Combining Constraint Programming and Metaheuristics for Aircraft Maintenance Routing with a Distribution Objective

Ida Gjergji[1](✉)[0000−0002−2970−1535], Lucas Kletzander[2][0000−0002−2100−7733], Hendrik Bierlee[3,4,5][0000−0001−6766−5435], Nysret Musliu[2][0000−0002−3992−8637], and Peter J. Stuckey[3,4][0000−0003−2186−0459]

[1] DBAI, TU Wien, Vienna, Austria
[2] Christian Doppler Laboratory for Artificial Intelligence and Optimization for Planning and Scheduling, DBAI, TU Wien, Vienna, Austria
{ida.gjergji,lucas.kletzander,nysret.musliu}@tuwien.ac.at
[3] Monash University, Melbourne, Australia
{hendrik.bierlee,peter.stuckey}@monash.edu
[4] OPTIMA ARC Industry Training and Transformation Centre, Melbourne, Australia
[5] KU Leuven, Belgium henk.bierlee@kuleuven.be

Abstract. In this paper we focus on a challenging version of the aircraft maintenance routing problem (AMRP) with a maintenance distribution objective (AMRP-D). For the AMRP-D, the flight legs with predefined start and end times are assigned to aircraft. In addition to the assigned flight legs, each aircraft has to satisfy certain regulations regarding the maintenance services that are mandatory in the scheduling period, while the maintenance should also be distributed evenly. We propose a two stage approach, where first we use a decomposition method that is solved using constraint programming, to cover the flight legs. To optimize the distribution objective, we propose two metaheuristic techniques based on Large Neighborhood Search (LNS) and Simulated Annealing (SA). The LNS method consists of different destroy operators and as repairer we use a constraint programming (CP) solver. The SA approach includes a novel neighborhood to deal with the distribution objective. Our experimental results show that the decomposition method is able to solve more instances than the exact approach, while SA provides better quality solutions for the optimization stage compared to LNS.

Keywords: Constraint programming · Scheduling · Metaheuristics.

1 Introduction

Airlines operate in a challenging environment including strong regulations, variable demand, high competition, and high operational cost. Several optimization problems arise in this context, and high quality solutions are very important for

successful operation [3]. Due to the overall complexity, several different problems are identified and usually solved separately. Typically these are the flight scheduling problem (FSP, generating a timetable for flights), the fleet assignment problem (FAP, assigning aircraft types to flights), the aircraft maintenance routing problem (AMRP, assigning flights and maintenance to individual aircraft), and finally the crew scheduling problem (CSP, assigning various types of crew). All these sub-problems are known to be NP-hard [16], and even small improvements in solutions can transfer to significant savings for airlines [15].

We focus on the AMRP, where flight legs and the type of aircraft to fly these are already fixed, but the specific aircraft still need to be assigned. Since aircraft need a range of different maintenance operations that depend on the sequence of flights, passed time and flight hours, and the current location, the assignment of flight legs, and maintenance operations is typically combined. We refer to reviews [15, 16] for an overview of work in this area. Problem formulations date back many years [4]. Solution methods include heuristics [6], network-based MIP formulations [7, 3], and branch-and-price techniques [13]. Heuristic approaches often outperform exact approaches on instances of realistic scale [1, 2].

Capacity constraints were rarely considered, or only with upper limits [3]. Eltoukhy et al. [3] highlight in their overview regarding future research directions, that the AMRP rarely includes more considerations on how the maintenance workers are utilized. However, this can have very negative effects on the workforce, leading to both very high peak workloads, creating stress and fatigue, or requiring high overall manpower, while at other times excess capacity leads to increased cost. This paper investigates the extended problem that we recently introduced [9], with an additional objective to optimize the distribution of maintenance work, called the AMRP-D (AMRP with maintenance distribution). The additional objective greatly helps to provide a well distributed workload for the maintenance department, however, it also makes it even more challenging to provide good solutions for the problem, which motivates this research. To the best of our knowledge, no solution methods have been proposed in the literature yet for this extended problem. The main contributions of this paper are:

- A solver-independent model with different objective functions to optimize the maintenance distribution, including a novel over-approximation model.
- A decomposition approach based on the CP model to obtain feasible solutions for instances of realistic size.
- A Large Neighborhood Search that includes various novel destroy operators and a CP solver as a repair operator.
- A Simulated Annealing approach composed of three neighborhoods, including a novel neighborhood heuristic for maintenance distribution.
- Critical evaluation and comparison of the proposed methods on the diverse set of instances.

The experimental evaluation shows that while the CP based approach is crucial for obtaining feasible solutions, combining it with LNS and SA allows for significant improvements in results for most instances regarding the distribution.

2 Problem Description

The Aircraft Maintenance Routing Problem (AMRP) is specified by a set of flight legs, regulations for maintenance, and a set of aircraft including their recent flight and maintenance history. All times are given as integers.

2.1 Flight Legs

A set of n flight legs $T = \{t_1, \ldots, t_n\}$ is given, each leg t_i is associated with a start time s_i, an end time e_i, and a flight time f_i (with $f_i < e_i - s_i$, as the flight leg contains both the flight time and the preparation/turnaround times). As such, flight legs are not allowed to overlap for the same aircraft, but there is no minimum distance between legs on the same aircraft. The first m legs in T denote the last leg of each of the m aircraft from the previous scheduling period which are fixed to the particular aircraft. Note that in general, each flight leg is also associated with a start location and an end location, defining the routing aspect of the problem. However, in this paper we deal with a version using a single home base where outbound and following inbound flights are fused to one flight leg (with no maintenance at the destination). Therefore, all locations are equal and omitted.

