# Chapter 1

# Airline Crew Pairing Optimization



## 1.1. Introduction

In the airline industry, optimizing and automating the building of crew pairings is a major financial and organizational issue. The problem consists of covering all the company's flights, programmed in a given time window, with teams made up of cockpit personnel (pilots, copilots) and of cabin personnel (stewardesses, stewards) at a minimum cost. With a frequency of several days (in the order of a week), each crew leaves from the base to which it is assigned, carries out a certain number of flights, and comes back to the base. This sequence of flights with a return to the base is called a *rotation*, or *pairing* . Drawing up the pairings of an airline company is highly constrained by international, national and internal work regulations, and by the limited availability of resources. These constraints make the problem particularly hard to solve. Besides the gains in terms of organization, security and calculation time, the use of optimization programs and models for this problem allows big companies to make substantial financial gains. It is not unusual for a reduction of 1% in the total cost of the rosters to result in savings of several tens of millions of dollars for big companies [DES 97], which is one reason for the abundant fundamental and applied research on this subject. The general crew pairing problem with resource constraints problem (CPP-RC) can be formulated as a minimum cost multicommodity flow problem with additional variables and resource constraints. Even though in most applications the cost of a rotation is non-linear [DES 97, LAV 88], in this chapter we will restrict ourselves to the case where the cost function is a linear approximation. The ways of constructing a network of feasible pairings, and calculating the cost of the rosters, along with the associated mathematical program, are presented in section 1.2. Section 1.3 gives an overview of

Chapter written by Laurent ALFANDARI and Anass NAGIH.

classical solution techniques, with a focus on the column generation method, whose associated subproblem is studied in section 1.4. Section 1.5 concludes the chapter.

## 1.2. Definition of the problem

The set of flights to be covered by the crews is denoted by $\mathcal{V} = \{1, \ldots, n\}$. The flight program and the associated timetables are established more or less exactly for the considered period, in the order of a month or a week depending on the size of the company. The term *flight* associated with each element $i \in \mathcal{V}$ is abusive in some cases, insofar as $i$ may in reality represent a sequence of aggregated and indivisible flights, that is a series of flights that can only be covered by the same crew. Also, the task to be covered by a crew is often not only a flight, but rather a flight service that may start before and end after the actual flight, to account for the time needed for preparing the plane and for accompanying passengers, for example. However, we will maintain this *flight* terminology, to help readability. For each flight $i \in \mathcal{V}$ we know:

i) the departure time $t^{\nearrow}(i)$;

ii) the arrival time $t^{\searrow}(i)$;

iii) the departure airport $a^{\nearrow}(i)$;

iv) the destination airport $a^{\searrow}(i)$.

A rotation must start and end at one of the company's bases. The set $\mathcal{B}$ of bases is generally made up of large interconnection platforms called *hubs*. The CPP often has resource constraints on the pairings. In order to take the pairing validity constraints into account, a classical modeling associates a subnetwork, constructed in the following way, with each crew.

### 1.2.1. *Constructing subnetworks*

The set of crews that may be used to cover the company's flights is indexed by $k \in \mathcal{K} = \{1, \ldots, K\}$. For $k \in \mathcal{K}$, $b^k \in \mathcal{B}$ refers to the departure and arrival bases for crew $k$. A graph $G^k = (\mathcal{X}^k, \mathcal{A}^k)$ is then associated with crew $k \in \mathcal{K}$, where $\mathcal{X}^k$ refers to the set of the network nodes and $\mathcal{A}^k$ to the set of arcs. The set $\mathcal{X}^k$ is divided into three subsets:

$$\mathcal{X}^k = \{o^k\} \cup \mathcal{V}^k \cup \{d^k\}$$

where, for $k \in \mathcal{K}$:

– the origin $o^k$ (or the destination $d^k$ respectively) refers to the source (or the sink respectively) of network $G^k$;

– $\mathcal{V}^k \subseteq \mathcal{V}$ refers to the set of flights programmed by the company that can be covered by crew $k$.

The arc set $\mathcal{A}^k$ is

$$\mathcal{A}^k = \mathcal{OV}^k \cup \mathcal{VV}^k \cup \mathcal{VD}^k \cup \{(o^k, d^k)\}$$

where

$$\begin{aligned}
\mathcal{OV}^k &= \{(o^k, i) : i \in \mathcal{V}^k, a^{\nearrow}(i) = b^k\} \\
\mathcal{VV}^k &\subseteq \mathcal{U}^k = \{(i,j) \in \mathcal{V}^k \times \mathcal{V}^k : a^{\searrow}(i) = a^{\nearrow}(j), \\
&\qquad t^{\searrow}(j) \geqslant t^{\nearrow}(i) + t_{min}(i,j)\} \\
\mathcal{VD}^k &= \{(i, d^k) : i \in \mathcal{V}^k, a^{\searrow}(i) = b^k\}
\end{aligned}$$

Passing through arc $(o^k, d^k)$ will denote that crew $k$ will not be used. $\mathcal{U}^k$ is the set of pairs of flights $(i, j)$ that satisfy the following necessary conditions for making the sequence of flights $(l, j)$ possible for the same crew:

i) the arrival airport of flight $i$ is the departure airport of flight $j$;

ii) the departure time of flight $j$ is later than the arrival time of flight $i$, with a gap greater than or equal to a value $t_{min}(i, j)$ fixed by the company or by the transit constraints of the airport.

Generally, $\mathcal{VV}^k \neq \mathcal{U}^k$ because the work regulations of the company impose a certain number of additional constraints (meal slots, breaks, overnight stops) that restrict the connection possibilities between flights. Furthermore, global constraints, which vary from one company to another, are generally associated with each rotation. Let us cite the following possible constraints:

– lower and/or upper bound on the total duration of the rotation;

– lower and/or upper bound on the total work time (total work time = total flight time + transfers + legal breaks);

– lower bound on the number of rest days;

– lower bound on the number of rest hours daily;

– upper bound on the number of flights;

– upper bound on the number of consecutive working hours.

These constraints can be modeled by the following parameters:

– a set $\mathcal{Q} = \{1, \ldots, Q\}$ of resources;

– resource consumptions $t_{ij}^{k,q}$ associated with each arc $(i, j) \in \mathcal{A}^k$ and each resource $q \in \mathcal{Q}$;

– of minimum threshold $a_i^{k,q}$ and maximum threshold $b_i^{k,q}$ for the consumption of resource $q \in \mathcal{Q}$, to be satisfied for each crew $k \in \mathcal{K}$ in each node $i$ of the network; if the resource constraints relate to the whole rotation, that is only to the destination node $d$ and not to the intermediate nodes, then $a_i^{k,q} = 0$ and $b_i^{k,q} = \infty$ for every node $i \neq d$.

### 1.2.2. *Pairing costs*

The calculation of the cost of a pairing is generally complex and varies depending on the company. This cost may be a non-linear function of several parameters such as resource consumption, total duration, and total flight time of the rotation [DES 97, LAV 88]. In order to establish a generic model for this chapter, we consider that the cost function is a linear approximation and can be decomposed by crew $k = 1, \ldots, K$ and by arcs $(i, j) \in \mathcal{A}^k$. The cost of the pairing made by crew $k \in \mathcal{K}$ will therefore be the sum of the costs $c_{ij}^k$ associated with the arcs $(i, j) \in \mathcal{A}^k$ that make up this rotation.

