KU LEUVEN

FACULTY OF ECONOMICS AND BUSINESS

A tabu search heuristic for building aircraft maintenance personnel rosters

De Bruecker P, Van den Bergh J, Beliën J, Demeulemeester E.

KBI_1409

A Tabu Search heuristic for building aircraft maintenance personnel rosters

Philippe De Bruecker[1], Jorne Van den Bergh[2,3], Jeroen Beliën[2,3], Erik Demeulemeester[1]

[1] KU Leuven, Research Center for Operations Management, Leuven (Belgium), philippe.debruecker@kuleuven.be, erik.demeulemeester@kuleuven.be
[2] HUBrussel, Center for Informatics, Modeling and Simulation, Brussels (Belgium), jorne.vandenbergh@kuleuven.be, jeroen.belien@kuleuven.be
[3] Affiliated researcher KU Leuven, Research Center for Operations Management, Leuven (Belgium)

Abstract

This paper presents a fast heuristic approach to optimize staffing and scheduling of the workforce at an aircraft maintenance company. Two linked Tabu Search algorithms are proposed to minimize the labour costs and to maximize employee satisfaction. To enhance the performance of the algorithm, strategic oscillation and pattern creation are used. We illustrate the performance of the algorithms with a computational experiment based on real life data from Sabena Technics, a large aircraft maintenance company located at Brussels Airport in Belgium. We analyze the results in different scenarios and compare them to a mixed integer linear programming model. The experiments demonstrate that Tabu Search can deliver acceptable solutions faster than the integer programming technique for small instances and outperforms the integer programming model for larger and more realistic problem sizes.

Keywords: Tabu Search, staffing, scheduling, aircraft line maintenance, strategic oscillation, pattern creation

1 Introduction

This paper presents a fast heuristic algorithm to optimize the staffing and scheduling of aircraft maintenance. Staffing and scheduling problems impose a huge challenge for corporate planners, especially when the scale of the problem is very large. First, information is required on the demand for service. This information must contain the timing and the amount of workload that is needed to satisfy the demand. For most workforce planning problems, the required work at each period is known in advance. Aircraft maintenance on the other hand can take place at multiple time intervals between the STA (Scheduled Time of Arrival) and the STD (Scheduled Time of Departure) of a flight.

Our model assumes information of all flights during a certain cycle period, e.g., a week (see Table 1). Next to the time interval during which the maintenance should take place, the estimated workload in man-hours is given. In the example of Table 1, 310 flights must be maintained every week.

Table 1: Example of the provided input information

Flight	Company	STA	STD	Workload (man-hours)
1	BA	Monday 22:05	Tuesday 07:15	4
...	...	...	...	...
111	AA	Thursday 07:30	Thursday 10:40	6
...	...	...	...	...
310	SN33	Sunday 05:30	Sunday 08:45	4.25

Given this recurring workload for each flight during one week and its STA and STD, we have to find the optimal workforce configuration in terms of the lowest total labour costs. We first explain how a workforce configuration is defined.

We assume that the maintenance personnel works in teams of a certain team size. Each team can only work one shift per day or have a day off. There are four possible shift types with overlapping working hours: a morning shift, a day shift, an evening shift and a night shift. This results in a certain sequence of shift types and days off assigned to a certain team for a certain week. An example of such a sequence is shown in Figure 1 where N represents a Night shift, D a Day shift, M a Morning shift and E an Evening shift.

Figure 1: Example of a single sequence

mo	tu	we	th	fr	sa	su
N			D	E		

Figure 2: Example of two cycles

mo	tu	we	th	fr	sa	su
	N	N	N	N		

(a) Cycle 1

mo	tu	we	th	fr	sa	su
N			D	E		
M	M		D	N		E
	M	E		E		
	M	E		N	N	N

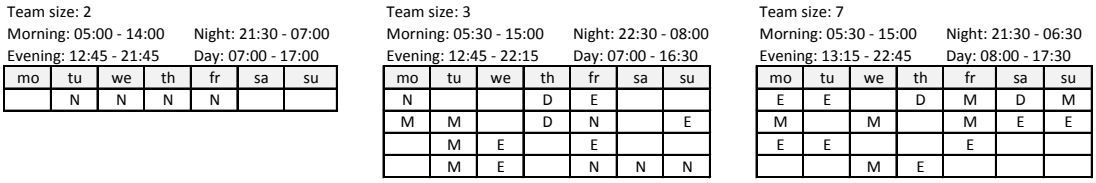
(b) Cycle 2

Because one cycle can contain multiple weeks, a team is not always assigned to the same sequence as the week before. Instead, a cyclic pattern is used in which the number of teams assigned to a cycle equals the number of weeks in the cycle. An example is presented in Figure 2(a) and 2(b).

In the first cycle, there is only one week. Therefore, there is only one team that is scheduled every week to work a night shift from Tuesday till Friday. In the second cycle, there are four teams scheduled. In the first week, the first team works a night shift on Monday, a day shift on Thursday and an evening shift on Friday. The second team works a morning shift on Monday and Tuesday and so on. In the second week, the first team will work the shifts in the second row of the cycle, the second team the shifts of the third row and so on. To complete the cycle, the fourth team will then work the shifts of the first row.

Finding a workforce configuration implies that we have to decide on the number of cycles, the number of weeks (= the number of teams) in each cycle, the composition of the shift sequences, the start and end times of every shift type in each cycle and the team size in each cycle. An example of such a workforce configuration with three cycles can be found in Figure 3, which shows an output example of the optimization algorithm.

Figure 3: Output example of the optimization algorithm



Team size: 2
Morning: 05:00 - 14:00 Night: 21:30 - 07:00
Evening: 12:45 - 21:45 Day: 07:00 - 17:00

mo	tu	we	th	fr	sa	su
	N	N	N	N		

Team size: 3
Morning: 05:30 - 15:00 Night: 22:30 - 08:00
Evening: 12:45 - 22:15 Day: 07:00 - 16:30

mo	tu	we	th	fr	sa	su
N			D	E		
M	M		D	N		E
	M	E		E		
	M	E		N	N	N

Team size: 7
Morning: 05:30 - 15:00 Night: 21:30 - 06:30
Evening: 13:15 - 22:45 Day: 08:00 - 17:30

mo	tu	we	th	fr	sa	su
E	E		D	M	D	M
M		M		M	E	E
E	E			E		
		M	E			

A feasible workforce configuration must cover the demand for services at any time. This is referred to as the coverage constraints. Real life staffing and scheduling problems are not limited to finding the lowest possible costs while satisfying the demand for the services. They must also follow strict union regulations. These regulations cover average working times, working in the weekend, break times, etc. They add extra constraints to the optimization problem. Both the coverage constraints and the union constraints can never be violated and are therefore called "hard" constraints. Furthermore, the employees might have specific preferences about the number and type of consecutive shifts. These preferences are translated into the optimization model as "soft" constraints. This means that as many as possible of these preferences should be satisfied, but it is not mandatory to satisfy all of them.

In the literature review in Section 2, we discuss the motivation and validity of this research. Section 3 presents the formal problem definition and Section 4 describes the methodology we used to solve the problem at Sabena Technics. Finally, the results are discussed in Section 5 followed by the conclusion in Section 6.

2 Literature review

Efficient employee scheduling has been the subject of many management science research papers. Each one has its own method to minimize overcapacity, costs, needed employees and has its unique set of constraints. Most research has been conducted on nurse scheduling at hospitals (Burke, De Causmaecker, Vanden Berghe & Van Landeghem, 2004). They deal with the planning of personnel working in shifts. The way of scheduling employees in different types of shifts (for example night, day, evening and morning shifts) is also a key ingredient for this research.

In a first effort to solve the problem at Sabena Technics, Beliën, Demeulemeester, De Bruecker, Van den Bergh and Cardoen (2013) proposed an iterative Mixed Integer Linear Programming (MILP) technique. MILP is a popular method that combines Integer Programming (IP) and Linear Programming (LP). The method describes a problem in a formal mathematical way which is then solved by specialized computer software. LP techniques have been used a lot in the past to solve employee scheduling problems (Ernst, Jiang, Krishnamoorthy & Sier, 2004). But even with very fast computers it

takes quite some time before one reaches the optimal solution. To overcome the large computation time, heuristic approaches are often used.

Metaheuristic algorithms such as Tabu Search, Genetic Algorithms and Simulated Annealing have been used in some cases to solve real life employee scheduling problems (Ernst et al., 2004). These techniques make it possible to overcome the large computation time of MILP models and deliver comparable results. It has even been proven that heuristic techniques often outperform techniques such as MILP for workforce scheduling problems of realistic dimensions (Beliën, 2006; Burke et al., 2004). Glover and McMillan (1986), for example, used Tabu Search to solve the general employee scheduling problem, a problem that strongly relates to the problem in this paper. Tabu Search (TS) is a local search combinatorial optimization technique developed by Glover and Laguna (1997). The technique has proven to deliver a performance and an adaptability that is superior to the ordinary LP techniques (El-Amin, Duffuaa & Abbas, 2000; Ernst et al., 2004).