2.2 Maintenance

Aircraft require different types of maintenance. Each maintenance type takes a different amount of time to complete. A maintenance on an aircraft cannot overlap with its other maintenance or flight legs. Some maintenance compete for a single hangar. Furthermore, some types are due periodically (regardless of flight time), while others are due based on the cumulative flight time. The following types of maintenance are used in this paper:

- Regular: An aircraft can fly for at most $regl = 47$ hours after the end of the previous regular maintenance, then it needs regular maintenance taking $regd = 2.5$ hours before taking off again.
- Weekly: An aircraft can fly for at most $weekl = 156$ hours (6.5 days) after the end of the previous weekly maintenance, then it needs weekly maintenance taking $weekd = 7$ hours before taking off again. Weekly maintenance includes regular maintenance.
- Major: There are four different types of major maintenance. Each of them is independent from the others. Each follows the same rule regarding time: After at most $majl = 950$ hours of cumulative flight time since the last maintenance of the same type, the aircraft needs major maintenance taking $majd = 14$ hours before taking off again. Each type of major maintenance includes regular and weekly maintenance. There are further differences regarding subtypes:
  - `MH1` and `MH2`: These require a single hangar, meaning that only one aircraft can perform any of these two types of maintenance at once.
  - `MR1` and `MR2`: These two types do not require the hangar.

2.3 Aircraft

A set of m aircraft $A = \{a_1, \ldots, a_m\}$ is given, each aircraft a_j is preassigned to the history flight leg t_j, and is associated with the time since the start of the planning horizon that the last regular maintenance ended r_j, the time since the start of the planning horizon that the last weekly maintenance ended w_j, and the cumulative flight time since each of the major maintenance types (including the last assigned leg) $majp_{kj}$ with $k \in \{1, 2, 3, 4\}$, where 1 and 2 correspond to `MH1` and `MH2`, while 3 and 4 correspond to `MR1` and `MR2`.

2.4 Solution

The legs for $j \in \{1, \ldots, m\}$ are fixed to the corresponding aircraft. A feasible solution assigns all remaining flight legs to aircraft, and the required types of maintenance to specific aircraft and time intervals, such that:

- No overlapping flight legs are assigned to any aircraft.
- No maintenance intervals are violated.
- At most one aircraft is assigned `MH1` or `MH2` at any point in time.

Traditionally, the AMRP features objectives like minimizing the total maintenance duration, or minimizing the earliness of each maintenance compared to its latest possible assignment. In contrast, the version with distribution objective (AMRP-D) includes a focus on even distribution of maintenance. This requirement stems from a real-world application, where the issue of uneven maintenance distribution leads to difficult assignments of maintenance personnel: Peaks need to be covered, requiring a large workforce at the same time, while in times with few maintenance tasks costly overstaffing is caused. Further, there might also be other resources like required machinery that get scarce during peaks.

For the optimization goal of AMRP-D, the total number of aircraft in any type of maintenance m_s is calculated for each minute s in the planning period, and $\sum_s m_s^2$ is minimized. This optimizes the distribution of maintenance tasks (since peaks are penalized more), and also the total amount of maintenance (leading to maximization of the available intervals between maintenance tasks).

Major maintenance of each type, however, typically occurs at most once per scheduling period (a scheduling period is up to a month, while the maximum flight time is well over a month of non-stop flying). Therefore, we limit each type of major maintenance to at most once per aircraft and type. In addition, it should be as late as possible within the allowed window. The earliness in minutes is added to the objective.

3 Constraint Model

The standalone model for the AMRP-D uses the following variables:

- T': the set of flight legs excluding the previously assigned legs $T' = T \setminus \{t_j \mid j \in A\}$.

$$as_{jj} = s_j, \;\; j \in A \tag{1}$$

$$\texttt{alternative}(s_i, e_i - s_i, \overline{as}_i, [e_i - s_i \mid j \in A]), \;\; i \in T' \tag{2}$$

$$\texttt{disj}([majs_{kj} \mid k \in 1..2, j \in A], [majd \mid k \in 1..2, j \in A]) \tag{3}$$

$$majp_{kj} + \sum_{i \in T',\ as_{ij} \neq \circ,\ as_{ij} < majs_{kj}} f_i \leq majl, \;\; k \in 1..4, j \in A \tag{4}$$

$$\texttt{strictly\_increasing}([ms_{lj} \mid l \in 1..mm]), \;\; j \in A \tag{5}$$

$$ms_{lj} = \circ \rightarrow ms_{l+1,j} = \circ, \;\; j \in A, 1 \leq l < mm \tag{6}$$

$$ms_{1j} = w_j - weekd \wedge mt_{1j} = 5, \;\; j \in A \tag{7}$$

$$ms_{2j} = r_j - regd \wedge mt_{2j} = 6, \;\; j \in A, r_j \neq w_j \tag{8}$$

$$\texttt{element}(mt_{lj}, [majd, majd, majd, majd, weekd, regd], md_{lj}), \;\; l \in 1..mm, j \in A \tag{9}$$