### 1.2.3. *Model*

The crew pairing problem with resource constraints (CPP-RC) can be modeled, if the cost function is linear, using mixed integer linear programming (MILP). We have a minimum cost multicommodity flow problem, with binary flow variables and continuous resource variables (RP-RC):

$$\text{Min} \quad \sum_{k=1}^{K} \sum_{(i,j)\in\mathcal{A}^k} c_{ij}^k x_{ij}^k \qquad [1.1]$$

$$\text{s.c.} \quad \sum_{k=1}^{K} \sum_{j:(i,j)\in\mathcal{A}^k} x_{ij}^k \geqslant 1 \qquad i \in \mathcal{V} = \{1, \ldots, n\} \qquad [1.2]$$

$$\sum_{i:(o^k,i)\in\mathcal{A}^k} x_{o^k i}^k = 1 \qquad k \in \mathcal{K} \qquad [1.3]$$

$$\sum_{i:(i,d^k)\in\mathcal{A}^k} x_{i,d^k}^k = 1 \qquad k \in \mathcal{K} \qquad [1.4]$$

$$\sum_{i:(i,j)\in\mathcal{A}^k} x_{ij}^k = \sum_{l:(j,l)\in\mathcal{A}^k} x_{jl}^k \qquad j \in \mathcal{V}^k \qquad [1.5]$$

$$T_i^{k,q} + t_{ij}^{k,q} - T_j^{k,q} \leqslant M(1 - x_{ij}^k) \quad (i,j) \in \mathcal{A}^k, k \in \mathcal{K}, q \in \mathcal{Q} \qquad [1.6]$$

$$a_i^{k,q} \leqslant T_i^{k,q} \leqslant b_i^{k,q} \qquad i \in \mathcal{V}^k, k \in \mathcal{K}, q \in \mathcal{K} \qquad [1.7]$$

$$x_{ij}^k \in \{0, 1\},\ T_i^{k,q} \geqslant 0 \qquad [1.8]$$

The binary variables $x_{ij}^k$ indicate whether the pairing uses arc $(i,j) \in \mathcal{A}^k$ (and therefore performs the sequence of flights $i$ and $j$ if $(i,j) \in \mathcal{VV}^k$), while variables $T_i^{k,q}$ indicate the cumulative consumption of each resource $q$ at each node $i$ of network $G^k$. Objective [1.1] minimizes the total cost of the pairings. Constraints [1.2] express the covering of each flight by at least one crew; if only one crew is allowed per flight, the constraint is set to equality. Constraints [1.3]–[1.5] define a path structure in the subnetwork $G^k$: the passage of a flow of one unit [1.3] or [1.4], and flow conservation at vertices [1.5]. Constraints [1.6]–[1.7] are the resource constraints associated with each rotation. Constraint [1.6], in which $M > 0$ is a very large parameter, can also be found in the following non-linear form:

$$x_{ij}^k (T_i^{k,q} + t_{ij}^{k,q} - T_j^{k,q}) \leqslant 0 \quad \text{for } (i,j) \in \mathcal{A}^k, k \in \mathcal{K}, q \in \mathcal{Q} \qquad [1.9]$$

The inequality in [1.6] or [1.9] stipulates that waiting is allowed for the crew; in the opposite case, the constraint is written as an equality. This constraint allows us to obtain the cumulated resource consumption $q$ at the node $j$, since we have:

$$T_j^{k,q} = \max(a_j^{k,q}, T_i^{k,q} + t_{ij}^{k,q})$$

Constraints [1.7] are bound constraints on the nodes of the network (time windows for example). Note that constraints [1.3]-[1.7] are local constraints only valid for subnetwork $G^k$. Only the covering constraints [1.2] are global constraints that link the $K$ subnetworks. Relaxing these linking constraints and decomposing the initial problem by subnetworks will therefore be an interesting solution option. Let us lastly note that the resource constraints [1.6]-[1.7] make the (CPP-RC) problem NP-hard. Even the associated feasibility problem is NP-complete.

### 1.2.4. *Case without resource constraints*

When the problem has no resource constraints [1.6]-[1.7], if all crews have the same valid pairings then we have a flow structure in a single network $G = (\mathcal{X}, \mathcal{A})$, where $\mathcal{X} = \{o\} \cup \mathcal{V} \cup \{d\}$, and $\mathcal{A}$ is the set of the possible connections between the nodes of the network. The model becomes:

$$\text{Min} \quad \sum_{(i,j)\in\mathcal{A}} c_{ij} x_{ij} \qquad [1.10]$$

$$\text{s.c.} \quad \sum_{i:(i,j)\in\mathcal{A}} x_{ij} \geqslant 1 \qquad \text{for } j \in \mathcal{V} = \{1, \ldots, n\} \qquad [1.11]$$

$$\text{(CPP)} \quad \sum_{i:(i,j)\in\mathcal{A}} x_{ij} = \sum_{l:(j,l)\in\mathcal{A}} x_{jl} \qquad \text{for } j \in \mathcal{V} \qquad [1.12]$$

$$x_{ij} \in \{0,1\} \qquad \text{for } (i,j) \in \mathcal{A} \qquad [1.13]$$

| Flight n° | Departure airport ($a^{\nearrow}$) | Arrival airport ($a^{\searrow}$) | Departure time ($t^{\nearrow}$) | Arrival time ($t^{\searrow}$) |
|---|---|---|---|---|
| AF456 | Paris CDG | London | 8:25 | 9:22 |
| AF132 | Paris CDG | Frankfurt | 8:01 | 8:55 |
| AF330 | Paris CDG | Zurich | 8:10 | 9:17 |
| AF254 | Frankfurt | Zurich | 11:57 | 12:42 |
| AF402 | Frankfurt | London | 12:00 | 13:18 |
| AF370 | Zurich | Frankfurt | 13:50 | 14:35 |
| AF411 | London | Paris CDG | 17:10 | 18:05 |
| AF245 | Frankfurt | Paris CDG | 19:38 | 20:30 |
| AF111 | Zurich | Paris CDG | 17:00 | 18:12 |

**Table 1.1.** *Data set example*

The parameter $c_{ij}$ is the cost associated with taking arc $(i, j) \in \mathcal{A}$. The (CPP) problem can be solved using a classical minimum cost maximum flow algorithm [FOR 62].

*Example of network construction*

We consider a simple example of the (RP) problem without resource constraints. The parameters are presented in Table 1.1.

In the graph in Figure 1.1 associated with the above parameters, we consider that the cost on an arc $(i, j) \in \mathcal{A}$ is equal to the time $t^{\nearrow}(j) - t^{\searrow}(i)$ spent between the two flights, and the objective function is to minimize the total time not spent working on flights over the set of crews. The optimal solution for the covering problem consists of employing four crews that make the following rotations:

1) AF456 + AF411;

2) AF132 + AFAF402 + AF411;

3) AF132 + AF254 + AF370 + AF245;

4) AF330 + AF111

for a total cost of 33 hours 38 minutes spent outside the planes. Of course, this fictitious example does not take into account either the resource constraints and time windows inherent in real applications, or the fact that a crew does not generally work only for the duration of the flight.