The application of TS to solve this workforce problem presents a new way of incorporating scheduling and staffing for aircraft maintenance as we intend to solve the staffing (i.e., deciding on the team sizes) and scheduling problem (i.e., deciding on the timing of the shifts) at the same moment. This creates more flexibility and increases the possibility of finding better solutions.

3 Problem definition

Our model minimizes the labour costs of the maintenance workforce without neglecting the employee satisfaction. Hence, besides the cost minimization, maximizing employee satisfaction is a second goal. These two goals are defined as the objective functions of two separate optimization models. The only link between these models is that the second model (the satisfaction model) uses the output from the first model (the cost model) as its input. The formal problem definitions are presented below.

3.1 The cost model

Indices and sets:

$t \in T$:	shift types $\{M, D, E, N\}$
$c \in C$:	cycles in the schedule
$d \in \{1, 2, ..., 7\}$:	days in the week
$p \in P$:	time periods in one week

$p \in P_f$: time periods during which flight f can be serviced. I.e.; $P_f = \{p | STA_f \leq p \leq STD_f\}$ with STA_f (STD_f) the scheduled time of arrival (departure) of flight f

$f \in F$: flights to be serviced

Parameters:

L_f : the workload (in quarters of an hour) of flight f

W^u : the maximum number of weeks (=teams) in a cycle

H^l : the minimum duration of a shift (in quarters)

H^u : the maximum duration of a shift (in quarters)

H^{break} : the duration of a lunch break (in quarters)

Λ_t^l : the earliest start time (in quarters) of shift type t

Λ_t^u : the latest start time (in quarters) of shift type t

S : the minimum average number of working quarters per week

U : the maximum average number of working quarters per week

K_{td} : the cost for one worker to work one quarter when assigned to a shift of type t on day d

Decision variables (all of them have integer values):

$m_c \in \{2, 3, ...\}$: team size in cycle c

$h_{tc} \in \{H^l, H^l + 2, H^l + 4, ..., H^u\}$: duration of shifts of type t (in quarters) in cycle c

$\lambda_{tc} \in \{\Lambda_t^l, \Lambda_t^l + 2, \Lambda_t^l + 4, ..., \Lambda_t^u\}$: start time of shifts of type t (in quarters) in cycle c

$x_{tdc} \in \{0, 1, ..., W^u\}$: number of shifts of type t scheduled on day d in cycle c

Derived variables:

n_c : the number of weeks (= teams) in cycle c

W_c^l : the lower bound of the number of weeks (= teams) in cycle c

B_{tc} : the fraction of workers available to work when assigned to a shift of type t in cycle c (< 1 due to break/lunch time)

q_p : the available capacity in period p

$g_{fp} \geq 0$: the number of workers assigned to maintain flight f during time period p

$e^{+}_{tdc} \in \{0, 1, ..., W^u\}$: the number of extra weeks needed in cycle c for day d caused by shifts of type t (with $t \in \{E, N\}$) on the preceding day.

$e^{-}_{Ndc} \in \{0, 1, ..., W^u\}$: the number of extra weeks needed in cycle c for day d caused by E shifts that can be compensated by an excess in N shifts on the preceding day.

In contrast to the decision variables, the derived variables are not directly altered by the combinatorial optimization model (i.e., the TS algorithm) since their value depends on that of other decision variables. The value of derived variables can be directly calculated (without solving an optimization problem) and is mainly used to evaluate certain constraints during the optimization procedure.

The optimization model:

$$\text{Minimize:} \quad \sum_{c \in C} \sum_{t \in T} \sum_{d=1}^{7} K_{td} m_c h_{tc} x_{tdc} \tag{1}$$

The objective of the cost optimization model is to minimize the total labour costs during one week. We therefore multiply the unit cost of each shift of a certain type on a certain day with the team size of the cycle in which that shift can be planned. We then multiply this by the duration of that shift type in that particular cycle and multiply this again by the number of shifts that are actually planned. Note that the objective function (1) is not linear because it multiplies three different decision variables.

Subject to:

Coverage constraints:

$$q_p \geq \sum_{f \in F} g_{fp}, \qquad \forall p \in P \tag{2}$$

$$\sum_{p \in P_f} g_{fp} = L_f, \qquad \forall f \in F \tag{3}$$

$$q_p > 0, \qquad \forall p \in P \tag{4}$$

The first three constraints (2 to 4) are the coverage constraints. They ensure the ability to cover the demand for aircraft maintenance at every time interval during the planning horizon. Constraints (2) and (3) make sure that the demand for maintenance for each

flight can be met by the available capacity. This means that there should be sufficient workers available between the STA and STD of each flight to perform the required maintenance in time. The available capacity q_p in constraint (2) is determined by the scheduling of the shifts (x_{tdc}). Constraint (4) makes sure that there is at least one team available during each period in order to take care of clients with a disrupted schedule.

Algorithm 1 Derivation of q_p and g_{fp}

```
//Derivation of q_p
for all p ∈ P do
    q_p = 0;
end for
for (t = M,D,E,N) do
    for (d = 1,...,7) do
        for (c = 1,...,C) do
            for (p = λ_tc,...,λ_tc + h_tc) do
                q_p = q_p + m_c × x_tdc × B_tc;
            end for
        end for
    end for
end for

//Derivation of g_fp
Sort all fights f ∈ F by increasing STD_f;
for all p ∈ P do
    RemainingCapacity_p = q_p;
end for
for all f ∈ F do
    p = STA_f;
    g_fp = min {RemainingCapacity_p, L_f};
    RemainingCapacity_p = RemainingCapacity_p − L_f;
    for (p = STA_f + 1, ..., STD_f) do
        if (RemainingCapacity_{p−1} ≤ 0) then
            g_fp = min {RemainingCapacity_p, −RemainingCapacity_{p−1}};
            RemainingCapacity_p = RemainingCapacity_p + RemainingCapacity_{p−1};
            RemainingCapacity_{p−1} = 0;
        else
            g_fp = 0;
        end if
        if (p = STD_f) and (RemainingCapacity_p < 0) then
            RemainingCapacity_p = 0;
            //↪ Coverage constraint (3) will fail!
        end if
    end for
end for
```

Because we allow for a certain flexibility regarding the exact timing of the maintenance of a flight between its STA and STD, the timing of the workload is an extra decision incorporated by the variable g_{fp}. g_{fp} is the number of workers assigned to maintain flight f during time period p and depends on the assignment rule that is used. An assignment rule specifies how the available capacity should be assigned to each flight. As we assume maintenance intervals of 15 minutes, the entire planning horizon of one week is divided

into 672 smaller periods ($|P| = 672$). Hence, the assignment rule determines the amount of maintenance work assigned to each aircraft during each quarter between the STA and STD of the flight. We use a first leave - first serve assignment rule such that flights that leave first receive the highest priority and are maintained first. This assignment rule is the best way to avoid delays in departure times.

Algorithm 1 shows how q_p and g_{fp} are derived and how the first leave - first serve assignment rule is applied. First, the available capacity q_p is derived from the scheduling of the shifts (x_{tdc}), the team sizes (m_c) and the availability fractions (B_{tc}). The derived variable B_{tc} has a value between zero and one to indicate a decrease in available capacity which can be calculated from equation (5). This capacity reduction is necessary because workers want to have a lunch break during each of their shifts. We assume that each worker can take this break at different times, thus the capacity reduction is spread over the whole length of a shift. Next, the available capacity is assigned to each flight that requires maintenance. When there is insufficient capacity to maintain a certain flight in time, the algorithm will enter the last **if** statement and the coverage constraints will be violated.

Shift constraints:

$$h_{tc}B_{tc} = h_{tc} - H^{break}, \qquad \forall t \in T, \forall c \in C \tag{5}$$

$$H^l \leq h_{tc} \leq H^u, \qquad \forall t \in T, \forall c \in C \tag{6}$$

$$\Lambda_t^l \leq \lambda_{tc} \leq \Lambda_t^u, \qquad \forall t \in T, \forall c \in C \tag{7}$$

Constraints (6) and (7) impose restrictions on the duration and the start time of a shift. We assume that the duration and start time for shifts are defined per cycle. This means that within one cycle, each M (D, E, N) shift has the same hours, but between different cycles, shift hours may differ.