$$\texttt{disj}(\overline{as}_j +\!\!+ \overline{ms}_j, [e_i - s_i | i \in T] +\!\!+ \overline{md}_j), \;\; j \in A \tag{10}$$

$$as_{ij} \neq \circ \rightarrow (\exists_{l \in 1..mm} ms_{lj} \neq \circ \wedge ms_{lj} < as_{ij} \wedge$$
$$ms_{lj} + md_{lj} + regl \geq e_i), \;\; i \in T', j \in A \tag{11}$$

$$as_{ij} \neq \circ \rightarrow (\exists_{l \in 1..mm} ms_{lj} \neq \circ \wedge mt_{lj} \neq 6 \wedge$$
$$ms_{lj} < as_{ij} \wedge ms_{lj} + md_{lj} + weekl \geq e_i), \;\; i \in T', j \in A \tag{12}$$

$$majs_{kj} \neq \circ \rightarrow \exists_{l \in 1..mm} ms_{lj} = majs_{kj} \wedge mt_{lj} = k, \;\; k \in 1..4, j \in A \tag{13}$$

$$mt_{lj} \in 1..4 \rightarrow majs_{(mt_{lj})j} = ms_{lj}, \;\; l \in 1..mm, j \in A \tag{14}$$

$$m_s = \sum_{l \in 1..mm, j \in A} (ms_{lj} \neq \circ \wedge ms_{lj} \leq s \wedge ms_{lj} + md_{lj} > s), \;\; s \in 0..\max_{i \in T}(e_i) \tag{15}$$

Fig. 1: Model of AMRP-D. `disj` shorthand for `disjunctive` for layout reasons.

- as_{ij}: (optional) start time if flight leg i is assigned to aircraft j.
- $majs_{kj}$: (optional) start time for major maintenance type k for aircraft j.
- mm: The maximum number of maintenance tasks for each aircraft in schedule. Setting this value too high has negative effect on the performance as maintenance calculation is very challenging. Setting it too low leads to restrictions in the search space, cutting off optimal or even all feasible solutions. Based on the most frequent type of maintenance (regular), we use Equation (16) for the definition, where $start_h$ is the earliest timeslot referenced in the history, and the addition of 5 was determined empirically based on a large set of possible sub-problems.

$$mm = \frac{\max_i(e_i) - start_h}{regl} + 5 \tag{16}$$

- ms_{lj}: (optional) start time for l^{th} maintenance task for aircraft j.
- md_{lj}: the duration of the l^{th} maintenance task for aircraft j.
- mt_{lj}: type of l^{th} maintenance on aircraft j in {1..6} where {1..4} are major, 5 is weekly, 6 is regular.

We use notation $s = \circ$ for an optional start time variable s to mean it is *absent*, that is does not occur. Constraints involving absent variables are considered to always hold.

The mathematical model is shown in Figure 1. Equation (1) ensures that aircraft j is assigned their last assigned flight leg starting at the correct time. We use $\overline{as}_i$ to represent the i^{th} row of matrix as, similarly $\overline{as}_j$ represents the j^{th} column, and similarly for other matrices. Equation (2) enforces that each leg is assigned to some aircraft, the `alternative` constraint enforces that exactly one of the optional start times for that leg equals the given start time. Equation (3) enforces that only one major maintenance task requiring the hangar is executing at any time; the optional task `disjunctive` enforces this, ignoring absent tasks. Equation (4) enforces that the total flight time of legs performed by aircraft j before a major maintenance of type k fits within the limit $majl$; note that if $majs_{kj} = \circ$ we assume the inequality $as_{ij} < majs_{kj}$ always holds. Equation (5) enforces that the start times for the list of maintenance tasks for each aircraft are strictly increasing; note absent tasks always satisfy this constraint. Equation (6) ensures that absent maintenance tasks are at the end of the list. Equation (7) makes the first maintenance task for aircraft j a weekly maintenance task ending at time w_j; note that this happens before all tasks to be scheduled. Similarly if $r_j \neq w_j$, Equation (8) makes the second maintenance task for aircraft j a regular maintenance task ending at time r_j. Equation (9) enforces that the duration of the l-th maintenance task for aircraft j matches the type of maintenance; the $\texttt{element}(i, a, x)$ constraint enforces that $x = a[i]$. Equation (10) enforces that there is no overlap in the (optional) flight and maintenance tasks for each aircraft j; here $+\!\!+$ represents sequence concatenation. Equation (11) enforces that there is a maintenance task before each leg i flown by aircraft j so that the end of the flight leg is within the regular maintenance limit. Similarly Equation (12) enforces that there is a (non-regular) maintenance task before each leg i flown by aircraft j so that the end is within the weekly limit. Equation (13) enforces that for each major maintenance task of type k that is actually scheduled for aircraft j, that it occurs in the list of maintenance tasks with the same start time. Equation (14) enforces that if a major maintenance type occurs in the list of maintenance tasks, then that major maintenance is scheduled.

3.1 Optimization Objectives

The objective function is to minimize the sum of squares of the number of simultaneous maintenance tasks occurring at each minute s, m_s. The calculation of m_s is given by Equation (15), leaving the basic objective as:

$$\text{minimize} \sum_{s \in 0..\max_i(e_i)} m_s^2 \tag{17}$$

There is a secondary requirement, to make the major maintenance happen as late as possible which is achieved by an additional objective term:

$$\sum_{k\in 1..4, j\in A} majl - \left(majp_{kj} + \sum_{i\in T', as_{ij}\neq\circ, as_{ij}<majs_{kj}} f_i \right) \tag{18}$$

Note that the expression calculates the remaining flight time allowed for each aircraft and major maintenance type (the same term is used in Equation (4) and the model defines the expression only once).

