European Journal of Operational Research 63 (1992) 432–444
North-Holland

Theory and Methodology

License class design: complexity and algorithms

Antoon W.J. Kolen

University of Limburg, P.O. Box 616, 6200 MD Maastricht, Netherlands

Leo G. Kroon

Erasmus University, P.O. Box 1738, 3000 DR Rotterdam, Netherlands

Received March 1990; revised November 1990

Abstract: In this paper a generalization of the Fixed Job Scheduling Problem (FSP) is considered, which appears in the aircraft maintenance process at an airport. A number of jobs have to be carried out, where the main attributes of a job are a fixed start time, a fixed finish time and an aircraft type. For carrying out these jobs a number of engineers are available. An engineer is allowed to carry out a specific job only if he has a license for the corresponding aircraft type. Furthermore, the jobs must be carried out in a non-preemptive way and each engineer can be carrying out at most one job at the same time. Within this setting natural questions to be answered ask for the minimum number of engineers required for carrying out all jobs or, more generally, for the minimum total costs for hiring engineers. In this paper a complete classification of the computational complexity of two classes of mathematical problems related to these practical questions is given. Furthermore, it is shown that the polynomially solvable cases of these problems can be solved by a combination of Linear Programming and Network Flow algorithms.

Keywords: Capacity planning; computational complexity; fixed job intervals; job scheduling

1. Introduction

Between the time of arrival and the time of departure of an aircraft at the main airport in the Netherlands the aircraft must be inspected before being allowed to take off again. If the stochastic elements at the airport are neglected, then such an inspection can be seen as a job with a fixed start time, a fixed finish time and an aircraft type. The start time and the finish time of a job might coincide with the time of arrival and the time of departure of the aircraft, but this is not necessary: a list of maintenance norms is available, which can be used for calculating the start and finish time of each job.

The jobs are carried out by a number of ground engineers. A ground engineer is allowed to carry out a specific job only if he has a license for the corresponding aircraft type. From the point of view of the operational management of the engineers it would be optimal if each ground engineer would have a license for each aircraft type. In that case the engineers could be considered as being qualitatively identical and, as a consequence, each job could be assigned to each of them. However, this solution is not a practical one, as a governmental rule in the Netherlands states that each ground engineer is allowed to have two licenses at most. Within this context both tactical capacity planning problems and operational job scheduling problems must be solved. Examples of such problems are the following.

- How many engineers are required for carrying out all jobs and what combination of licenses must each of these engineers obtain?
- How to schedule the jobs, if both the number of available engineers and the combination of licenses of each engineer are known?

In this paper the tactical capacity planning problem asking for the minimum number of engineers and for the licenses of each engineer is studied. A complete classification of the computational complexity of two classes of mathematical problems related to this practical problem is presented. In this paper it is assumed that all engineers are continuously available at the airport. Some of the consequences of the presence of a shift system are considered in some detail by Kolen and Kroon [14].

The operational job scheduling problem asking for an optimal schedule for the jobs, given the number of engineers and the licenses per engineer, has been studied by Kolen and Kroon [13]. In the present paper the results of [13] are used several times, as it turns out that there is a close connection between the tactical capacity planning problem and the operational job scheduling problem.

The remainder of this paper is organized as follows. In Section 2 a formal definition of two classes of mathematical problems related to the tactical capacity planning problem is presented. In Section 3 it is shown that some special cases of these problems can be solved by a combination of Linear Programming and Network Flow algorithms. In Section 4 a complete classification is presented of the computational complexity of the mathematical problems, that were defined in Section 2. This paper is finished with some concluding remarks in Section 5.

2. Problem definition

Suppose there are J jobs to be carried out, where for $j = 1,2,\ldots,J$ job j requires continuous processing in the time interval (s_j, f_j) and corresponds to an aircraft of aircraft type a_j. The start and finish time of job j are assumed to be rational numbers with $s_j < f_j$ and a_j is an integer with $1 \leqslant a_j \leqslant A$. Here A denotes the number of aircraft types in the system.

The jobs must be carried out in a non-preemptive way by a number of engineers. An engineer is allowed to carry out a specific job only if he has a license for the corresponding aircraft type. In principle there are $2^A - 1$ different license combinations. However, some license combinations may be excluded a priori. The number of remaining license combinations is denoted by C. The costs per engineer depend on the license combination of the engineer. For $c = 1,2,\ldots,C$ the costs of one engineer with license combination c are denoted by k_c.