Union constraints:

$$Sn_c \leq \sum_{t \in T} \sum_{d=1}^{7} h_{tc} x_{tdc} \leq Un_c, \qquad \forall c \in C \tag{8}$$

$$W_c^l \leq n_c \leq W^u, \qquad \forall c \in C \tag{9}$$

$$W_c^l = max\ \{\Gamma_c,\ \Pi_c,\ \Omega_c\}, \qquad \forall c \in C \tag{10}$$

$$\Gamma_c = \max_{d\ =\ 1,\ldots,7} \{\sum_{t \in T} x_{tdc} + e^+_{Ndc} + e^+_{Edc}\}, \qquad \forall c \in C \tag{11}$$

$$\Pi_c = 2 \times \sum_{t \in T} x_{t6c}, \qquad \forall c \in C \tag{12}$$

$$\Omega_c = 2 \times (\sum_{t \in T} x_{t7c} + e^+_{N7c} + e^+_{E7c}), \qquad \forall c \in C \tag{13}$$

$$e^+_{N(d+1)c} = max\ \{x_{Ndc} - x_{N(d+1)c},\ 0\}, \qquad \forall d \in \{1,...,6\}, \forall c \in C \qquad (14)$$

$$e^+_{E(d+1)c} = max\ \{x_{Edc} - x_{E(d+1)c} - e^-_{N(d+1)c},\ 0\}, \qquad \forall d \in \{1,...,6\}, \forall c \in C \qquad (15)$$

$$e^-_{N(d+1)c} = max\ \{x_{N(d+1)c} - x_{Ndc},\ 0\}, \qquad \forall d \in \{1,...,6\}, \forall c \in C \qquad (16)$$

$$e^+_{N1c} = max\ \{x_{N7c} - x_{N1c},\ 0\}, \qquad \forall c \in C \qquad (17)$$

$$e^+_{E1c} = max\ \{x_{E7c} - x_{E1c} - e^-_{N1c},\ 0\}, \qquad \forall c \in C \qquad (18)$$

$$e^-_{N1c} = max\ \{x_{N1c} - x_{N7c},\ 0\}, \qquad \forall c \in C \qquad (19)$$

Constraints (8) to (19) define the union constraints. The compliance of these rules has a huge impact on the satisfaction of the workforce and, therefore, on the quality of the service. Constraint (8) is inserted to comply with the regulations concerning the minimal and maximal average working hours. Constraint (9) ensures that the number of weeks n_c (= the number of teams) in cycle c lies between the two extrema W^l_c and W^u_c. W^u_c is the maximum number of weeks in a cycle (and hence the maximum number of teams) that is considered to be manageable. W^l_c is determined by the succession and weekend constraints (see constraints (10) to (19)).

The union constraints also include succession and weekend constraints. The succession constraints entail that a night shift can only be followed by another night shift (or no shift) and an evening shift can only be followed by another evening shift or a night shift (or no shift). There is no limitation to shifts succeeding a morning or a day shift. Because a team can only work one shift a day, these succession constraints ensure that there is at least 12 hours between two shifts for each team. The weekend constraints state that workers can only be scheduled to work maximum one out of two weekends. This means that a team that is scheduled for a shift on Saturday and/or Sunday in one (two, three, etc.) weeks of the cycle, must have the whole weekend off in at least one (two, three, etc.) other weeks of the cycle.

To satisfy the succession constraints and the weekend constraint, each cycle must have at least a certain number of weeks. The minimal number of weeks required to satisfy both the succession and weekend constraints is called W^l_c. Constraint (10) calculates the derived variable W^l_c by taking the maximum of Γ_c, Π_c and Ω_c. W^l_c can now be used in constraint (9) to define the set of possible values for the derived variable n_c. Next, this set of values can be compared to the set of possible values for n_c defined by constraint (8). When the intersection of those two sets is not empty, the union constraints are satisfied. These two sets of possible values for n_c will be used in Section 4.4.3 to guide the optimization search to a feasible solution.

Domains of the decision variables:

$$x_{tdc} \in \{0, 1, ..., W^u\}, \qquad \forall t \in T, \forall d \in \{1, ..., 7\}, \forall c \in C \qquad (20)$$

$$e^+_{tdc} \in \{0, 1, ..., W^u\}, \qquad \forall t \in \{E, N\}, \forall d \in \{1, ..., 7\}, \forall c \in C \qquad (21)$$

$$e^-_{Ndc} \in \{0, 1, ..., W^u\}, \qquad \forall d \in \{1, ..., 7\}, \forall c \in C \qquad (22)$$

$$g_{fp} \geq 0, \qquad \forall f \in F, \forall p \in P \qquad (23)$$

$$h_{tc} \in \{H^l, H^l + 2, H^l + 4, ..., H^u\}, \qquad \forall t \in T, \forall c \in C \qquad (24)$$

$$\lambda_{tc} \in \{\Lambda^l_t, \Lambda^l_t + 2, \Lambda^l_t + 4, ..., \Lambda^u_t\}, \qquad \forall t \in T, \forall c \in C \qquad (25)$$

$$n_c \in \{W^l, W^l + 1, ..., W^u\}, \qquad \forall c \in C \qquad (26)$$

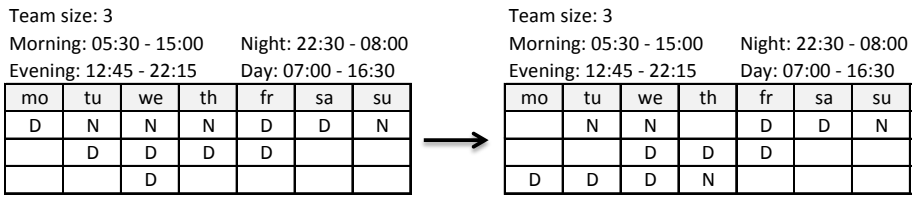
$$q_p \geq 0, \qquad \forall p \in P \qquad (27)$$

$$m_c \in \{2, 3, ...\}, \qquad \forall c \in C \qquad (28)$$

3.2 The satisfaction model

The cost model results in feasible decisions about the team size, durations, the start times and the number of shifts of a certain type on a particular day in a certain cycle. It does, however, not specify how these shifts should be distributed over the different weeks in the cycle. Cycle A in Figure 4 shows an example of the output of the cost model for one cycle. While it is known with certainty that this workforce configuration can lead to a feasible cycle in terms of coverage, shift and union constraints, it is not specified how to achieve this feasible cycle. It is for example not feasible to have a day shift following a night shift as is the case on Friday in the first week of cycle A. The satisfaction optimization model will therefore make Cycle A feasible and transform it into Cycle A'.

Figure 4: Cycle A and Cycle A'



The objective of the satisfaction optimization model is not only to create a feasible cycle, but also to maximize employee satisfaction. Workers might have some specific preferences of their own that are not imposed by the management or labour unions. These preferences are modeled as soft constraints because they can be violated in a feasible solution. Because the shift and union constraints can never be violated in a feasible solution, these constraints are modeled as hard constraints.

In order to maximize the employee satisfaction, the satisfaction model maximizes the extent to which the soft constraints are satisfied. Working with this "degree" of constraint satisfaction allows the violation of some constraints in the final solution. Satisfying these soft constraints is encouraged by rewards. Violating the hard constraints is discouraged by very large penalties. The formal model is described below.

Indices and sets:

$c \in C$: cycles in the schedule

$l \in L$: possible block lengths

$s \in S$: shifts that must be planned according to the output of the cost model

The index s refers to one particular shift that must be assigned to a certain week. For example: consider the following value for the decision variable from the cost model:

$$x_{(t=M)(d=Monday)(c=1)} = 3$$

This means that three morning shifts should be planned on Monday in the first cycle. These three shifts are then added to the set S because we now have to decide during which weeks these shifts should be scheduled.

The soft constraints:

Soft constraints can be violated in the final solution. They apply to a special case of shift succession and to block lengths. First, employees prefer to work only one type of shift within a block. A block is defined as a sequence of consecutive days with active working shifts (not interrupted by a day off). The block length is defined as the number of days in this sequence. Second, this length is preferably between 5 and 8 shifts. Hence, blocks with a length between those two values are given the highest reward. Blocks with a length smaller than 5 but greater than 1, are given a reward proportional to their length l in order to avoid too small block lengths. Blocks with a length greater than 8 do not receive a reward. These two preferences are modeled as soft constraints in the satisfaction model.

The hard constraints:

Hard constraints can never be violated in a final solution. There are two hard constraints in this model that are also present in the cost model. The first hard constraint is given by the shift succession constraints: a night shift can only be followed by another night shift and an evening shift can only be followed by another evening shift or a night shift. The second hard constraint applies to the weekend constraint: a team is only allowed to work one out of two weekends. Because the timing (the start time and duration) and the daily number of shifts for each shift type found by the cost model cannot be changed by the satisfaction model, the other constraints of the cost model will always continue to be satisfied during the satisfaction model. Therefore, it is not needed to include them in the satisfaction model.

Note that the satisfaction model theoretically allows for the violation of the hard constraints. But since it is known with certainty that the initial solution supplied by the cost model can lead to feasible cycles, it is only *during* the search of the optimal solution that some hard constraints will be violated. To avoid an infeasible *final* solution, the violation of the hard constraints is awarded a very large penalty. This penalty was set to be a value 100 times larger than the rewards given for the soft constraints.