3.2 Modeling the Objective Function

Using a sum of squares over all timeslots in the planning horizon is not a feasible approach in CP, as the number of timeslots is too high (flights are in granularity of 5 minutes, the horizon is up to a month), and the squaring is not efficient for most CP solvers either. Equation (18) is easy to handle and is included in all options described here.

$$\text{minimize} \sum_{l\in 1..mm, j\in A, ms_{lj}\neq\circ} md_{lj} \tag{19}$$

Equation (19) shows a simple objective to minimize the total duration of the maintenance tasks. This is easy to model, however, it does not directly help with the distribution objective. It can be combined with a `cumulative` constraint to minimize the maximum number of concurrent maintenance tasks as well. This approach has been conducted in our preliminary work [9], however, the distribution objective was only poorly optimized with this version.

A more efficient way to capture the true objective function is to introduce an additional array $maint$ of length $2 \cdot mm \cdot m + 1$ which contains the sorted array of all maintenance start and end times of all aircraft (plus a dummy at the end). This can be achieved with the `sort` constraint, but is more efficient in a custom mapping that is skipped here for brevity.

$$act_s = \sum_{l\in 1..mm,\ j\in A} ms_{lj} \neq \circ \ \wedge\ ms_{lj} \leq maint_s \wedge\ ms_{lj} + md_{lj} > maint_s \tag{20}$$

$$\text{minimize} \sum_{s\in 1..2\cdot mm\cdot m} act_s \cdot act_s \cdot (maint_{s+1} - maint_s) \tag{21}$$

Equation (20) defines the number of active maintenance tasks at point $maint_s$ in time (the sum counts the number of times the expression evaluates to true), while Equation (21) only sums at all points where a maintenance starts and ends, instead of all timeslots.

However, Equation (21) is still costly to evaluate, therefore, we propose a new overestimation of the objective that is much less computationally involved, but optimizes in a similar way to the exact objective.

$$appr_{lj} = \sum_{x \in 1..mm,\ y \in A} ms_{lj} \neq \circ \ \wedge \ \text{overlap}(ms_{lj}, md_{lj}, ms_{xy}, md_{xy}) \tag{22}$$

$$\text{minimize} \sum_{l \in 1..mm, j \in A} appr_{lj} \cdot md_{lj} \tag{23}$$

Equation (22) approximates the number of active maintenance during maintenance lj by all maintenance (including itself) that overlap in any way with this one. In reality, they will not overlap for the whole duration, which leads to an over-approximation. In Equation (23) the approximate number is assumed for the whole duration. For each maintenance, each overlapping one contributes to the overall sum, therefore the square is not needed in the sum itself.

4 Obtaining an Initial Feasible Solution

An initial solution is required for local search procedures. One of the challenges of this problem is the attainment of a *feasible* initial solution. As AMRP is a *hard to satisfy* problem, getting an initial feasible solution with a greedy heuristic is often not possible due to its complexity. Therefore, in the first part of this work, we present a decomposition approach with backtracking to provide initial feasible solutions. The goal is to solve smaller (and therefore faster) subproblems when possible, while iteratively increasing the size when needed.

The planning horizon is divided into D slices. To allocate the legs to slices, we first identify the leg with the earliest start time and then assign this leg and all overlapping unassigned legs to a slice. Another strategy we explore is the use of 2 legs with the earliest start time to define the legs per slice. The total number of slices D varies from instance to instance. Each leg can belong to only one slice. Then, in an iteration we consider the legs of slice $d \in \{1, 2, \ldots, D\}$ and all the aircraft, sequentially moving through the slices in the planning horizon. At slice d we consider the legs of that slice whilst the legs from the slice 1 to slice $d-1$ are fixed to their previously assigned aircraft. Maintenance tasks from previous slices are not fixed, they are re-scheduled at each slice. If a feasible solution at slice d is found, we proceed to the subsequent slice, otherwise we *backtrack*. With this method, the aircraft are always in a feasible status, however it can happen that legs can not be assigned to the aircraft. One reason for this is that in between flights the aircraft need to be maintained (regular, weekly or major maintenance), so a rearrangement of the legs is needed before moving on to the next slice. If backtracking occurs at slice d, then the flights belonging to slice *d-1* need to be reassigned. More slices are iteratively unfixed until a feasible assignment of legs and maintenance tasks for all aircraft can be scheduled according to the regulations up to the current slice. For both approaches, the model is solved to find feasible solution (no objective function). After an initial solution has been reached, the procedure optimizes the maintenance for each aircraft individually using Equation (23) (as the initial feasible solution can contain redundant maintenance in the absence of an objective function).

5 Large Neighborhood Search for AMRP-D

Large Neighborhood Search (LNS) is a local search algorithm proposed by Shaw [14]. LNS starts with an initial solution that commonly is reached with a greedy heuristic. With the principle of *destroy* and *repair*, LNS selects a portion of the solution using the destroy operator/s and removes fully or partially the assignments in this portion. The other part of the solution remains fixed. Then, using repair operator/s the selected portion of the solution is restored. Multiple destroy and repair operators can be included in the LNS. If the new solution improves the current one, the solution is updated, otherwise the next iteration of the algorithm starts from the old solution. This procedure is carried out repeatedly until the stopping criterion is met.