The aircraft types per license combination are represented in the $A \times C$ zero/one matrix L, where the rows of L correspond to the aircraft types and the columns of L correspond to the license combinations. The interpretation of the matrix L is as follows. $L_{ac} = 1 \Leftrightarrow$ aircraft type a belongs to license combination c. Examples of matrices L, which will be studied explicitly in the sequel, are the following:

$$L_0 = [1], \quad L_1 = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}, \quad L_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix},$$

$$L_3 = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}, \quad L_4 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}, \quad L_5 = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}.$$

For example, the matrix L_0 represents a situation, where all aircraft types are considered as being identical and where, as a consequence, all jobs can be carried out by all engineers. The matrix L_2 represents a situation with 3 aircraft types and 2 license combinations, where for $c = 1,2$ an engineer with license combination c has licenses for the aircraft types c and 3.

Let the $A \times C$ zero/one matrix L be given. Then the problem License Class Design with respect to the matrix L, or LCD(L) for short, is defined as follows.

Instance of LCD(L):

- J jobs (s_j, f_j, a_j) to be carried out.
- C integers k_c, for $c = 1,2,\dots,C$, representing the costs of one engineer with license combination c.

Question:

- How to choose the numbers of engineers such that a feasible non-preemptive schedule exists for all jobs and such that the total costs of the engineers are minimum?

Note that the matrix L is assumed to be part of the type of the problem LCD(L) and that it does not belong to the instances of the problem LCD(L). Therefore a whole class of combinatorial problems has been defined, indexed by the zero/one matrices L. For a given matrix L the problem License Class Design with Uniform Cost with respect to the matrix L, or LCDUC(L) for short, is the special case of LCD(L) assuming that $k_c = k$ for $c = 1,2,\dots,C$.

For a given $A \times C$ zero/one matrix L the operational job scheduling problem mentioned in Section 2 is called License Class Scheduling with respect to the matrix L, or LCS(L) for short. Here the numbers of engineers are known and the question is whether or not there exists a feasible non-preemptive schedule for all jobs. That is, LCS(L) can be stated more formally as follows.

Instance of LCS(L):

- J jobs (s_j, f_j, a_j) to be carried out.
- C integers E_c for $c = 1,2,\dots,C$, representing the number of available engineers with license combination c and satisfying $E_c \leqslant J$.

Question:

- Does a feasible non-preemptive schedule exist for all jobs, which takes into account the licenses of the engineers?

Note again that the matrix L is assumed to belong to the type of the problem LCS(L) and that it does not belong to the instances of the problem LCS(L). The problems LCS(L) have been studied extensively by Kolen and Kroon [13]. They provide a complete classification of the computational complexity of the problems LCS(L). This classification is given in Theorem 8 in the present paper.

For any matrix L both LCS(L), LCD(L) and LCDUC(L) can be seen as generalizations of the well known Fixed Job Scheduling Problem, or FSP for short. FSP is equivalent to LCD(L_0). That is, in FSP the assumption is that all aircraft types can be considered as being identical and that, therefore, all jobs can be carried out by all engineers.

Next to applications in several job scheduling environments [9], FSP and related problems have applications in other areas, such as vehicle routing [3] and computer wiring [10,11]. The following lemma, which will be used in the sequel, characterizes the optimal solution of an instance of FSP.

Lemma 1. *In any instance of* FSP *the minimum number of engineers required for carrying out all jobs is equal to the maximum job overlap.*

Here the job overlap at the time instant t, denoted by D_t, and the maximum job overlap, denoted by D, are defined as follows.

$$D_t = |\{j \mid s_j < t < f_j\}|.$$
$$D = \max\{D_t \mid -\infty < t < \infty\}.$$

Lemma 1 is a direct consequence of a theorem of Dilworth [4] stating that in any partially ordered set the minimum number of chains required for covering all elements is equal to the size of a maximum antichain. Gupta, Lee and Leung [10] present a straightforward algorithm which can be used to calculate the maximum job overlap in $O(J \log J)$ time. This is optimal on a sequential processor, as follows from [7].

Similar problems were studied by several authors. Dondeti and Emmons [5,6] show that LCD(L_1) and LCD(L_2) can be solved in polynomial time by iteratively calculating a Maximum Flow in a specially structured directed network. Furthermore, they study the complexity introduced by limited availability of the engineers.