Decision variable (integer values):

w_s : index number of the week that is assigned to shift s

Derived variables (all of them have integer values):

X_c^α : number of times a shift is followed by a shift of the same type in cycle c
X_{lc}^β : number of times there is a block length l in cycle c
X_c^χ : number of times a shift in cycle c is succeeded by a shift of a certain type that is not allowed to succeed that previous shift
X_c^δ : $= 1$ if the weekend constraint is violated in cycle c, $= 0$ otherwise

These derived variables are of course not directly altered by the optimization model, but they depend on the value of w_s. We used several small algorithms to calculate the values for these variables in practice.

Parameters:

R^α : reward given for a shift that is followed by a shift of the same type
R_l^β : reward given for a block length l
P^χ : penalty for violating a succession constraint
P^δ : penalty for violating the weekend constraint

The objective function:

$$\text{Maximize} \quad \sum_{c \in C} (X_c^\alpha \times R^\alpha + \sum_{l \in L} (X_{lc}^\beta \times R_l^\beta) - X_c^\chi \times P^\chi - X_c^\delta \times P^\delta) \tag{29}$$

Figure 4 gives an example of how the rewards and penalties for the soft and hard constraints in Function (29) lead to cycle A', which satisfies all hard constraints and outperforms cycle A in terms of employee satisfaction.

4 Methodology

4.1 Algorithm layout

We present two Tabu Search (TS) algorithms. Each TS tries to optimize the objective function of its respective optimization model defined in the problem definition in Section 3. The first Tabu Seach is therefore called the cost Tabu Search and the second the satisfaction Tabu Search. The cost TS is the most important step in the optimization process. Its goal is to produce basic workforce configurations that will be enhanced during the satisfaction TS. The satisfaction algorithm does not alter any of the decisions made in the cost TS, but assigns shifts to weeks in a particular cycle in a certain way to maximize employee satisfaction.

4.2 Search space

The search space for the cost TS contains the number and type of shifts on each day in each cycle, the start time and duration of every type of shift for each cycle and the team size in each cycle. In the satisfaction TS these variables have fixed values and constitute the initial solution. The search space of the satisfaction TS therefore only consists of the possible assignments of shifts in the workforce configuration to a specific week in its respective cycle.

Each search space can be divided into a feasible and infeasible space. The feasible search space is the search space for which all the variables have values satisfying all constraints. The infeasible search space contains variable values outside the bounds specified by the constraints. Although this search space is called infeasible, the search can still move through it. The search cannot, however, accept a solution found in the infeasible space as a global solution to the problem. It can only make use of these intermediate solutions to find its way to a better solution in the feasible search space. The use of such a feasible and infeasible search space will be exploited in the proposed algorithm. Therefore, it is important to identify these search spaces for both Tabu Searches.

The feasible search space of the cost TS consists of the combination of all possible values of its decision variables for which the coverage, the shift and the union constraints are satisfied. The infeasible space is equal to the feasible one, except that it does allow for the relaxation of the union constraints. This means that during the cost TS, the search can never enter a situation where the coverage constraints are violated. Solutions where the coverage constraints are not met are not part of any search space. Only the union constraints can be relaxed during the cost TS.

The search space of the satisfaction TS consists of all possible assignments of the shifts to the different weeks in each cycle. This satisfaction search does also have an infeasible

and feasible search space. In the feasible search space, the search can only move through solutions where the hard constraints are met. In the infeasible search space, the hard constraints are violated. The soft constraints can be satisfied or violated in both search spaces.

4.3 Initial solution

The cost TS uses an initial solution based on a fastest descent procedure. In a fastest descent method, the search selects the first move that improves the solution and then stops exploring the neighborhood instead of looking for a better option. In our case, a fastest descent algorithm was provided with an initial solution having the following characteristics:

$$
\begin{aligned}
m_c &= 3, && \forall c \in C \\
h_{tc} &= \text{random number between } (8.5 \times 4) \text{ periods} \\
&\quad \text{and } (9.5 \times 4) \text{ periods}, && \forall t \in T, \forall c \in C \\
x_{tdc} &= 3, && \forall t \in T, \forall d \in 1, ..., 7, \forall c \in C \\
\lambda_{tc} &= \text{random number between } \Lambda_t^l \text{ and } \Lambda_t^u, && \forall t \in T, \forall c \in C
\end{aligned}
$$

he fastest descent algorithm randomly lowers the value of x_{tdc} $\forall t \in T$, $\forall d \in \{1, ..., 7\}$, $\forall c \in C$ as long as the resulting solution can cover the demand for maintenance. The algorithm is allowed to continue in this way for 1000 iterations after which the final (=best) solution is saved. This procedure is repeated 15 times and results in 15 (possibly) different solutions. Then the best solution found over these repetitions is used as the initial solution for the cost TS. Note that during the search of the initial solution it is allowed to violate the union constraints.

The initial solution for the satisfaction TS is arbitrarily constructed based on the output of the cost TS. When the cost TS results in a solution in which for example

$$x_{(t=M)(d=Monday)(c=1)} = 3,$$

then three morning shifts are created on Monday of cycle 1: the first shift in the first week, the second shift in the second week and so on.

4.4 The cost Tabu Search algorithm

4.4.1 Neighborhood

The different types of moves that the search can use to create a neighborhood of a current solution are listed below. Musliu, Schaerf and Slany (2004) use similar moves in their local search for shift design. The first six moves are considered basic moves because they can only change one decision variable in the model.

Basic moves:

1.RemoveShift move: tries to remove a certain shift on a certain day in one of the cycles. Because the search wants to create promising workforce configurations, it favors the removal of shifts on peak moments (moments with high capacity). This way the available man-hours can be made smoother.
2.CreateShift move: creates a shift of a certain type on a certain day in one of the cycles. As with the RemoveShift moves, the search favors the creation of shifts on moments with the least available man-hours in order to smooth the capacity.
3.ChangeLength move: changes the end time of a certain shift type in one of the cycles and consequently changes the length or the duration of the shift.
4.ChangeStart move: changes the start time of a certain shift type in one of the cycles.
5.IncreaseTeamsize move: increases the team size in one of the cycles by one.
6.DecreaseTeamsize move: decreases the team size in one of the cycles by one.

Recall that the coverage constraints can never be violated during the search. This fact can create barriers in the search space that prevent the search from visiting some solutions. This decreases the reachability and limits the performance of the TS. Therefore, some extra moves, called complex moves, are added on top of the basic moves. Musliu et al. (2004) refer to these moves as the composite moves because they are composed of two or more basic moves.

The same problem arises with the union constraints. The fact that the search is not allowed to violate the union constraints also decreases the reachability and, hence, decreases the performance of the algorithm. Also with respect to the union constraints, special moves are added on top of the other moves. Dowsland (1998) refers to this as ejection chains because they allow to bridge many unimproving or infeasible moves which would not have been made naturally. Glover and Laguna (1997) call this method "pattern creation".

Complex moves:

7.SwapShiftBetweenCycle move: deletes a shift in a certain cycle and creates one in another cycle. The shift is swapped between two cycles. In an effort to smooth the capacity, the search favors the deletion of a shift on a day with a peak in the available man-hours. It also tries to lower the number of weekend shifts in each cycle.
8.SwapShiftInCycle move: swaps a shift between two days in the same cycle. This move is useful to decrease the number of shifts on weekend days and to smooth the available capacity if there are shifts that were only added to comply with the union rules.
9.SwapNightAndMorningShiftInAndBetweenCycle move: changes a night shift on that day in a morning shift on the next day, or a morning shift on that day in a night shift on the previous day. This move also facilitates the satisfaction of the weekend constraint. It can also decrease costs because a morning shift is less expensive than a night shift.
10.CollectiveShiftRemoval move: removes as many shifts as possible at the same time while the final result is still feasible in terms of union constraints (and of course of coverage constraints).
11.MoveBorders move: changes both the start and end time of a certain shift type in one of the cycles.

Move 5 and 6 (IncreaseTeamsize and DecreaseTeamsize) are not used in their basic form. They are joined by a series of other moves to form an ejection chain:
12.IncreaseTeamSize move: increases the team size by a certain amount. Because an increase in team size will cause more capacity and consequently more costs, a sequence of moves is added that tries to remove as many shifts as possible starting with shifts during the highest capacity day in order to find a better solution that still satisfies the union constraints. This way the IncreaseTeamSize move is able to decrease the costs in some situations.
13.DecreaseTeamSize move: decreases the team size by a certain amount. Because this leads to a decrease in capacity, it is possible that the coverage constraints will be violated. Therefore, a sequence of moves is added that adds shifts in order to cover the demand in the least expensive way without violating union constraints.