5.1 Destroy Operators

By using the destroy operators, a part of the solution at any iteration is selected. While the rest of the solution remains intact, this part of the solution is destroyed which means that those existing assignments are removed. The design of the destroy operators is crucial for achieving good results with LNS. Typically, the destroyers should explore the structural properties of the problem.

After a careful analysis of the AMRP-D, we propose five different destroy operators, which are utilized in the LNS based on their respective weight using the *roulette wheel approach* [5]. Each destroy operator is used only on some consecutive days of the scheduling period, determined by a parameter d, in order to maintain a reasonable runtime per iteration. The other assigned flight legs and maintenance tasks for the aircraft present in the sub-problem are considered as *history* (or future) assignment. The parameter k is used to identify up to k aircraft of interest based on the respective operator. For each selected aircraft, all legs in the selected time window are part of the sub-problem. Considering a *destroy size* of k aircraft and a duration of d days (illustrated in Table 1), the following operators are used:

- `Non-overlapping Legs (NLegs)`: For a randomly selected flight leg t, identify $k-1$ flight legs that do not overlap with t which are closest to the start/end time of t, each aircraft covering any of these legs is part of the sub-problem. The period included in the sub-problem starts at earliest start time of these legs.
- `Overlapping Legs (OLegs)`: For a randomly selected flight leg t, select $k-1$ overlapping flight legs, and add the aircraft these are assigned to. The period included in the sub-problem starts at the earliest start time of these legs.
- `Peak Aircraft (PAir)`: Peak is defined to be the time in the planning period when the highest number of maintenance tasks occurs simultaneously and peak aircraft are the aircraft being maintained in the peak time. Then in the sub-problem an aircraft in the peak time and $k-1$ aircraft not in the peak time are selected. The start time for the sub-problem begins one day earlier than the peak time.

- `Size Aircraft (SAir)`: Select the aircraft with the maximum number of flight legs and the minimum number of flight legs assigned. The start time is randomly selected within the scheduling period.
- `Random Aircraft (RAir)`: For a randomly selected aircraft, select one of its maintenance start times (either regular or weekly maintenance). For the picked maintenance task, select $k-1$ flight legs that overlap with it, add the initial aircraft and those having one of the selected legs in their assignment. The start time of this operator is the minimum start time among these legs and the maintenance.

5.2 Repair Operator

In the repair phase, the focus is on the sub-problem which is formed based on the destroy operators. Some additional data regarding the aircraft not present in the subproblem is needed. Specifically, the start time and duration of the maintenance tasks of other aircraft is passed each iteration to the CP model in order to use the *overestimation* the objective function based on Equation (23). Using the exact objective function is not time efficient even for the sub-problem. Further, the next fixed leg, and the latest possible times for maintenance are specified for each aircraft similar to the history assignments.

5.3 Parameters

The parameters used in the LNS are described in Table 1. In the first column, the destroy operators are listed. The number of aircraft and the number of days per operator, given in column two and three, are determined based on preliminary experiments to keep the runtime per iteration moderate while maximizing the effect of the operator. Probabilities p_1, p_2, p_3, p_4 represent the selection probabilities for the destroy operators `NLegs`, `OLegs`, `PAir`, and `SAir` respectively. Then, the `RAir` destroy operator is used with $p_5 = 1 - (p_1 + p_2 + p_3 + p_4)$. The selection probabilities are determined with the automatic parameter configuration tool `irace` [10]. For this setup, we use 17 instances with a budget of 10000 experiments, where each experiment has a timeout of 600 seconds. The domains given to the selection probabilities are given in the fourth column of Table 1 while in the fifth column we display the chosen values from `irace`. Lastly, the repair time given to the CP solver is set with manual trials to $t_r = 200$ seconds.

6 Simulated Annealing for AMRP-D

As a second improvement method we propose to use Simulated Annealing [8]. It starts from a given initial solution, with an initial temperature $t = T_0$, and randomly selects moves with given probabilities. The new solution is accepted if it is better than before, or with probability $e^{-\Delta/t}$, where Δ is the change in solution value, and t is the current temperature. A number of inner iterations *inner* is performed on the same temperature, then the temperature is reduced

Table 1: Parameter values used for the LNS

Operator	Nr aircraft	Nr days	Domain	Tuned
`NLegs`	3	5	$p_1 \in [0, 0.5]$	0.50
`OLegs`	5	3	$p_2 \in [0, 0.5]$	0.22
`PAir`	3	5	$p_3 \in [0, 0.5]$	0.06
`SAir`	2	6	$p_4 \in [0, 0.5]$	0.18
`RAir`	3	5	$p_5 \in [0, 0.5]$	0.04

by $t' = c \cdot t$, where c is the cooling rate. Hard constraint violations can occur and are treated with high penalties (missing maintenance or overlap of flight legs).

Three different types of moves are used in our Simulated Annealing. The first two deal with the assignment of flight legs and either reassign one flight leg (`move_leg`), or a sequence of consecutive legs (`move_legs`, the length of the sequence is randomly chosen), from one aircraft a_1 to another a_2. All legs overlapping with the moved legs in a_2 are moved back to a_1. The third move `recalc_maintenance`, which is a novel addition for the AMRP-D, recalculates all maintenance for a randomly selected aircraft with the procedure introduced in the following, while the assignment of legs is fixed. This allows us to reoptimize the maintenance distribution.