To conclude this section, let us note that studying the literature on the crew pairing problem allows us to identify several interesting variants or extensions of the problem. One of them proposes to establish in detail the composition of each crew (number of

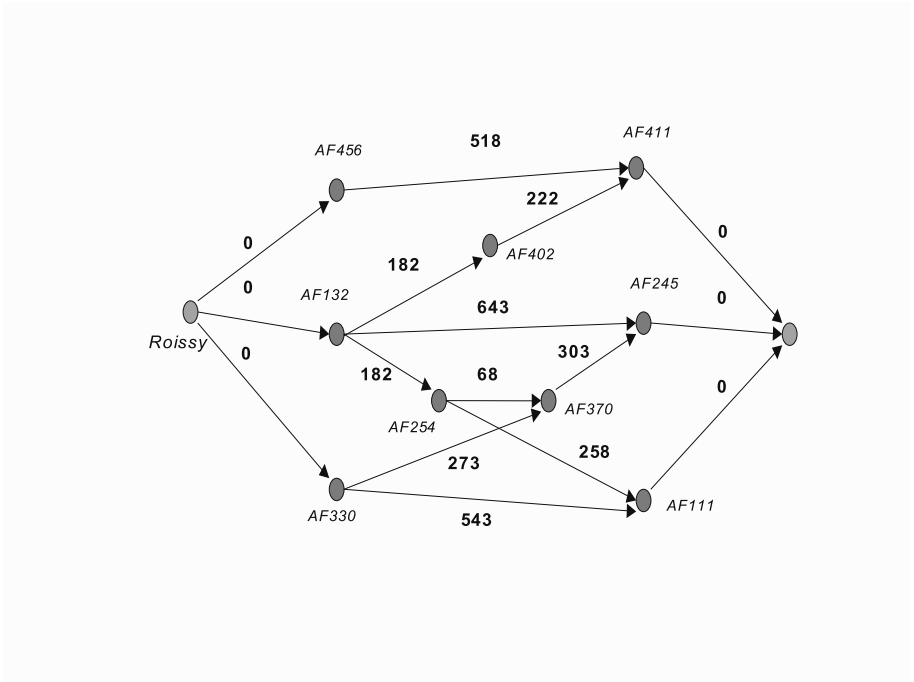


**Figure 1.1.** *A simple network example*

copilots, cabin chiefs, stewards, etc.) according to the needs expressed by the personnel category for each flight [YAN 02]. Another extension of the problem [COR 00] consists of globally and simultaneously processing the plane scheduling problem and the crew pairing problem, which are usually processed sequentially. Thus, in the classical approach, the plane scheduling problem is first solved in such a way as to establish which type of plane will cover each flight $i \in \mathcal{V}$, which allows us to deduce the set of crews $k$ that can cover the flight, that is if $i \in \mathcal{V}^k$ or $i \notin \mathcal{V}^k$. This sequential approach is clearly suboptimal with regard to a global approach, but is less complex to process.

## 1.3. Solution approaches

### 1.3.1. *Decomposition principles*

We distinguish two types of constraints in system [1.2]–[1.7]:

i) so-called linking or global covering constraints [1.2], which link the set of crews $k = 1, ..., K$;

ii) constraints [1.3]–[1.7] specific to each crew $k \in \{1, \ldots, K\}$ that define a legal itinerary for a pairing.

Since the matrix associated with constraints [1.3]–[1.7] is block diagonal, and objective [1.1] is separable (because it is linear), solving the continuous relaxation of this model can be based on Dantzig–Wolfe's decomposition. In this type of decomposition, constraints [1.3]–[1.7] define $K$ independent subproblems and global constraints [1.2] are conserved in the master problem. In a scheme of the column generation type, we must alternately solve the master problem and the $K$ subproblems. To obtain an integer solution, this scheme can be applied at each node of the search tree. The principal difficulty lies in solving the subproblems whose state spaces can increase exponentially with the number of resources $Q$, which makes the use of heuristics unavoidable. Furthermore, because the convergence of the column generation scheme is affected by the quality of the solutions provided by solving its subproblems, effectively solving real instances that come from industry requires finding a good compromise between the quality of the solutions and the solution time of the subproblems. In what follows, we give the details of the general principle of column generation for the (CPP-RC) problem.

### 1.3.2. *Column generation, master problem and subproblem*

Column generation methods (see Volume 1, Chapter 8)have been successfully applied to crew pairing problems [CRA 87, LAV 88]. In this approach, the problem is reformulated as a set covering problem (SCP) (or set partitioning problem if the covering constraint of the flights is an equality:)

$$\text{Min} \quad \sum_{r \in \mathcal{R}} c_r x_r \qquad [1.14]$$

$$\text{s.c.} \quad \sum_{r \in \mathcal{R}} a_{ir} x_r \geqslant 1 \quad \text{for } i \in \mathcal{V} = \{1, \ldots, n\} \qquad [1.15]$$

$$\text{(SCP)} \quad x_r \in \{0, 1\} \quad \text{for } r \in \mathcal{R} \qquad [1.16]$$

where $\mathcal{R}$ refers to the set of admissible pairings that satisfy the resource and flight sequence constraints, $c_r$ represents the cost of pairing $r \in \mathcal{R}$, $a_{ir} = 1$ if and only if pairing $r$ covers the flight $i$, and the binary variable $x_r$ indicates whether pairing $r$ is chosen or not in the solution.

We denote by $(\overline{SCP})$ the continuous relaxation of the (SCP) problem where integrity constraints [1.16] are replaced by $x_r \geqslant 0$ for $r \in \mathcal{R}$. Since the total number of

admissible rotations $|\mathcal{R}|$ is generally an exponential function of the number $n = |\mathcal{V}|$ of flights to be covered, exhaustive enumeration of $\mathcal{R}$ is to be avoided. Despite this, it is possible to find, in a reasonable time, an optimal solution of $(\overline{SCP})$ by only generating a limited subset of rotations (that is of columns of the constraint matrix). The principle is as follows. Let $\mathcal{R}^0$ be a feasible solution for (SCP), which includes a restricted number of rotations from $\mathcal{R}$, generated by any heuristic. We can solve, using linear programming (for example using the simplex algorithm) the program $(\overline{CP}^0)$, which is the restriction of $(\overline{CP})$ to the subset of rotations $\mathcal{R}^0$. This solution also provides a multiplier or dual variable vector:

$$(\delta_1^0, \delta_2^0, \ldots, \delta_n^0)$$

associated with the $n$ flights to be covered. The optimality criterion according to which all pairings have positive reduced cost at optimality leads us to look for the rotation of smallest reduced negative cost, that is:

$$r^0 = arg \min_{r \in \mathcal{R}} \left( c_r - \sum_{i=1}^{n} \delta_i^0 a_{ir} \right) \qquad [1.17]$$