4.4.2 Strategic oscillation

Besides the use of complex moves to increase reachability, we implement a special technique called strategic oscillation. Strategic oscillation allows the search to relax one of the constraints and to make a transition from the feasible to the infeasible search space (Dowsland, 1998). After moving around in the infeasible search space for a certain number of iterations, the search can then reenter the feasible search space at a different place and explore different areas hopefully leading to a better solution. There are two advantages associated with this technique. First, it allows the search to jump over some of the barriers created by the union constraints. Second, it creates a possibility

to diversify the search. This technique is also called diversification (Glover & Laguna, 1997). The idea of using and implementing strategic oscillation in this research is based on Dowsland (1998).

The three phases that constitute the strategic oscillation method are schematically represented in Appendix A. These three phases represent the stages of (1) reentering the feasible search space, (2) moving around in the feasible search space and (3) moving around in the infeasible space.

4.4.3 Phase 1

In the first phase, the algorithm tries to bring the search back to the feasible search space. There can be two reasons for the infeasibility at the start of the first phase. First, infeasibility can be caused by the fastest descent algorithm, because it was allowed to violate the union constraints during the construction of the initial solution. Second, infeasibility can be caused by the relaxation of the union constraints during the third phase of the strategic oscillation. To model this first phase as an optimization problem that can be solved with TS, the degree of infeasibility is used as a temporary objective function. The search tries to minimize the degree of infeasibility by making moves such that the union constraints are no longer violated.

The degree of infeasibility is calculated by a continuous distance function which indicates the relative distance to the nearest feasible solution with an upper bound equal to one. This upper bound, indicating total infeasibility, does not lead to plateau move problems but facilitates the calculations. For each of the cycles in the workforce configuration, the degree of infeasibility can be defined as in Function (30).

Let $\mathbf{I}_c$ be the set of all *integer* values of n_c for which Constraint (8) holds. Then:

I_c^{min} = Smallest value in $\mathbf{I}_c$
I_c^{max} = Largest value in $\mathbf{I}_c$

$$\text{Degree of infeasibility}_c = \begin{cases} = 1 & \text{if there is infeasibility of type 1} \\ = 0 & \text{if there is no infeasibility of type 1 and 2} \\ = 1 - \frac{I_c^{max}}{W_c^l} & \text{if there is only infeasibility of type 2} \end{cases} \tag{30}$$

Infeasibility of type 1: exists when it is impossible to comply with the average working hours constraint. That is when there is no possible integer value of n_c for which Constraint (8) holds. The set $\mathbf{I}_c$ is empty in this case.

Infeasibility of type 2: exists when Expression (31) holds.

$$I_c^{max} < W_c^l \tag{31}$$

Expression (31) is deduced from Expression (32) by setting W^u to ∞. Recall from section 3.1 that we only want to consider solutions where there are at most W_c^u weeks in a cycle. We assume that the maximum number of manageable weeks in one cycle is 8. Setting W^u to ∞ instead of 8 increases the reachability and the performance of the TS, but can result in cycles with too many weeks. Therefore, the algorithm is only allowed to accept a solution as a final solution to the problem when each cycle contains at most 8 weeks. Solutions with at most 8 weeks are called *promising* to distinguish them from solutions with more than 8 weeks per cycle. Observe that a larger team size requires less weeks in a cycle and vice versa. Therefore, as the team size is restricted by a lower bound in Constraint (28), the number of weeks is rarely too high. Because we set the parameters of constraint (8) as $S = 36$ hours $\times$ 4 = 144 quarters and $U = 38$ hours $\times$ 4 = 152 quarters, I_c^{min} will equal I_c^{max} when constraint (8) is satisfied. Therefore, when the degree of infeasibility is zero, I_c^{min} will equal I_c^{max} because constraint (8) is satisfied (cfr. supra) and the value of the derived variable n_c is set to I_c^{min} (= I_c^{max}).

$$\left[I_c^{min},\ I_c^{max}\right] \cap \left[W_c^l,\ W^u\right] = \emptyset \tag{32}$$

In phase 1, the degree of infeasibility is minimized in each cycle c. The search starts exploring the neighborhood of the current solution by trying all possible moves of the move types described above. The move that leads to the solution that lowers the degree of infeasibility the most is then selected and the process continues. When the degree of infeasibility equals zero for all cycles, the search exits phase 1 and enters phase 2 of the algorithm.

4.4.4 Phase 2

In the second phase, the search has just reentered the feasible search space. Because phase 1 is only concerned with finding a feasible solution in terms of union constraints, it neglects the minimization of the total labour costs. Therefore, the labour costs tend to be relatively high at the beginning of phase 2. Thanks to the aggressive approach of phase 1, the search is very likely to reenter the feasible search space in an unexplored area providing the search with the opportunity to find a new better solution.

In phase 2, the objective is to minimize the cost function as described in Function (1).

The TS starts exploring the neighborhood based on all moves described above. During this phase, the search can only make moves where both coverage and union constraints are satisfied. This means that the search is restricted to the feasible search space. It is during this phase that promising solutions are produced. A solution is called promising when it satisfies all constraints of the cost model (Constraints (2) to (28)). This also means that each promising solution has no more than 8 weeks in each cycle. When such a promising solution has a cost equal to or lower than the previously saved promising solution, it is added to the initial solution list of the satisfaction TS algorithm.

When the search is unable to find a better solution for a certain number of iterations, the search leaves phase 2 and enters phase 3. One iteration is defined as the investigation of the current neighborhood and the execution of the best move. Preliminary test showed that after 700 iterations, the search is unlikely to find a better result. When the best solution found in phase 2 is better than the current global solution (i.e., the best solution found since the start of the algorithm), the global solution is updated.

4.4.5 Phase 3

As mentioned above, phase 3 starts with the last visited solution in phase 2. The search is then allowed to relax the union constraints. It aggressively tries to lower the costs by exploring the neighborhood. While doing so, the search will most likely make the transition to the infeasible search space.

Because of this aggressive approach, the overall algorithm performance is quite sensitive to the number of iterations in phase 3. Preliminary computational tests have indicated that the use of too many phase 3 iterations results in a situation where it is impossible to make the solution feasible again in phase 1. On the other hand, in case of too few phase 3 iterations, the algorithm converges too slowly. Instead of using a fixed number of iterations for this purpose, we have opted for a random approach. Using a little randomness makes the TS algorithm less susceptible to cycling. Cycling arises when the search gets stuck in an infinite loop over the same solutions. When this happens, the TS will not find a better global solution anymore. We use a uniform distribution between 1 and 10 to decide on the number of iterations in phase 3 every time the TS enters the third phase.

When the search exits phase 3, the best solution obtained during that phase is set as the initial solution of phase 1. The algorithm ends immediately after phase 3 when the allowed computation time has passed.

4.4.6 Oscillating tenure strategy

We use 35 as the standard tabu list length. This value was empirically found to yield good results. When the search does not find a new better solution after a certain (large) number of iterations, it could be stuck in a difficult local optimum. When the search is unable to escape from the local optimum on its own (i.e., with the use of strategic oscillation, the standard tabu list length and the basic and complex moves), we intervene by doubling the tabu list length. After a number of iterations with the doubled tabu list length, the search is hopefully far enough from the local optimum so that the standard length of 35 can be used again.

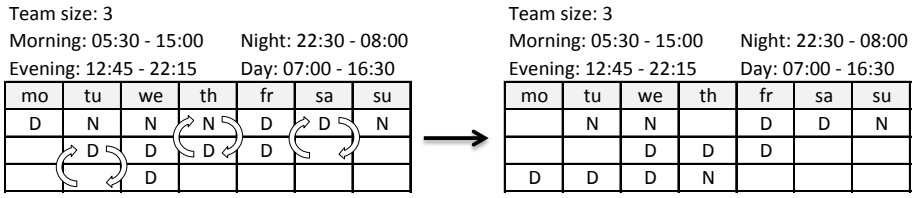
Because this oscillating tenure strategy can force the search to look somewhere else, it is a diversification strategy. Although the effects of this strategy can theoretically also be obtained with a larger static tabu list, it is more effective to increase the tabu list length only when necessary.

4.5 The satisfaction Tabu Search algorithm

4.5.1 Neighborhood

The neighborhood of the satisfaction TS is rather small compared to the neighborhood of the cost TS. It is constructed by only one type of move: the SwapShift move. This move swaps a shift between two weeks on the same day of the cycle. Figure 5 gives an example of three such moves in a cycle. Figure 5 also shows the final objective of the satisfaction TS, that is to achieve a cycle where the employee satisfaction defined in Function (29) is maximized.

Figure 5: Example of three *SwapShift* moves to reach the final objective



4.5.2 Mechanism

Although the search space of the satisfaction TS also consists of an infeasible and a feasible space, the optimization process does not run in phases. There is no specific phase to help the search return to the feasible space after it has entered the infeasible space.