Kolen, Lenstra and Papadimitriou [16] consider a situation with A aircraft types and A license combinations, where a job on aircraft type a can be carried out by an engineer with license combination $c \Leftrightarrow a \geqslant c$. This corresponds in our setting to an $A \times A$ lower triangular matrix L. In [16] it is shown that in this case the problem LCS(L) is NP-complete if $A \geqslant 3$. Note that the cases with $A = 1$, $A = 2$ and $A = 3$ are represented by the matrices L_0, L_1 and L_5, respectively.

Kolen and Kroon [12] describe a situation with A aircraft types, where each engineer has exactly B licenses. One of the results in [12] is that in this situation the operational job scheduling problem is NP-complete $\Leftrightarrow 1 < B < A$. Furthermore, the tactical capacity planning problem is NP-hard $\Leftrightarrow 1 < B < A$.

Arkin and Silverberg [1] present a Dynamic Programming formulation of the operational job scheduling problem LCS(L), which shows that for any zero/one matrix L the problem LCS(L) can be solved in $O(J^{E+1})$ time. Here E denotes the total number of available engineers. Hence, if the total number of engineers is fixed, then this algorithm is polynomial in the number of jobs. This result is also significant with respect to the problems LCD(L) and LCDUC(L), as it will be shown in Section 4 that strong connections exist between the problems LCS(L) and the problems LCD(L) and LCDUC(L). Related problems in the context of the assignment of classes to classrooms in universities and high schools were studied by Carter and Tovey [2].

3. LP and network flow algorithms

In this section it is shown that LCD(L_2) can be solved in polynomial time by a combination of Linear Programming and Network Flow algorithms. As LCDUC(L_2) is a special case of LCD(L_2), this result implies that LCDUC(L_2) can also be solved in polynomial time. These results are similar to the results of dondeti and Emmons [6]. However, the method used here to establish these results is different from the method in [6].

Let I be an instance of LCD(L_2) containing J jobs to be carried out with respect to 3 aircraft types and 2 integers k_1 and k_2 representing the costs per engineer. Without loss of generality it is assumed that $0 < k_1 \leqslant k_2$. The notation $\{t_p \mid p = 0,1,2,\ldots,P\}$ is used for the set of start and finish times of the jobs in chronological order. That is, $\{t_p \mid p = 0,1,2,\ldots,P\} = \{s_j,\ f_j \mid j = 1,2,\ldots,J\}$ and $t_{p-1} < t_p$ for $p = 1,2,\ldots,P$. For $p = 1,2,\ldots,P$ and $a = 1,2,3$, the number D_{ap} denotes the job overlap in the interval (t_{p-1}, t_p) of the jobs on aircraft type a. Now an Integer Program, which can be used for solving I, contains the following decision variables.

Y_c :An integer variable indicating the required number of engineers with license combination c ($c = 1,2$).

X_{jc} :A binary variable indicating whether or not job j on aircraft type 3 has to be carried out by an engineer with license combination c ($j = 1,2,\ldots,J$ and $a_j = 3$ and $c = 1,2$).

Note that for $a = 1,2$ a job on aircraft type a can be carried out by an engineer with license combination a only. Therefore decision variables concerning these jobs are not required. The objective and the constraints of the Integer Program can be stated as follows:

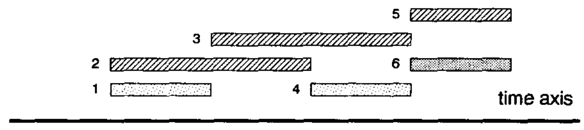
$$\min \quad \mathrm{IP} = k_1 Y_1 + k_2 Y_2 \tag{3.1}$$

subject to

$$X_{j1} + X_{j2} = 1 \quad \text{for } j = 1,2,\ldots,J \text{ and } a_j = 3, \tag{3.2}$$

$$\sum_{\{j \mid a_j = 3 \text{ and } s_j < t_p \leqslant f_j\}} X_{jc} + D_{cp} \leqslant Y_c \quad \text{for } c = 1,2 \text{ and } p = 1,2,\ldots,P, \tag{3.3}$$

All variables are integer. (3.4)



□ = aircraft type 1 ▤ = aircraft type 2 ▨ = aircraft type 3

Figure 1

The objective function (3.1) expresses that one is interested in minimizing the total costs for hiring engineers. The constraints (3.2) guarantee that each job on aircraft type 3 is carried out exactly once. The constraints (3.3) specify that for $c = 1,2$ the maximum job overlap of the jobs, which are assigned to the engineers with license combination c, should not exceed the number of available engineers with license combination c. Hence, according to Lemma 1, any feasible solution of the Integer Program can be transformed into a feasible assignment of jobs to engineers and vice versa. Finally, the integrality constraints (3.4) specify the integer character of the decision variables. The LP-Relaxation of the Integer Program is obtained by relaxing the integrality constraints (3.4) on the variables. The optimal value of the LP-Relaxation is denoted by LP.

