oper. Res. 55 (3), 439–456.

Shafipour-Omrani, B., Komijan, A.R., Sadjadi, S.J., Khalili-Damghani, K., Ghezavati, V., 2021. A flexible mathematical model for crew pairing optimization to generate n-day pairings considering the risk of COVID-19: a real case study. Kybernetes.

Tahir, A., Desaulniers, G., El Hallaoui, I., 2022. Integral column generation for set partitioning problems with side constraints. INFORMS J. Comput..

Terashima-Marín, H., Flores-Alvarez, E., Ross, P., 2005. Hyper-heuristics and classifier systems for solving 2D-regular cutting stock problems. In: Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation. ACM, pp. 637–643.

Walker, J.D., Ochoa, G., Gendreau, M., Burke, E.K., 2012. Vehicle routing and adaptive iterated local search within the hyflex hyper-heuristic framework. In: Learning and Intelligent Optimization. Springer, pp. 265–276.

Wen, X., Sun, X., Sun, Y., Yue, X., 2021. Airline crew scheduling: Models and algorithms. Transp. Res. E 149, 102304.

Yaakoubi, Y., Soumis, F., Lacoste-Julien, S., 2020. Machine learning in airline crew pairing to construct initial clusters for dynamic constraint aggregation. EURO J. Transp. Logist. 9 (4), 100020.

Zeighami, V., Saddoune, M., Soumis, F., 2020. Alternating Lagrangian decomposition for integrated airline crew scheduling problem. European J. Oper. Res. 287 (1), 211–224. http://dx.doi.org/10.1016/j.ejor.2020.05.005, URL https://www.sciencedirect.com/science/article/pii/S0377221720304161.

Zeighami, V., Soumis, F., 2019. Combining benders' decomposition and column generation for integrated crew pairing and personalized crew assignment problems. Transp. Sci. 53 (5), 1479–1499.

Zeren, B., Özkol, I., 2012. An improved genetic algorithm for crew pairing optimization. J. Intell. Learn. Syst. Appl. 4, 70–80. http://dx.doi.org/10.4236/jilsa.2012.41007.

Zeren, B., Özkol, I., 2016. A novel column generation strategy for large scale airline crew pairing problems. Expert Syst. Appl. 55, 133–144.