6.1 Maintenance Optimization

Maintenance optimization for aircraft a is performed in a sequential pass in the order of time. The last possible maintenance location (denoted by the gap between legs t_x and t_y consecutively assigned to aircraft a) for each type of maintenance is kept in memory. For each flight leg t_j, first for each type of maintenance the last possible location is updated to (t_i, t_j) (where t_i is the predecessor of t_j on a) if the location is suitable. If adding t_j would trigger a maintenance violation, the last possible location is chosen for the maintenance. The assignment might be infeasible if no possible location was found before t_j. This part of the assignment procedure ensures that maintenance is not done excessively early, but just when needed.

The second part of the assignment procedure picks the best spot between the consecutive legs t_i and t_j where the maintenance is assigned. In general, the interval $[e_i, s_j]$ is available for the assignment, but the beginning might be restricted further to $e_i + x$ if necessary for the feasible assignment of the currently evaluated leg. Within this interval, the maintenance is added at the latest possible spot where the maintenance distribution objective is minimal. For efficiency, the evaluation is limited to all points in time in the interval where the overall maintenance assignment of the problem changes.

This concept was also used in a greedy heuristic, which sorts the flight legs by start time, and then assigns each leg to the aircraft with the least increase in the objective, using the above procedure for maintenance assignment. However, this approach was not successful in avoiding conflicts.

Table 2: Parameter values used for the SA

Parameter	Values	Default	Tuned
T_0	$\{1, 10, 100, 1000, 10000\}$	100	100
c	$\{0.9, 0.95, 0.99, 0.995, 0.999, 0.9995, 0.9999\}$	0.999	0.9999
p_1	$(0; 1)$	0.05	0.22
p_2	$(0; 1)$	0.475	0.52

6.2 Parameters

The parameters used for SA are described in Table 2. The two parameters T_0 and c for the cooling scheme, as well as the probabilities p_1 for `move_leg` and p_2 for `move_legs` are tuned. `recalc_maintenance` is chosen with $p_3 = 1-(p_1+p_2)$. The number of inner iterations $inner = |T|$ is fixed (with a lower bound of 100, and an upper bound of 1000) to allow scaling for the instance size, but using more inner iterations with faster cooling would be similar to less inner iterations with slower cooling, therefore, only one of the parameters is tuned. Simulated Annealing is implemented with efficient delta evaluation, only reevaluating parts of aircraft schedules when needed. Additionally, there is an early stopping criterion: if there is no improvement in 100 temperature steps, the algorithm stops, since further improvements are very unlikely. The parameters are again tuned with `irace` [10] on the same 17 instances as for LNS, but with a budget of 1000 experiments with the full timeout of 1 hour (the cooling scheme needs the full runtime to be properly tuned).

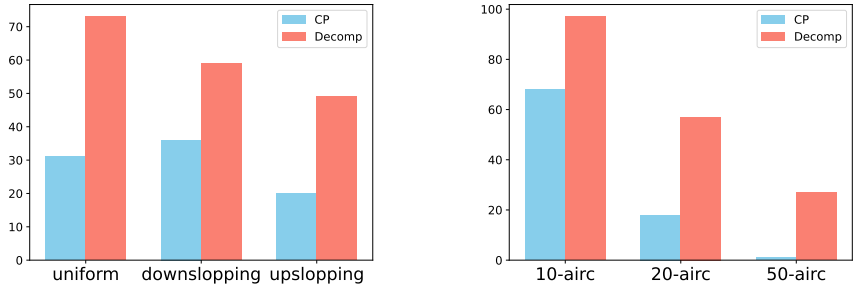
7 Experimental Evaluation

The experiments were done on a computing cluster, equipped with two Intel Xeon E5-2650v4 @ 2.20 CPUs with 12 cores. CP-SAT 9.10 [12] is used for solving all versions of the CP model described in Section 3 which was modeled in MiniZinc [11] and PyPy 7.3.11 for running the other code written in Python. The runs of the decomposition method were executed in 8 threads using a time budget of 12000 seconds. These runs are deterministic and performed only once. The runs of LNS and of SA were executed in a single thread using a time budget of 3600 seconds. As the LNS and the SA are not deterministic, we perform 10 runs per instance for each method. The seeds are stored for reproducibility.

The available benchmark instances are created to resemble structures of real-life instances from a practical use case (single hub, mid- to long-distance flights, with seasonal fluctuations and daily departure peaks). The instance set combines four different features as shown in Table 3, leading to 324 instances. More dense instances include more flight legs for the same number of aircraft, making it harder to even find feasible solutions. d4-d9 represents demand that reduces during the scheduling period (downsloping), while u4-u9 represents upsloping demand. These combinations result in 70 to 1695 flight legs per instance.