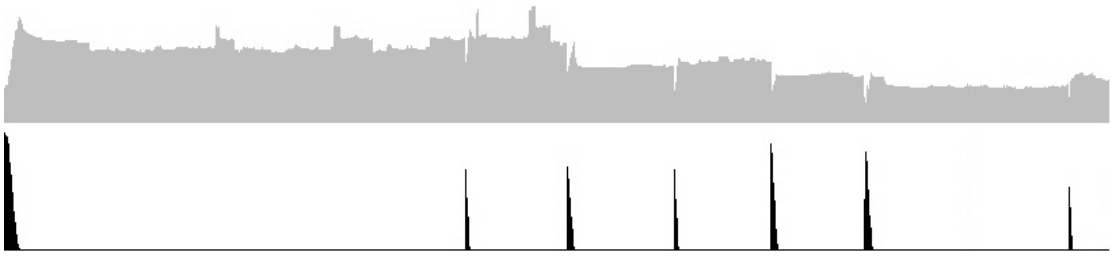
Finding a feasible solution that does not violate any hard constraints is assured by two mechanisms. First, the very high penalty of violating the hard constraints is a major incentive for the search to avoid violating these constraints. Second, it is known with certainty that there exists a feasible solution. This is true because the cost TS is subjected to the same hard constraints and the satisfaction model only uses the output from the cost model as an input.

Besides finding a feasible solution, the satisfaction TS wants to maximize the employee satisfaction (i.e., maximizing the degree of soft constraints satisfaction). The search therefore starts to explore its neighborhood by swapping the shifts. The stopping criterion for the satisfaction TS is met when 400 iterations have passed. This appears to be enough in order to obtain a good and feasible solution. Appendix B gives a graphical representation of the satisfaction TS.

4.6 Visualization

The graphs in Figure 6 give a graphical representation of the optimization progress during the cost TS algorithm. The progress of the labour costs is represented by the upper graph in gray and the progress of the degree of infeasibility is represented by the lower graph in black. These graphs can help to understand the use of strategic oscillation. Each time a black peak emerges, the search enters phase 3. The black graph will in most cases climb as long as the search stays in this phase. This is because the union constraints are then relaxed allowing the degree of infeasibility to increase. When the black peak starts to decrease again, the search enters phase 1. When the peak disappears, the search is located in phase 2. It is in this phase that promising solutions are found and saved in the initial solution list of the satisfaction TS. A similar evolution of the costs and the degree of infeasibility is also observed by Dowsland (1998) during the optimization process.

Figure 6: An example of the progress of the costs and the degree of infeasibility



The gray graph represents the costs (vertical axis) and the black graph represents the degree of infeasibility (vertical axis) at each iteration during the optimization procedure (horizontal axis).

5 Results and discussion

5.1 Computational results

The TS heuristic has been applied to a test set containing 40 instances created randomly based on real-life data dimensions from Sabena Technics. Sabena Technics is a large aircraft maintenance company located at Brussels Airport in Belgium. The company faces a weekly, cyclic demand pattern which changes only twice a year. The goal is to build the cheapest workforce configuration in order to maintain all flights in time.

The entire test set can be divided into 8 groups of 5 instances. Each group has its own specific characteristics. The first 20 instances (called 1_*_*_*) contain only 100 flights, while the next 20 instances (2_*_*_*) contain 300 flights. Another difference between the 8 groups is the distribution of the workload. This is the amount of work that each flight requires. Half of the groups are assigned workloads drawn from a uniform distribution between 0 and 10 hours (*_1_*_*), while the other half is assigned workloads drawn from an exponential distribution with an average of 3.5 hours (*_2_*_*). The last difference between the 8 groups is the way how the flight arrivals are chosen. Half of the groups is assigned peak arrivals (*_*_1_*), while the other half is assigned uniform arrivals (*_*_2_*). In the case of peak arrivals, more flights require maintenance either in the morning or the afternoon, while the demand for maintenance is uniformly distributed over the entire day in the case of uniform arrivals. Finally, 5 random instances are generated (*_*_*_1, *_*_*_2, *_*_*_3, *_*_*_4 and *_*_*_5) for each of the above cases resulting in 2x2x2x5 = 40 instances.

In Table 2, the TS results are compared with the Mixed Integer Linear Programming (MILP) model of Beliën et al. (2013) for solving the cost model. We only focus on the optimization of the cost model since the satisfaction Tabu Search only takes about one second to solve and resulted in a feasible solution regarding the succession and weekend constraints for each test instance we considered. Both the TS and the MILP model were given a computation time of 3600 seconds to solve the cost model. The TS is programmed in C++ and the MILP model was solved using CPLEX© 12.2. The table distinguishes between two scenarios in which the number of cycles was first fixed to two and then to three. For each scenario, the MILP lower bounds (LB), as well as the obtained costs and gaps for the two optimization methods are shown. The MILP lower bound (LB) is the result from applying the MILP procedure and represents the theoretically best possible objective value. The gap is the relative difference (%) between the solution value and the MILP lower bound:

$$gap = \frac{solution\ value - LB}{solution\ value} \cdot 100$$

Table 2: Results of the Tabu Search and MILP method

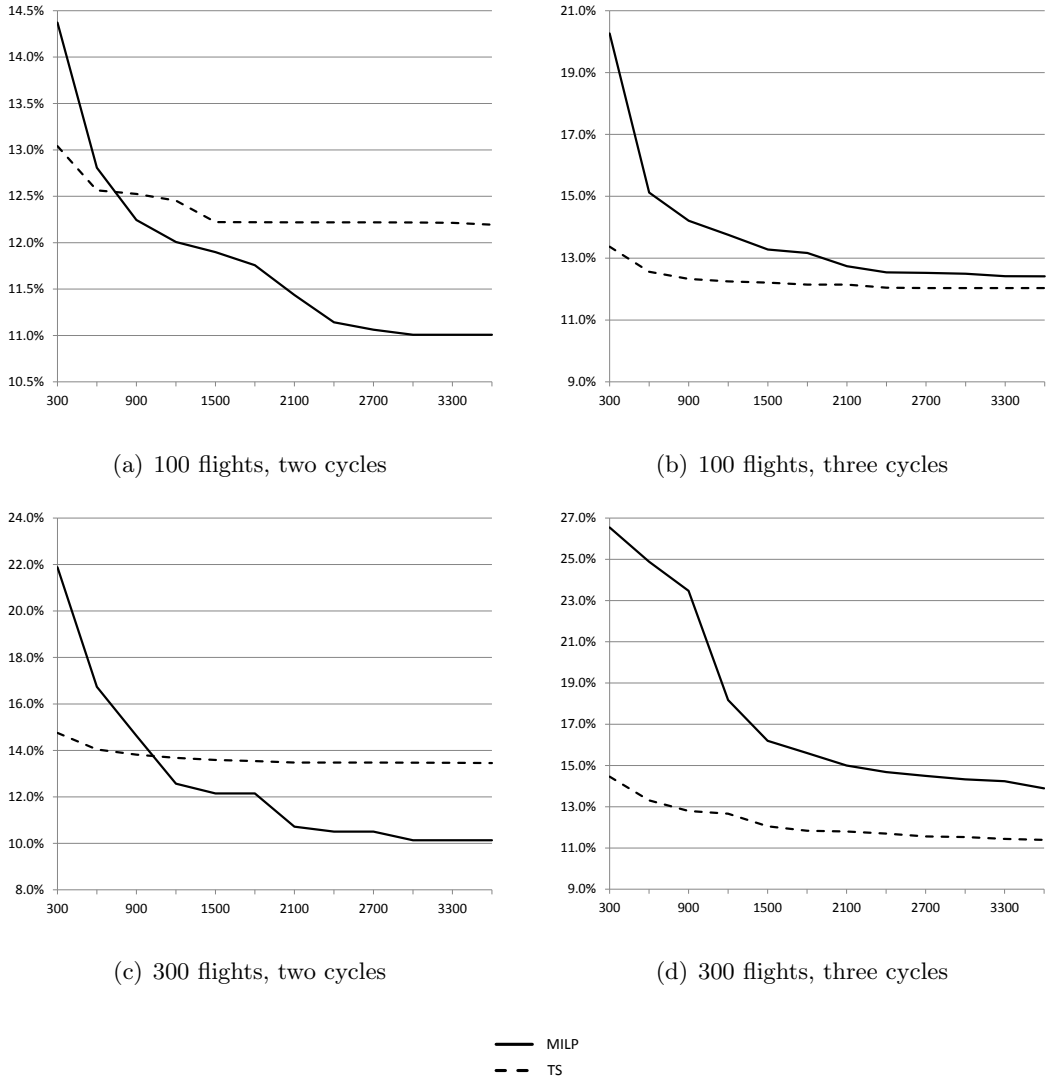
	Two cycles					Three cycles				
		TS		MILP			TS		MILP	
Test Set	LB	costs	gap(%)	costs	gap(%)	LB	costs	gap(%)	costs	gap(%)
1_1_1_1	13307	14992	11.24	14624	9.00	13303	14627	9.05	14958	11.06
1_1_1_2	17660	19391	8.93	19136	7.71	17601	19382	9.19	20506	14.17
1_1_1_3	13397	15975	16.14	16367	18.15	13381	15843	15.54	16290	17.86
1_1_1_4	13863	15571	10.97	16314	15.03	13864	15461	10.33	15647	11.40
1_1_1_5	16386	17143	4.41	17077	4.05	16352	17225	5.07	17151	4.66
1_1_2_1	11419	13355	14.50	13277	13.99	11405	13513	15.60	13386	14.80
1_1_2_2	10656	12872	17.21	14070	24.26	10675	12901	17.25	13993	23.71
1_1_2_3	18374	21113	12.97	20411	9.98	18369	21274	13.66	21454	14.38
1_1_2_4	14759	16657	11.39	16255	9.20	14705	16467	10.70	16502	10.89
1_1_2_5	14115	17308	18.45	15447	8.63	14055	17985	21.85	16431	14.46
1_2_1_1	12124	13685	11.40	13578	10.71	12184	13677	10.92	13718	11.19
1_2_1_2	14781	16594	10.92	16195	8.73	14781	16273	9.17	16673	11.35
1_2_1_3	10371	11269	7.97	11210	7.49	10341	11272	8.26	11199	7.66
1_2_1_4	10301	11837	12.98	11820	12.85	10103	11680	13.50	11809	14.44
1_2_1_5	12864	15340	16.14	14821	13.21	12865	15058	14.57	14369	10.47
1_2_2_1	8941	10872	17.76	10043	10.97	8935	10865	17.76	10771	17.04
1_2_2_2	10712	11362	5.72	11338	5.52	10685	11366	5.99	11475	6.89
1_2_2_3	12632	13305	5.06	12977	2.66	12627	13061	3.33	12982	2.73
1_2_2_4	9119	11020	17.25	10913	16.44	9119	10909	16.41	11091	17.78
1_2_2_5	9517	10872	12.46	10762	11.56	9505	10863	12.50	10717	11.31
Avg 100 flights:	12765	14527	12.19	14332	11.01	12743	14485	12.03	14556	12.41
2_1_1_1	31256	36759	14.97	34539	9.50	31245	33858	7.72	36260	13.83
2_1_1_2	29904	32824	8.90	32297	7.41	29828	32946	9.46	34066	12.44
2_1_1_3	32627	37372	12.70	34604	5.71	32625	35412	7.87	34430	5.24
2_1_1_4	30612	33041	7.35	34134	10.32	30609	32703	6.41	37925	19.29
2_1_1_5	30071	34001	11.56	34248	12.20	30068	33799	11.04	36852	18.41
2_1_2_1	28281	31333	9.74	30154	6.21	28153	31556	10.78	29322	3.99
2_1_2_2	35894	50413	28,80	38171	5,96	35708	46507	23,22	41071	13,06
2_1_2_3	27980	31187	10.28	29853	6.27	27972	30341	7.81	29277	4.46
2_1_2_4	27663	32566	15.05	30795	10.17	27663	30886	10.43	32361	14.52
2_1_2_5	27525	29922	8.01	31423	12.41	27521	30924	11.01	32868	16.27
2_2_1_1	17853	19791	9.80	19988	10.68	17852	19865	10.13	22926	22.13
2_2_1_2	16736	19525	14.28	19344	13.48	16736	18546	9.76	21058	20.53
2_2_1_3	19963	23672	15.67	21811	8.47	19949	23540	15.26	21772	8.38
2_2_1_4	17330	21544	19.56	21027	17.58	17330	20974	17.37	22985	24.60
2_2_1_5	18084	20877	13.38	22282	18.84	18083	20879	13.39	21561	16.13
2_2_2_1	15253	17675	13.70	16643	8.35	15251	17197	11.32	16886	9.68
2_2_2_2	16738	18598	10.00	18858	11.24	16738	18753	10.74	23206	27.87
2_2_2_3	18690	21751	14.07	21438	12.82	18688	21629	13.60	20445	8.60
2_2_2_4	19045	23300	18.26	21020	9.40	19045	22077	13.73	21454	11.23
2_2_2_5	20289	23338	13.06	21498	5.62	20272	21763	6.85	21847	7.21
Avg 300 flights:	24090	27974	13,46	26706	10,13	24067	27208	11,40	27929	13,89
Avg total:	18427	21250	12,83	20519	10,57	18405	20846	11,71	21242	13,15

Note: Computation time for each instance is 3600 seconds. The gap is the relative difference to the LB.

5.2 Discussion

To analyze the results of the TS, Table 2 is used in combination with Figure 7. While Table 2 only looks at the final results after 3600 seconds, Figure 7 presents the average gaps in 4 different scenarios for each five minutes.

Figure 7: Comparison of the evolution of the average gaps in four scenario's



(a) 100 flights, two cycles

(b) 100 flights, three cycles

(c) 300 flights, two cycles

(d) 300 flights, three cycles

The gaps (as percentages) are presented on the vertical axes. The computation times (in seconds) are presented on the horizontal axes.

5.2.1 3 cycles versus 2 cycles

We first compare the three cycles scenario with the two cycles scenario. From Table 2 and Figure 7 we can see that the TS is clearly outperformed by the MILP model after a computation time of 3600 seconds in the scenario with two cycles. It is only during the first 600 to 900 seconds (depending on the considered number of flights) that the TS obtains better solutions compared to the MILP model. However, when the number of cycles is fixed to three, the TS also outperforms the MILP model after 900 seconds when we look at the average gaps in Figure 7. While the average gap of the TS decreases with an extra cycle, the average gap of the MILP model increases. Clearly, the addition of the extra cycle increases the flexibility of the search which is beneficial for the TS as compared to the MILP model.

In the three cycles scenario, the TS can obtain better solutions than the MILP model, but the obtained solutions are on average not better than those of the MILP model in the scenario with two cycles. It is only for a limited number of instances that the costs decrease with the addition of the third cycle. Also, the MILP lower bound is almost the same in both scenarios suggesting that no better solutions can be found by adding an extra cycle. Regardless of the fact that more cycles will not lead to lower costs, constructing a workforce configuration with more cycles can be desirable for the company when taking into consideration the individual employee preferences.

Hence, in situations where the management wants to consider the use of three cycles, or when the size of the problem requires more than two cycles to obtain a good solution, TS is a promising alternative.

5.2.2 300 flights versus 100 flights

Figure 7 shows that the addition of more flights to the problem makes it much more difficult for the MILP model to minimize the costs in the scenario with two cycles during the first 1200 seconds. The impact of the extra flights remains relatively limited for the TS. Therefore, the TS finds better solutions than the MILP model during a longer time in the scenario with 300 flights and two cycles.

In the scenario with three cycles, the addition of 200 extra flights makes the TS clearly outperform the MILP model. Hence, when a larger number of flights must be maintained and the complexity of the problem is increased by the addition of an extra cycle, TS can deliver much better results compared to the MILP model. Moreover, when only a limited computation time is allowed (i.e., 600 to 900 seconds), TS outperforms the MILP model. This last property makes the TS algorithm very attractive in case multiple optimization problems must be solved in limited time.

6 Conclusion

This paper presents a heuristic approach for building workforce rosters for an aircraft line maintenance company. We describe two linked Tabu Search algorithms to minimize the labour costs and maximize the employee satisfaction. The cost algorithm selects promising solutions which are used as initial solutions for the satisfaction algorithm. We successfully implemented strategic oscillation and exploited the application of complex moves to avoid reachability problems caused by the various constraints.

The performance of the optimization program has been tested on 40 randomly generated instances based on real life data. We allowed for a total computation time of one hour per instance and report the results of the Tabu Search after every five minutes. The obtained solutions by the Tabu Search are then compared with those of a Mixed Integer Linear Programming (MILP) model. We found that the Tabu Search algorithm outperforms the MILP model during the first 600 to 900 seconds of computation time. After 900 seconds, the MILP approach obtains better results. When the complexity increases, i.e., when an extra cycle is added and when the number of aircraft that have to be maintained increases, the Tabu Search algorithm performs better than the MILP model. These observations make that the Tabu Search algorithm should be applied when time is of the essence and when the problem size and complexity increases.

For future research, the optimization problem can be extended in several ways, increasing the complexity. First, the timing of the lunch breaks can be optimized instead of spreading the decrease in capacity over the whole shift. In addition, part time workers or weekend cycles can be included to increase the flexibility of the optimization search. Second, the optimization model can be extended to create more realistic solutions. In some cases, the different types of aircraft a worker can maintain are limited by the skills of the worker. Moreover, aircraft maintenance takes place in a stochastic environment and one should also take into account the uncertainty in the workload or in the arrival times of the aircraft.

References

Beliën, J. (2006). *Exact and heuristic methodologies for scheduling in hospitals: problems, formulations and algorithms.* Leuven: Katholieke Universiteit Leuven. (Ph.D. dissertation)

Beliën, J., Demeulemeester, E., De Bruecker, P., Van den Bergh, J. & Cardoen, B. (2013). Integrated staffing and scheduling for an aircraft line maintenance problem. *Computers & Operations Research*, *40*(4), 1023–1033.