If this pairing $r^0$ can be found in a reasonable time, we can then restart solving the covering program $(\overline{SCP})$ on the set $\mathcal{R}^1 = \mathcal{R}^0 \cup \{r^0\}$, adding the column $a_{r^0}$ to the constraint matrix. In general, at each iteration $t$ we solve the *master problem* $(\overline{SCP}^t)$:

$$\text{Min} \quad \sum_{r \in \mathcal{R}^t} c_r x_r \qquad [1.18]$$

$$\text{s.c.} \quad \sum_{r \in \mathcal{R}^t} a_{ir} x_r \geqslant 1 \quad \text{for } i \in \mathcal{V} = \{1, \ldots, n\} \qquad [1.19]$$

$$(\overline{SCP}^t) \qquad x_r \geqslant 0 \qquad \text{for } r \in \mathcal{R}^t \qquad [1.20]$$

such that:

$$\mathcal{R}^t = \mathcal{R}^{t-1} \cup \{r^{t-1}\}$$

where, if $\delta^{t-1}$ refers to the multiplier vector associated with the $n$ flights in solving $\overline{SCP}^{t-1}$, the pairing $r^{t-1}$ of negative reduced smallest cost is defined by:

$$r^{t-1} = arg \min_{r \in \mathcal{R}} \left( c_r - \sum_{i=1}^{n} \delta_i^{t-1} a_{ir} \right) \qquad [1.21]$$

The term *column generation* comes from adding column $a_{r^t}$ to the constraints matrix of the master problem at each iteration $t$. This process of iteratively solving the master problem [1.18]–[1.20] and subproblem [1.21] is stopped when all pairings are of

positive reduced cost in solving the subproblem – a sign that the continuous optimum has been reached – that is at the iteration $s$ such that:

$$\min_{r \in \mathcal{R}} \left( c_r - \sum_{i=1}^{n} \delta_i^s a_{ir} \right) > 0$$

A variant of this method, which allows us to accelerate the process [LAV 88], consists of adding, at each iteration, a subset of rotations of negative reduced cost instead of the single best rotation of subproblem [1.21]. The maximum size of this subset of entering columns can be configured in such a way as to evolve during the algorithm. The global complexity of the method strongly depends on the complexity of the subproblem, which the resource constraints make NP-hard. It is often possible, however, to solve it in a reasonable time, thanks to an *implicit* enumeration of $\mathcal{R}$, by exploiting the graph structure of the subproblem and applying variants of shortest path algorithms. This will be explained in detail in section 1.4.

### 1.3.3. *Branching methods for finding integer solutions*

In section 1.3.2, the optimal solution of the covering problem $(\overline{SCP})$ found at the end of the column generation procedure generally has a large proportion of integer components (see [LAV 88] for numerical results), but can be fractional. An initial approach for obtaining an integer solution consists of solving the (SCP) problem in integer variables with the set $\mathcal{R}^s$ of columns generated during the process. Of course, this approach does not give any theoretical guarantee of optimality but is found to be nearly optimal in practice [LAV 88].

Another tree approach of the *branch and bound* type consists of branching on the variables $x_{ij}^k$ of LP [1.2]–[1.7] and evaluating each node of the tree using the continuous relaxation of the LP. Since the number of variables and constraints is very high in the explicit model [1.2]–[1.7], the bound is generally calculated using the column generation method seen in section 1.3.2. This tree method based on column generation, commonly known as *branch and price*, proves to be more effective than the *branch and bound* method where the lower bound would be calculated by the continuous relaxation of the initial problem [1.2]–[1.7]. Another advantage of this approach is that column generation allows us to obtain a large subset of rotations that satisfy the resource constraints, whose associated variables are equal to 1 at the optimum of $(\overline{SCP})$. The number of branchings necessary to end up with the integer solution of the problem can therefore be limited for problems of medium size.

An alternative approach, known as *branch and cut*, consists of iteratively generating valid polyhedral cuts, that is cuts that only exclude fractional solutions of the problem, until the solution found is integer or it is no longer possible to add any cuts. This approach is described by Hoffman and Padberg [HOF 93]. In section 1.4, we

give details on solving the associated subproblem [1.21] for methods based on column generation.

## 1.4. Solving the subproblem for column generation

### 1.4.1. *Mathematical formulation*

In the case of several subnetworks $k = 1, \ldots, K$, since solving subproblem [1.21] can be decomposed to subnetworks, we will omit index $k$ and the graph of the subproblem will be denoted by $G = (\{o\} \cup \mathcal{V} \cup \{d\}, \mathcal{A})$.

The shortest path problem with resource constraints (SP-RC), is formulated as follows:

$$\min \quad \sum_{(i,j)\in\mathcal{A}} c_{ij}x_{ij} \qquad [1.22]$$

$$s.c. \quad \sum_{i:(i,j)\in\mathcal{A}} x_{ij} - \sum_{l:(j,l)\in\mathcal{A}} x_{jl} = \begin{cases} -1 & \text{if } j = o \\ 0 & \text{if } j \in \mathcal{V} \\ 1 & \text{if } j = d \end{cases} \qquad [1.23]$$

$$x_{ij} \geqslant 0 \qquad \forall\,(i,j) \in \mathcal{A} \qquad [1.24]$$

$$x_{ij}\left(T_i^q + t_{ij}^q - T_j^q\right) \leqslant 0 \qquad \forall\,(i,j) \in \mathcal{A},\ q \in \mathcal{Q} \qquad [1.25]$$

$$T_j^q \in [a_j^q, b_j^q] \qquad \forall\, j \in \mathcal{X},\ q \in \mathcal{Q} \qquad [1.26]$$

where variables are defined as in section 1.2.3 with exponent $k$ removed.

### 1.4.2. *General principle of effective label generation*

For some air transport problems where resource constraints relate to the duration of the crew pairings, Lavoie *et al.* [LAV 88] showed that is was possible to solve subproblem [1.21] in polynomial time using a shortest path algorithm in a graph $G$ where the valuation of each arc $(i,j) \in \mathcal{A}$ is modified in $c_t(i,j) = c(i,j) - \delta_j^{t-1}$.

However, in the general resource constraints case, verifying that resource thresholds are satisfied for the set of paths associated with the pairings is of exponential or pseudo-polynomial complexity. The objective therefore is to take advantage of the acyclic structure of the graph associated with the subproblem by employing dynamic programming techniques to limit enumeration of paths.

DEFINITION 1.1.– *We associate, with each path from the origin $o$ to node $j$, a label $(T_j, C_j) = (T_j^1, T_j^2, ..., T_j^Q, C_j)$ that represents the state of its resources and its cost.*

*The set of labels associated with the feasible paths from o to $j$ (that is which satisfy the bound constraints) is denoted by $\mathcal{E}_j$. We denote by $\mathcal{E} = \cup_{j \in V \cup \{d\}} \mathcal{E}_j$ the set of labels of the feasible paths from o to every node $j$.*

DEFINITION 1.2.– *Let $(T_j, C_j)$ and $(T'_j, C'_j)$ be two labels of $\mathcal{E}_j$. $(T_j, C_j)$ dominates $(T'_j, C'_j)$, and we state $(T_j, C_j) \preceq (T'_j, C'_j)$, if and only if:*

$$(T_j, C_j) \neq (T'_j, C'_j), \qquad C_j \leqslant C'_j \qquad and \qquad T_j^q \leqslant T_j^{'q} \ \forall\, q \in \mathcal{Q}$$

DEFINITION 1.3.– *A label $(T_j, C_j) \in \mathcal{E}_j$ is said to be efficient if it is minimal in the sense of the order relation $\preceq$, that is:*

$$\nexists (T'_j, C'_j) \in \mathcal{E}_j : (T'_j, C'_j) \preceq (T_j, C_j)$$

*A path is said to be efficient if it is associated with an efficient label. The set of efficient labels is denoted by $\mathcal{E}^{eff}$.*

The general principle of dynamic programming, which allows us to generate the set of efficient labels, first proposed in [DES 88], is as follows. In each node, the algorithm generates labels by extending the paths that correspond to the efficient labels present at the predecessor nodes. An extension is validated if it provides a legal path, otherwise it is removed. The dominance rule is then applied in order to eliminate all paths that correspond to non-efficient labels.