Table 3: Instance characteristics

Number of aircraft	10, 20, 50
Number of days	7, 14, 28
Daily departure peaks	0 (no) or 1 (yes)
Density	5-10, d4-d9, u4-u9



(a) Instance structure

(b) Instance size

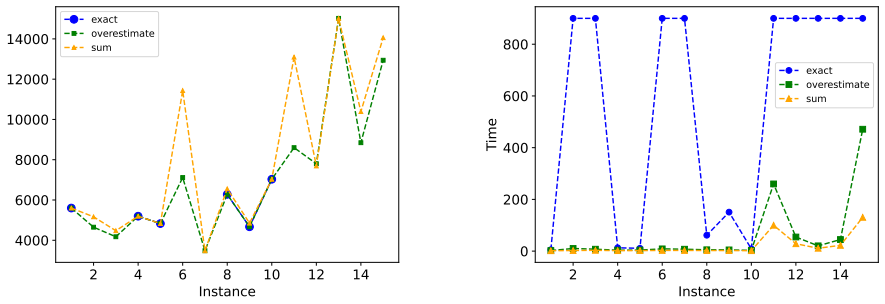
Fig. 2: Number of solved instances based on instance structure and size

7.1 Feasibility Results

Figure 2 shows a comparison for the number of feasible solutions, solving only for satisfaction, using the model from Section 3 (CP) and the decomposition approach from Section 4 (Decomp) that uses the same model at each slice d. Note that the greedy approach mentioned in Section 6 could only provide 7 feasible solutions for instances of very low density and is not shown here. CP mostly covers small sized instances with a total of 87 feasible instances, while the decomposition approach with two strategies provides 181 feasible instances. There is only one instance solved only by CP and not the decomposition method. CP struggles with instances that have upsloping demand (Figure 2a) and with the medium to large sized instances (Figure 2b). The decomposition technique handles instances of diverse structures well, but as the size of the instances increases, fewer feasible solutions are obtained.

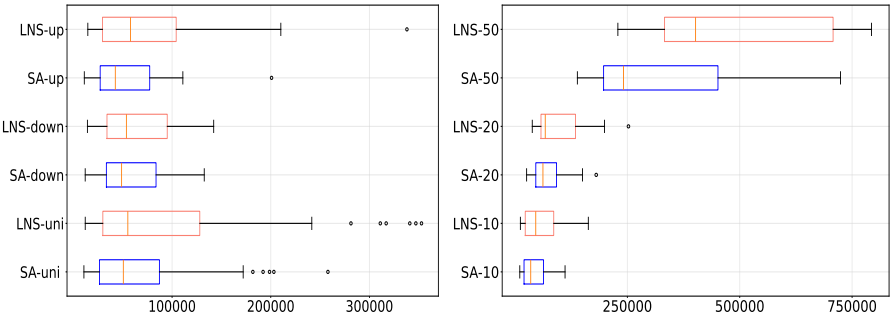
7.2 Reformulating the Objective in MiniZinc

In order to evaluate the proposed objective function formulations we perform experiments using a timeout of 900 seconds on LNS sub-problems. In Figure 3a the exact cost for 15 different sub-problems obtained from the exact Equation (21), overestimate Equation (23) and sum approaches Equation (19) is shown whilst in Figure 3b the respective time of each approach for the same sub-problems is displayed. Note that in Figure 3a the missing values for the exact approach indicate that no solution was returned in this timeout, which happens for 9 out of 15



(a) Cost from different formulations (b) Time from different formulations

Fig. 3: Comparing different formulations of objective function



(a) Average values boxplot (structure) (b) Average values boxplot (size)

Fig. 4: Boxplots for the average values obtained from SA and LNS

sub-problems. This indicates that the exact approach might not be adequate for the iterative procedure of LNS. For the sub-problems where the exact method returns a solution, the overestimate approach and the sum approach match their cost. Nevertheless, their runtime is lower than the one of the exact method. Comparing the overestimate and the sum approach, the sum approach runs faster, but it provides lower quality solutions. Therefore, we include the overestimate approach in the MiniZinc model used for the decomposition approach to remove redundant maintenance tasks, and for the LNS procedure.

7.3 Optimization Results

In this part we compare the results from SA and LNS, when both algorithms use the initial solutions obtained from the decomposition approach.

In Figure 4, we present the boxplots for the average values from SA and LNS based on the instance structure (Figure 4a) and based on the instance size (Figure 4b). Some outliers are outside the visualized range, almost all for LNS.

For the minimum values achieved in 10 runs, LNS provides better results than SA for 14 instances, provides equal results with SA for 6 instances and SA offers the best minimum values for the remaining 161 instances. Regarding the average values over 10 runs, LNS provides better results than SA for 14 instances, provides same results with SA for 3 instances and for the other 164 instances SA delivers better average results. Most of the instances for which LNS provides better minimum or average values are instances with 10 aircraft and uniform density. As shown in Figure 4a, SA and LNS perform similarly for instances with downsloping density, while for instances with upsloping or uniform density their performance differences are more noticeable. In Figure 4b, the largest performance difference is for instances with 50 aircraft. The average improvement of SA and LNS compared to the objective returned by the decomposition approach for instances with 10 aircraft is -23.94% for SA and -13.89% for LNS; for instances with 20 aircraft -33.46% for SA and -14.64% for LNS; for instances with 50 aircraft -36.66% for SA and -7.20% for LNS. Moreover, LNS provides feasible results for all instances over all runs unlike SA that at times return infeasible solutions. This is due to the fact that SA explores also infeasible regions and might get stuck in such regions. When this happens the cost returned by SA is the cost of the initial solution. For LNS, subproblems of larger size might be needed for better results, however, these quickly become very time-consuming.