Burke, E. K., De Causmaecker, P., Vanden Berghe, G. & Van Landeghem, H. (2004). The state of the art of nurse rostering. *Journal of Scheduling*, *7*(6), 441–499.

Dowsland, K. (1998). Nurse scheduling with Tabu Search and strategic oscillation. *European Journal of Operational Research*, *106*, 393–407.

El-Amin, I., Duffuaa, S. & Abbas, M. (2000). A Tabu Search algorithm for maintenance scheduling of generating units. *Electric Power Systems Research*, *54*, 91–99.

Ernst, A. T., Jiang, H., Krishnamoorthy, M. & Sier, D. (2004). Staff scheduling and rostering: A review of applications, methods and models. *European Journal of Operational Research*, *153*, 3–27.

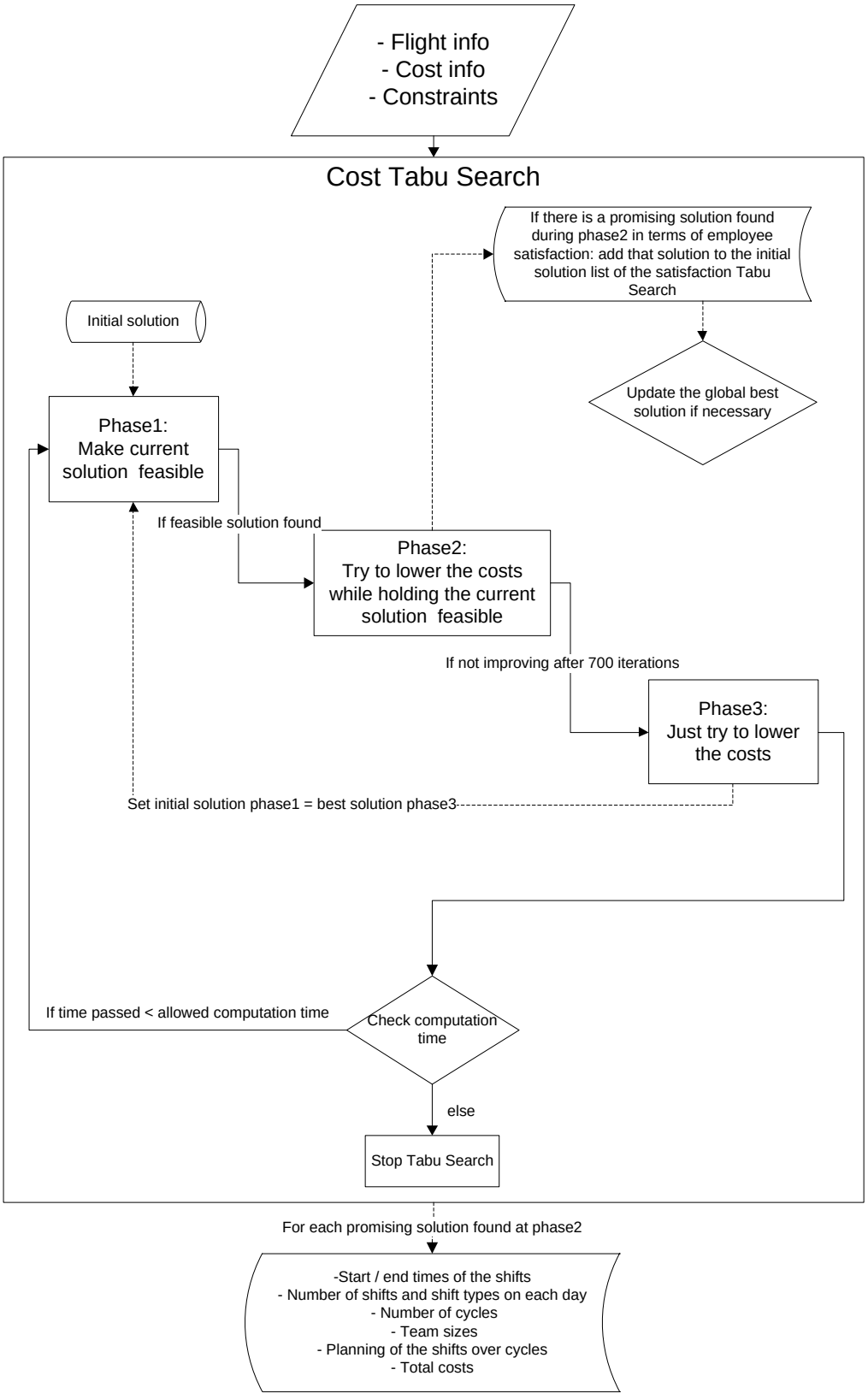
Glover, F. & Laguna, M. (1997). *Tabu Search.* Boston: Kluwer Academic Publishers.

Glover, F. & McMillan, C. (1986). The general employee scheduling problem: An integration of MS and AI. *Computers and Operations Research*, *13*(5), 563–573.

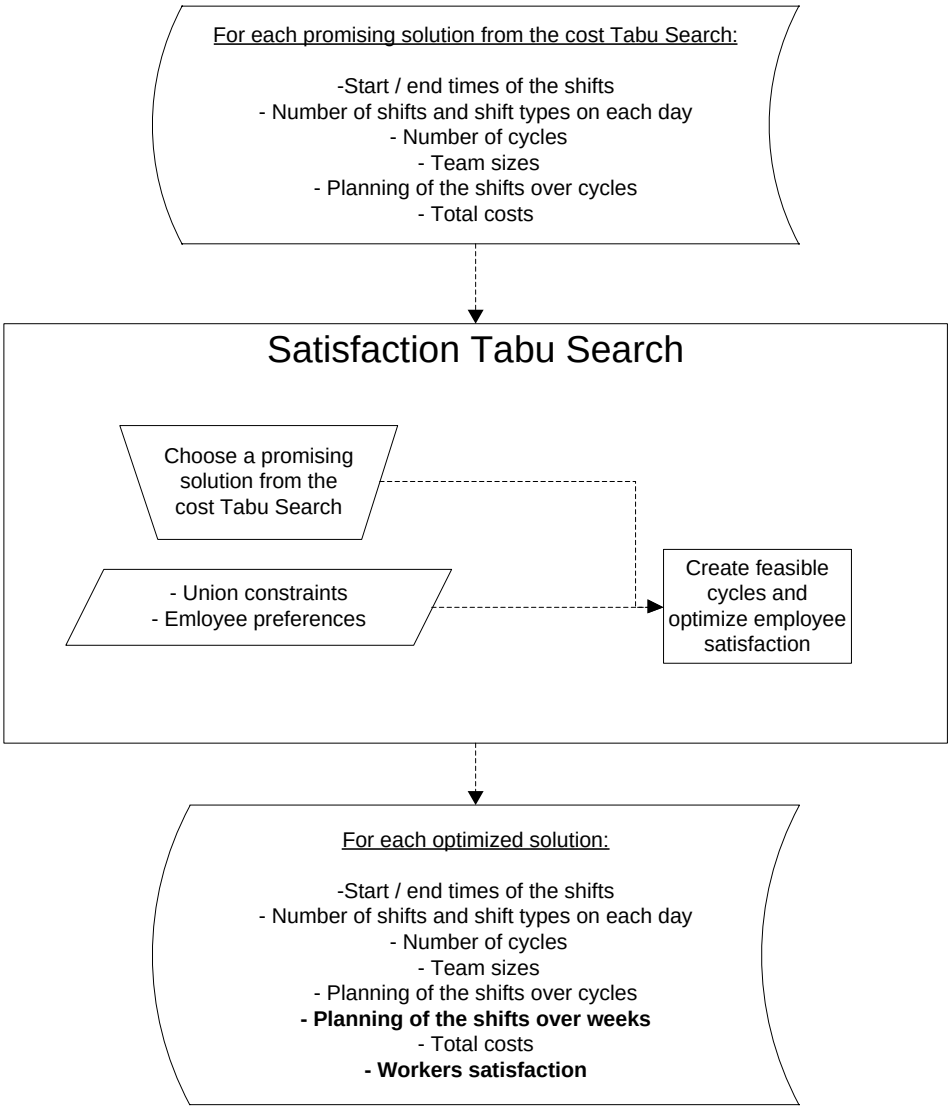
Musliu, N., Schaerf, A. & Slany, W. (2004). Local search for shift design. *European Journal of Operational Research*, *153*, 51–64.

7 Appendix

A The cost Tabu Search



B The satisfaction Tabu Search



FACULTY OF ECONOMICS AND BUSINESS
Naamsestraat 69 bus 3500
3000 LEUVEN, BELGIË
tel. + 32 16 32 66 12
fax + 32 16 32 67 91
info@econ.kuleuven.be
www.econ.kuleuven.be