This algorithm therefore proceeds in two main stages. In each node $j \in \mathcal{V}$, it carries out the following operations:

1) extension of the paths (label generation and feasibility test);

2) dominance (elimination of the non-efficient labels).

Formally, for a given node $j \in \mathcal{V} \cup \{d\}$, labels are created by extending those present in nodes $i$ such that $(i, j) \in \mathcal{A}$. A new label $(T_j, C_j)$ given by:

$$\begin{array}{lcl} T_j^q & = & \max\{a_j^q, T_i^q + t_{ij}^q\},\ q \in \{1, \cdots, Q\} \\ C_j & = & C_i + c_{ij} \end{array}$$

is created at node $j$ if $T_i^q + t_{ij}^q \leqslant b_j^q,\ \forall\, q \in \{1, \cdots, Q\}$.

By considering that all predecessors of node $j \in \mathcal{V} \cup \{d\}$ have already been processed, the dominance at node $j$ can be interpreted as establishing the Pareto optima

of the multicriteria problem with $(Q+1)$ functions:

$$\left[\begin{array}{ll} \min_i C_i + c_{ij}, & \min_i \max\left\{a_j^q, \left(T_i^q + t_{ij}^q\right)\right\}, q = 1, ..., Q \\ (i,j) \in \mathcal{A} & (i,j) \in \mathcal{A} \\ T_i + t_{ij} \leqslant b_j & T_i + t_{ij} \leqslant b_j \end{array}\right] \quad [1.27]$$

Since the dominance relation $\preceq$ is a partial order relation, the number of efficient labels to be processed increases exponentially according to the number of resources, which makes the extension procedure potentially intractable. In what follows, we describe two heuristics that allow us to accelerate the efficient label generation process:

i) in the case of one single resource, the *bucket-based labeling* method of Desrochers and Soumis [DES 88];

ii) in the case of numerous resources, the *resource space reduction* method of Nagih and Soumis [NAG 06].

### 1.4.3. *Case of one single resource: the bucket method*

This method was developed in [DES 88] for the case with time windows (single resource, $Q = 1$). For each flight $i \in \mathcal{V}$, we therefore have time windows $[a_i, b_i]$. Desrochers and Soumis propose inspecting the labels associated with the paths in a defined order, calculated in the following way. Let us state:

$$(t^*, c^*) = \overset{L}{\min_{(i,j)\in\mathcal{A}}}\{(t_{ij}, c_{ij})\} \quad [1.28]$$

that is:

$$\begin{aligned} t^* &= \min_{(i,j)\in\mathcal{A}} t_{ij} \\ c^* &= \min\{c_{ij} : (i,j) \in \mathcal{A}, t_{ij} = t^*\} \end{aligned}$$

We also denote by $<^L$ the lexicographical order relation on the labels, that is $(T_i, C_i) <^L (T_j, C_j)$ if and only if $T_i < T_j$ or ($T_i = T_j$ and $C_i < C_j$).

The set of labels is decomposed into a number $H$ of distinct and ordered subsets $\mathcal{S}^h$, $h = 1, \ldots, H$, called *buckets*, of equal depth $(t^*, c^*)$. The buckets are ordered in the sense that a label belongs to one and only one bucket $\mathcal{S}^h$ and:

$$\forall h = 1, \ldots, H-1,\ \forall((T_i, C_i), (T_j, C_j)) \in \mathcal{S}^h \times \mathcal{S}^{h+1}, \quad (T_i, C_i) <^L (T_j, C_j)$$

The bucket $\mathcal{S}^h$ from stage $h$ is generated in the following way. We denote by $\mathcal{E}^0 = \emptyset$ and

$$\mathcal{E}^{h-1} = \bigcup_{l=1}^{h-1} \mathcal{S}^l$$

the set of buckets of labels generated up until stage $h-1$. Let us also state:

$$(t^h, c^h) = \overset{L}{\min}_{(T_j,C_j)\in\mathcal{E}\setminus\mathcal{E}^{h-1}} \{T_j, C_j)\} \qquad [1.29]$$

At stage $h=1$, we therefore have $(t^h, c^h) = (t^*, c^*)$. At stage $h$, bucket $\mathcal{S}^h$ is then defined by:

$$\begin{aligned} \mathcal{S}^h &= \{(T_j, C_j) \in \mathcal{E} \setminus \mathcal{E}^{h-1} : (t^h, c^h) \leqslant^L (T_j, C_j) \\ &\quad <^L (t^h, c^h) + (t^*, c^*)\} \end{aligned} \qquad [1.30]$$

The set of efficient labels created at stage $h$ according to the *extension-dominance* process described in section 1.4.2 is denoted by $\mathcal{E}^{eff,h}$. Let us take the network example in Figure 1.2 to illustrate bucket construction. We voluntarily omit the time windows in order to concentrate on the stages of the bucket construction.

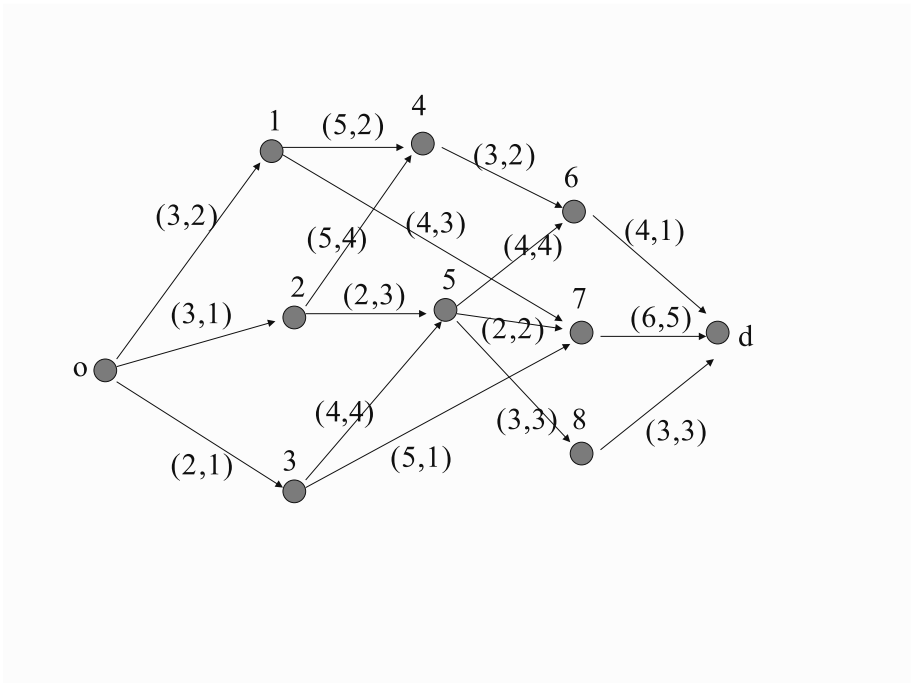


**Figure 1.2.** *Example of a graph to illustrate the bucket method*

– *Stage 1*
$(t^1, c^1) := (2, 1) \qquad (= (t^*, c^*))$
Bucket interval = [(2,1), (4,2)[
$\mathcal{S}^1 = \{(2,1)_3, (3,1)_2, (3,2)_1\}$
$\mathcal{E}^{eff,1} = S^1 \cup \{(5,4)_5, (7,2)_7, (8,4)_4\}$
Labels $(6,5)_5$, $(7,5)_7$ and $(8,5)_4$, dominated by $(5,4)_5, (7,2)_7$ and $(8,4)_4$, respectively, are eliminated.