Additionally, we perform the Wilcoxon signed-rank test with the Python module `scipy`. The obtained p-value is 5.08×10^{-25}. With a significance value of $\alpha = 0.05$, these results show that SA and LNS perform statistically different.

For some smaller instances, we could find the optimum for the total maintenance duration (Equation (19)), and verified that optimizing the distribution does not increase this metric too much (average within 2.3 %), while the distribution is well optimized in our new results.

8 Conclusions

In this paper we investigated a specific version of the aircraft maintenance routing problem that focuses on the distribution of maintenance tasks. We provided a solver-independent model that can be used to solve small instances directly, as well as in our decomposition approach, which resulted in far more feasible solutions. We proposed an LNS method with new destroy operators and a CP solver in the repair stage. We analyzed different ways of formulating the objective function in the MiniZinc model to improve the efficiency of the repair stage, and showed that our over-approximation method can provide a large speed-up with only moderate deviations in the objective. Furthermore, we presented an SA method with three distinct neighborhoods. While both improvement methods can achieve large improvements in the distribution objective, Simulated Annealing provides the best results. Our methods contribute to a better integration of aircraft routing and scheduling of maintenance work. For future work we would like to investigate the branch-and-price technique for the AMRP-D and investigate multi-hub scenarios, where maintenance can be done in multiple locations.

Acknowledgments. The financial support by the Austrian Federal Ministry of Labour and Economy, the National Foundation for Research, Technology and Development and the Christian Doppler Research Association is gratefully acknowledged. This research was also funded by the Austrian Science Fund (FWF), project I 5443-N, Grant-DOI 10.55776 and by the Doctoral Program Vienna Graduate School on Computational Optimization, Austrian Science Foundation (FWF), under grant No.: W1260-N35 (https://vgsco.univie.ac.at/). The work of P.J. Stuckey and H. Bierlee is partially supported by the Australian Research Council (OPTIMA ITTC IC200100009).

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Başdere, M., Bilge, Ü.: Operational aircraft maintenance routing problem with remaining time consideration. European Journal of Operational Research **235**(1), 315–328 (2014)
2. Cui, R., Dong, X., Lin, Y.: Models for aircraft maintenance routing problem with consideration of remaining time and robustness. Computers & Industrial Engineering **137**, 106045 (2019)
3. Eltoukhy, A.E., Chan, F.T., Chung, S.H.: Airline schedule planning: a review and future directions. Industrial Management & Data Systems **117**(6), 1201–1243 (2017)
4. Feo, T.A., Bard, J.F.: Flight scheduling and maintenance base planning. Management Science **35**(12), 1415–1432 (1989)
5. Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley Longman Publishing Co., Inc., USA, 1st edn. (1989)
6. Gopalan, R., Talluri, K.T.: The aircraft maintenance routing problem. Operations research **46**(2), 260–271 (1998)
7. Haouari, M., Shao, S., Sherali, H.D.: A lifted compact formulation for the daily aircraft maintenance routing problem. Transportation Science **47**(4), 508–525 (2013)
8. Kirkpatrick, S., Gelatt Jr, C.D., Vecchi, M.P.: Optimization by simulated annealing. Science **220**(4598), 671–680 (1983)
9. Kletzander, L., Gjergji, I., Musliu, N.: Combining aircraft maintenance routing with a distribution objective. In: Proceedings of the 14th International Conference on the Practice and Theory of Automated Timetabling, PATAT 2024. pp. 325–328 (2024)
10. López-Ibáñez, M., Dubois-Lacoste, J., Pérez Cáceres, L., Birattari, M., Stützle, T.: The irace package: Iterated racing for automatic algorithm configuration. Operations Research Perspectives **3**, 43–58 (2016)
11. Nethercote, N., Stuckey, P.J., Becket, R., Brand, S., Duck, G.J., Tack, G.: Minizinc: Towards a standard CP modelling language. In: Bessiere, C. (ed.) Principles and Practice of Constraint Programming - CP 2007, 13th International Conference, CP 2007, Providence, RI, USA, September 23-27, 2007, Proceedings. Lecture Notes in Computer Science, vol. 4741, pp. 529–543. Springer (2007). https://doi.org/10.1007/978-3-540-74970-7_38, https://doi.org/10.1007/978-3-540-74970-7_38

12. Perron, L., Didier, F., Gay, S.: The CP-SAT-LP solver (invited talk). In: Yap, R.H.C. (ed.) 29th International Conference on Principles and Practice of Constraint Programming, CP 2023, August 27-31, 2023, Toronto, Canada. LIPIcs, vol. 280, pp. 3:1–3:2. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2023)
13. Sarac, A., Batta, R., Rump, C.M.: A branch-and-price approach for operational aircraft maintenance routing. European Journal of Operational Research **175**(3), 1850–1869 (2006)
14. Shaw, P.: Using constraint programming and local search methods to solve vehicle routing problems. In: International conference on principles and practice of constraint programming (CP-98). pp. 417–431. Springer (1998)
15. Temucin, T., Tuzkaya, G., Vayvay, O.: Aircraft maintenance routing problem–a literature survey. Promet-Traffic&Transportation **33**(4), 491–503 (2021)
16. Xu, Y., Wandelt, S., Sun, X.: Airline scheduling optimization: Literature review and a discussion of modeling methodologies. Intelligent Transportation Infrastructure (2023)