– *Stage 2*
$(t^2, c^2) := (5, 4)$
Bucket interval = [(5,4), (7,5)[
$\mathcal{S}^2 = \{(5,4)_5, (7,2)_7\}$
$\mathcal{E}^{eff,2} = \{(8,7)_8, (9,8)_6, (13,7)_d\}$
Label $(7,6)_7$, dominated by $(7,2)_7$, is eliminated.

– *Stage 3*
$(t^3, c^3) := (8, 4)$
Bucket interval = [(8,4), (10,5)[
$\mathcal{S}^3 = \{(8,4)_4\}$
$\mathcal{E}^{eff,3} = \{(11,6)_6\}$ – *Stage 4*
$(t^4, c^4) := (8, 7)_8$
Bucket interval = [(8,7), (10,8)[
$\mathcal{S}^4 = \{(8,7)_8, (9,8)_6\}$
$\mathcal{E}^{eff,4} = \{(11,10)_d\}$ Label $(13,9)_d$, dominated by $(13,7)_d$, is eliminated.

The last stage with $\mathcal{S}^5 = \{(11,6)_6, (11,10)_d, (13,7)_d\}$ generates only dominated labels. In total, therefore, we have generated the following set of efficient labels:

$$\begin{aligned}\mathcal{E}^{eff} \quad = \quad & \{(2,1)_3, (3,1)_2, (3,2)_1, (5,4)_5, (7,2)_7, (8,4)_4, (8,7)_8, (9,8)_6, \\ & (11,6)_6, (11,10)_d, (13,7)_d\}\end{aligned}$$

Moreover, we can show that labels from the same bucket $\mathcal{S}^h$ can be processed in any order conserving optimality of the algorithm, thanks to the following proposition.

PROPOSITION 1.1.– *[DES 88] No label from bucket $\mathcal{S}^h$ is dominated by the extension of another label from $\mathcal{S}^h$, that is:*

$$\forall (T_j, C_j) \in \mathcal{S}^h, \nexists (T_i, C_i) \in \mathcal{S}^h : \ (T_i, C_i) + t_{ij} \preceq (T_j, C_j)$$

Proof: Let $(T_i, C_i) \in \mathcal{S}^h \cap \mathcal{E}_i$ and $(T_j, C_j) \in \mathcal{S}^h \cap \mathcal{E}_j$. If the extension of $(T_i, C_i)$ allowed us to eliminate $(T_j, C_j)$, we would have:

$$\begin{aligned}
(T_i, C_i) \quad &\preceq \quad (T_j, C_j) - (t_{ij}, c_{ij}) \\
&<^L \quad (T_j, C_j) - (t_{ij}, c_{ij}) \\
&<^L \quad (t^h, c^h) + (t^*, c^*) - (t_{ij}, c_{ij}) \qquad \text{because } (T_j, C_j) \in \mathcal{S}^h \\
&<^L \quad (t^h, c^h) \qquad \text{because } (t^*, c^*) \leqslant^L (t_{ij}, c_{ij})
\end{aligned}$$

which contradicts [1.30].

This bucket method can be generalized to several resources. However, its complexity is only pseudo-polynomial and increases exponentially with the number of resources [DES 88]. When this number becomes relatively large, one possible heuristic consists of reducing the resource space.

### 1.4.4. *Case of many resources: reduction of the resource space*

#### 1.4.4.1. *Reduction principle*

When solving the subproblem, the size of the search space can be reduced by strengthening the elimination of the dominance procedure. This can be done, for example, by projecting the label vectors of dimension $Q + 1$ onto a space of smaller dimension, then by applying the dominance rules in this space of reduced dimension to define the efficient labels. More exactly, let $\Pi = (\pi_p^q)$ be a real matrix $P \times (Q+1)$, with $P < Q$; the new resource vector $\tilde{T}$ is defined as follows:

$$\left\{ \begin{array}{lcl} \tilde{T}^1 & = & \pi_1^0 C + \pi_1^1 T^1 + ... + \pi_1^Q T^Q \\ \vdots & & \\ \tilde{T}^P & = & \pi_P^0 C + \pi_P^1 T^1 + ... + \pi_P^Q T^Q \end{array} \right. \qquad [1.31]$$

Selecting efficient labels based on the $P$-vector of the resources $\tilde{T}$ actually allows us to reduce the number of labels processed in each node. However, depending on the projection matrix used, we may eliminate optimal paths in this process. An initial improvement consists of defining, for each node $j$, a local projection matrix $\Pi_j = (\pi_p^q)_j$, which depends on the node processed, instead of a single global projection matrix. A second improvement can be made by considering a projection matrix $\Pi_{ij}$ whose coefficients depend on the arcs $(i, j)$ that arrive at the processed node $j$. In other words, it depends on the processed node and its predecessors [NAG 06]. However, the number of coefficients to be adjusted will be greater.

COMMENT 1.1.– In order to avoid problems of resource symmetry in the new dominance space, we restrict ourselves to block diagonal form projection matrices, that is for each $q \in \mathcal{Q} = \{0, ..., Q\}$, there exists at most one $p \in \mathcal{P} = \{1, ..., P\}$ such that the coefficient $\pi_p^q$ is non-zero. This projection thus consists of establishing a partition of the resources before combining them.

COMMENT 1.2.– The matrix $\Pi$ is only defined for the elements of $\mathcal{V}$ since there is no path to process at the origin node, and at the destination node $d$ we select the label of minimal cost.

By considering that all the predecessor nodes of node $j \in V \cup \{d\}$ have been processed, dominance in the projected space comes down to replacing the solving of [1.27] with that of the multicriteria problem with $P$ functions ($P < Q$):

$$\left[\begin{array}{l} \min_i \left\{\pi_p^0 \times (C_i + c_{ij}) + \sum_{q=1}^{Q} \pi_p^q \times \max\left\{a_j^q, \left(T_i^q + t_{ij}^q\right)\right\}\right\} \\ (i,j) \in \mathcal{A}, \\ T_i^q + t_{ij}^q \leqslant b_j^q,\ q \in \{1, \ldots, Q\} \\ p = 1, \cdots, P \end{array}\right] \quad [1.32]$$

The objective is to minimize cost $C_d(\Pi)$ at the terminal node $d$, which is the value of subproblem [1.22]–[1.26] where we substituted the new resource space $\mathcal{P}$ for the initial resource space $\mathcal{Q}$. The best approximation of the optimal value of (SP-RC) provided by dominating in the projected space (that is with regard to the resource vector $\widetilde{T} = \Pi(T, C)$), is obtained by solving:

$$\left\{\begin{array}{ll} \min_{\Pi} & C_d(\Pi) \\ s.t. & \pi_p^q \geqslant 0 \quad \forall\, q \in \{1, \ldots, Q\},\ p \in \{1, \ldots, P\} \end{array}\right. \quad [1.33]$$

which consists of establishing the best projection matrix $\Pi$. Now, function $\Pi \mapsto C_d(\Pi)$ is neither monotonic nor convex (see [NAG 06]), which makes solving this problem very hard.

The non-monotonic and non-convex behavior of the function $C_d(\Pi)$ is essentially due to the resource constraints. With resource windows, the constrained shortest path problem can have a non-zero duality jump, but without these windows the problem has the integrality property. In order to get round this hardness, Nagih and Soumis [NAG 06] propose two approaches based on the Lagrangian and surrogate relaxations of the (SP-RC) problem, where dominance is carried out in a projected space, in order to obtain lower and upper bounds and a feasible solution of the problem.

1.4.4.2. *Approach based on the Lagrangian relaxation*

Before dualizing the upper bound constraints in [1.26] on the resource consumptions:

$$lT_j^q = \max\{a_j^q, T_i^q + \tau_{ij}^q\} \leqslant b_j^q, \quad \forall (i,j) \in \mathcal{A}, q \in \{1, \ldots, Q\} \qquad [1.34]$$

we propose rewriting them in an equivalent form which allows us to express dominance in an analogous form to that in equation [1.32]. On the one hand, this new formulation must link the resource-consumption variable, the flow variable, and the dual variable associated with the upper bound constraint on the resource consumption. On the other hand, it must end up with dual variables by nodes and not by arcs.

Knowing that the bound constraints are only pertinent to the nodes $j$ that belong to an optimal path, we obtain an equivalent formulation by multiplying constraints [1.34] by $x_{ij},\ (i,j) \in \mathcal{A}$. This new constraint remains unchanged when summing over all predecessors $i$ of $j$ because the flow is borne by at most one arc.

In this way, the shortest path problem with resource constraints (SP-RC) can be rewritten in the following equivalent form:

$$\min \quad \sum_{(i,j)\in\mathcal{A}} c_{ij} x_{ij} \qquad [1.35]$$

$$s.c. \quad \sum_{i:(i,j)\in\mathcal{A}} x_{ij} - \sum_{l:(j,l)\in\mathcal{A}} x_{jl} = \begin{cases} -1 & \text{if } j = o \\ 0 & \text{if } j \in \mathcal{V} \\ 1 & \text{if } j = d \end{cases} \qquad [1.36]$$

$$x_{ij} \geqslant 0 \quad \forall (i,j) \in \mathcal{A} \qquad [1.37]$$

$$x_{ij}\left(T_i^q + t_{ij}^q - T_j^q\right) \leqslant 0 \quad \forall (i,j) \in \mathcal{A}, q \in \mathcal{Q} \qquad [1.38]$$

$$T_j^q \geqslant a_j^q \quad \forall j \in \mathcal{V}, q \in \mathcal{Q} \qquad [1.39]$$

$$\sum_{i:(i,j)\in\mathcal{A}} x_{ij}\left(\max\{a_j^q, T_i^q + t_{ij}^q\} - b_j^q\right) \leqslant 0 \quad \forall j \in \mathcal{V} \cup \{d\}, q \in \mathcal{Q} \qquad [1.40]$$

By dualizing a subset $\mathcal{Q}_1 \subset \mathcal{Q}$ of constraints [1.40], we obtain the following Lagrange function:

$$\begin{aligned}\mathcal{L}(u) \quad = \quad & \min_{[1.36]-([1.40] \text{ with } \mathcal{Q}\backslash\mathcal{Q}_1)} \sum_{(i,j)\in\mathcal{A}} (c_{ij} \\ & + \sum_{q_1\in\mathcal{Q}_1} u_j^{q_1}(\max\{a_j^{q_1}, T_i^{q_1} + t_{ij}^{q_1}\} - b_j^{q_1}) \Bigg) x_{ij} \end{aligned} \tag{1.41}$$

where $u_j^{q_1} \geqslant 0, j \in \mathcal{V} \cup \{d\},\ q_1 \in \mathcal{Q}_1$ are the associated Lagrange multipliers.

The Lagrangian dual is:

$$(LD) \quad \begin{cases} \max_u \quad \mathcal{L}(u) \\ \\ s.t. \quad u_j^{q_1} \geqslant 0, \quad \forall\, j \in \mathcal{V} \cup \{d\},\ \forall\, q_1 \in \mathcal{Q}_1 \end{cases} \tag{1.42}$$

By using formulation [1.35]–[1.40] and proceeding in the same way as in section 1.4.2, the dominance in each node $j$ then corresponds to establishing the Pareto optima of the multicriteria problem:

$$\left[ \begin{array}{l} \min_i \ \left\{ C_i + c_{ij} + \sum_{q_1\in\mathcal{Q}_1} u_j^{q_1}(\max\{a_j^{q_1}, T_i^{q_1} + t_{ij}^{q_1}\} - b_j^{q_1}) \ \mid \right. \\ \qquad\qquad \left. (i,j) \in \mathcal{A},\ T_i^{\mathcal{Q}_2} + t_{ij}^{\mathcal{Q}_2} \leqslant b_j^{\mathcal{Q}_2} \right\} \\ \\ \min_i \ \left\{ \max\left\{ a_j^{q_2}, \left(T_i^{q_2} + t_{ij}^{q_2}\right) \right\} \ \mid \right. \\ \qquad\qquad \left. (i,j) \in \mathcal{A},\ T_i^{\mathcal{Q}_2} + t_{ij}^{\mathcal{Q}_2} \leqslant b_j^{\mathcal{Q}_2} \right\}, q_2 \in \mathcal{Q}_2 = \mathcal{Q}\backslash\mathcal{Q}_1 \end{array} \right] \tag{1.43}$$

COMMENT 1.3.– Calculating the Lagrangian relaxation from formulation [1.35]–[1.40], unlike [1.22]–[1.26], allows us to express the dominance in the relaxed space as a Lagrangian relaxation of [1.27].

COMMENT 1.4.– Contrary to the approach described in section 1.4.2, solving the $(LD)$ problem can provide non-feasible solutions, and its value $v(LD)$ is a lower bound on the optimal value $v^\star$ of the (SP-RC) problem.

To establish a feasible solution we proceed as follows: a) calculate a lower bound by solving (SP-RC) by dominating over the Lagrangian cost and a subset of resource set $\mathcal{Q}_2$, and by relaxing the upper bound constraints on the resources $\mathcal{Q}_1 = \mathcal{Q}\backslash\mathcal{Q}_2$; b) calculate Lagrange multipliers associated with $\mathcal{Q}_1$; and c) calculate an upper bound and a feasible solution, solving (SP-RC) by dominating over the

Lagrangian cost and the resources $\mathcal{Q}_2$, and by validating the resource windows in the original space $\mathcal{Q}$.

In this way we deduce an approach for solving the (SP-RC) problem, based on Lagrangian relaxation, where the dominance is carried out in a projected space in order to obtain lower and upper bounds as well as a feasible solution. The Lagrange multipliers associated with the dualized constraints are used to adjust the projection matrix. The numerical results concerning this approach can be found in [NAG 06].

1.4.4.3. *Approach based on the surrogate relaxation*

Let $\mathcal{Q}_p, p = 1, \cdots, P$ ($P < Q = |\mathcal{Q}|$) be a partition of resource set $\mathcal{Q}$ into $P$ subsets. Let us consider the surrogate relaxation of the (SP-RC) problem obtained by aggregating, for each subset $\mathcal{Q}_p$, constraints [1.25] and bound constraints [1.26] in order to obtain one single resource per subset. Following the example in section 1.4.4.2, in order to obtain dual variables by nodes, we aggregate constraints [1.25] after having summed over the index $i$ of the predecessor nodes of node $j$. We obtain the (SP-RC) problem with the new resource space $\tilde{\mathcal{Q}}$:

$$\min \quad \sum_{(i,j)\in\mathcal{A}} c_{ij}x_{ij} \qquad [1.44]$$

$$s.c. \quad \sum_{i:(i,j)\in\mathcal{A}} x_{ij} - \sum_{l:(j,l)\in\mathcal{A}} x_{jl} = e_j \qquad \forall j \in \mathcal{V} \qquad [1.45]$$

$$x_{ij} \geqslant 0 \qquad \forall\,(i,j) \in \mathcal{A} \qquad [1.46]$$

$$\sum_{i:(i,j)\in\mathcal{A}} x_{ij}\left(\tilde{T}_i^p + \tilde{t}_{ij}^p - \tilde{T}_j^p\right) \leqslant 0 \qquad \forall\, j \in \mathcal{V}, p \in \tilde{\mathcal{Q}} \qquad [1.47]$$

$$\tilde{T}_j^p \in [\tilde{a}_j^p, \tilde{b}_j^p] \quad \forall\, j \in \mathcal{V}, p \in \tilde{\mathcal{Q}} \qquad [1.48]$$

with $\tilde{T}_j^p = \sum_{q\in\mathcal{Q}_p} w_j^q T_j^q$, $\tilde{a}_j^p = \sum_{q\in\mathcal{Q}_p} w_j^q a_j^q$, $\tilde{b}_j^p = \sum_{q\in\mathcal{Q}_p} w_j^q b_j^q$ and $\tilde{t}_{ij}^p = \sum_{q\in\mathcal{Q}_p} w_j^q t_{ij}^q$,, where $w_j^q$ is the multiplier of the surrogate relaxation. The associated dual is expressed by:

$$(SD) \qquad \begin{cases} \max_w \quad \sigma(w) \\ s.t. \quad w_j^q \geqslant 0, \quad \forall\, j \in \mathcal{N},\ \forall\, q \in \mathcal{Q} \end{cases} \qquad [1.49]$$

where

$$\sigma(w) = \min_{[1.45]-[1.48]} \sum_{i,j} c_{ij}x_{ij} \qquad [1.50]$$

By using formulation [1.44]–[1.48], and by proceeding in the same way as in section 1.4.4.2, the dominance in each node $j$ corresponds to establishing the Pareto optima of the multicriteria problem:

$$\left[\begin{array}{lll} \min_i \quad C_i + c_{ij}, & \min_i \quad \max\left\{\tilde{a}_j^1, \left(\tilde{T}_i^1 + \tilde{t}_{ij}^1\right)\right\}, & \ldots, \\ (i,j) \in \mathcal{A} & (i,j) \in \mathcal{A} & \\ \tilde{T}_i^{\tilde{\mathcal{Q}}} + \tilde{t}_{ij}^{\tilde{\mathcal{Q}}} \leqslant \tilde{b}_j^{\tilde{\mathcal{Q}}} & \tilde{T}_i^{\tilde{\mathcal{Q}}} + \tilde{t}_{ij}^{\tilde{\mathcal{Q}}} \leqslant \tilde{b}_j^{\tilde{\mathcal{Q}}} & \\ & & \\ & \ldots, \quad \min_i \max\left\{\tilde{a}_j^Q, \left(\tilde{T}_i^Q + \tilde{t}_{ij}^Q\right)\right\} & \\ & (i,j) \in \mathcal{A} & \\ & \tilde{T}_i^{\tilde{\mathcal{Q}}} + \tilde{t}_{ij}^{\tilde{\mathcal{Q}}} \leqslant \tilde{b}_j^{\tilde{\mathcal{Q}}} & \end{array}\right] \quad [1.51]$$

COMMENT 1.5.– Solving the (SD) problem provides, in general, non-feasible solutions, and its value $v(SD)$ is a lower bound on the optimal value $v^\star$ of the (SP-RC) problem.

The projection matrix associated with the *surrogate* approach will be calculated as follows: a) solve (SP-RC) with the set of resources $\tilde{\mathcal{Q}}$; b) deduce from this set the multipliers of the *surrogate* associated with the various resources; and then c) solve (SP-RC) by dominating over the cost and the resources $\tilde{\mathcal{Q}}$, and by validating the resource windows in the original space $\mathcal{Q}$.

## 1.5. Conclusion

In this chapter devoted to solving the crew pairing problem with resource constraints (CPP-RP), we have mainly developed column generation and master/subproblem decomposition approaches. Since the difficulty of solving the subproblem is directly linked to the number of resources, we have especially studied the techniques for reducing the resource space, this notion of reduction being a key element in the efficiency of the global solving of the problem. Indeed, while in a strategic planning context the calculation time can prove to be less critical than the global cost of the rotations schedule, in an operational context, the gain in time for solving the subproblem becomes a major stake. There are many research prospects on this problem, for example problems of reconstructing a robust solution following a disruption of the initially constructed schedule by some unforeseen event. These re-optimization problems are arousing increasing interest in engineers responsible for scheduling in large transportation companies, and open particularly interesting and promising research paths.

## 1.6. Bibliography

[COR 00] CORDEAU J., STOJKOVIC G., SOUMIS F., DESROSIERS J., Benders Decomposition for simultaneous aircraft routing and crew scheduling, Report num. G-2000-37, Cahiers du GERAD, 2000.

[CRA 87] CRAINIC T., ROUSSEAU J., "The column generation principle and the airline crew scheduling problem", *INFOR*, vol. 25, p. 136–151, 1987.

[DES 88] DESROCHERS M., SOUMIS F., "A generalized permanent labelling algorithm for the shortest path problem with time windows", *INFOR*, vol. 26, p. 191–212, 1988.

[DES 97] DESAULNIERS G., DESROSIERS J., DUMAS Y., MARC S., RIOUX B., SOLOMON M., SOUMIS F., "Crew pairing at Air France", *European Journal of Operational Research*, vol. 97, p. 245–259, 1997.

[FOR 62] FORD L., FULKERSON D., *Flows in Networks*, Princeton University Press, Princeton, 1962.

[HOF 93] HOFFMAN K., PADBERG M., "Solving airline crew scheduling problems by branch-and-cut", *Management Science*, vol. 39, p. 657–682, 1993.

[LAV 88] LAVOIE S., MINOUX M., ODIER E., "A new approach for crew pairing problems by column generation with an application to air transportation", *European Journal of Operational Research*, vol. 35, p. 45–58, 1988.

[NAG 06] NAGIH A., SOUMIS F., "Nodal aggregation of resource constraints in a shortest path problem", *European Journal of Operational Research*, vol. 172, p. 500–514, 2006.

[YAN 02] YAN S., TUNG T., TU Y., "Optimal construction of airline individual crew pairings", *Computers & OR*, vol. 29, p. 341–363, 2002.