Now it will be shown that there exists a strong connection between the solutions of the LP-Relaxation and compatible flows in some specially structured networks. This connection will be used then to establish several corollaries. Let a rational number Δ be given, which is greater than or equal to the maximum job overlap of all jobs in I. Then the network $N_1(\Delta)$ is defined as follows. The node set of $N_1(\Delta)$ is the set $\{t_p \mid p = 0,1,2,\dots,P\}$. For each job j on aircraft type 1 there is an arc from node s_j to node f_j with lower and upper capacity equal to 1. For each job j on aircraft type 3 there is an arc from node s_j to node f_j with lower capacity 0 and upper capacity 1. Furthermore, for $p = 1,2,\dots,P$ there is a dummy arc (t_{p-1}, t_p) with lower capacity 0 and upper capacity $\Delta - D_{1p} - D_{2p} - D_{3p}$. Note that, by assumption, this latter number is non-negative.

Example. The construction of the network $N_1(\Delta)$ is illustrated by the following example. Suppose that the jobs which have been represented in Figure 1 have to be carried out.

In this example the maximum job overlap is equal to 2. Hence let Δ be greater than or equal to 2. Then the network $N_1(\Delta)$ can be represented as in Figure 2.

The lower and the upper capacities of the arcs 1 and 4 are equal to 1. The upper capacities of the arcs 2, 3 and 5 are equal to 1. The straight arcs are the dummy arcs. The upper capacities of these arcs are equal to $\Delta - 2$.

Lemma 2. *If the network $N_1(\cdot)$ is constructed as described, then the following relations hold:*

(i) *If (X, Y) is a feasible solution of the* LP*-Relaxation, then there exists a flow of Y_1 units from node t_0 to node t_P in the network $N_1(Y_1 + Y_2)$.*

(ii) *If there exists a flow of δ units from node t_0 to node t_P in the network $N_1(\Delta)$, then there exists a feasible solution of the* LP*-Relaxation with $Y_1 = \delta$ and $Y_2 = \Delta - \delta$.*

Proof. In the proof of this lemma some of the technical details have been left out. For more details the reader is referred to [15], which is an extended version of the present paper.

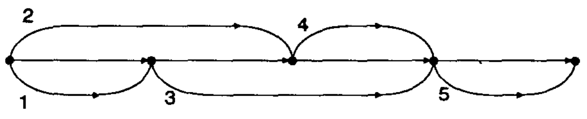


Figure 2

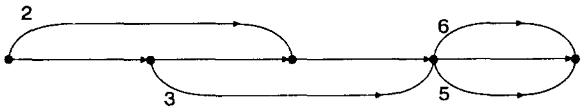


Figure 3

(i) If (X, Y) is a feasible solution of the LP-Relaxation, then $Y_1 + Y_2$ is greater than or equal to the maximum job overlap of the jobs in I. Hence the network $N_1(Y_1 + Y_2)$ is well defined. Now a flow of Y_1 units from node t_0 to node t_P in the network $N_1(Y_1 + Y_2)$ can be defined as follows. For each job on aircraft type 1 there is one unit of flow assigned to the corresponding arc in the network and for each job j on aircraft type 3 there are X_{j1} units of flow assigned to the corresponding arc in the network. Finally, for $p = 1,2,\dots, P$ there are $Y_1 - \Sigma_{\{j \mid s_j < t_p \leqslant f_j\}} X_{j1} - D_{1p}$ units of flow assigned to the dummy arc (t_{p-1}, t_p). It is not difficult to prove (see [15]) that the flow conservation rules and the capacity constraints are satisfied and that the total amount of flow is equal to Y_1 units. This completes the proof of statement (i).

(ii) If there exists a flow of δ units from node t_0 to node t_P in the network $N_1(\Delta)$, then a solution of the LP-Relaxation can be defined as follows: $Y_1 := \delta$ and $Y_2 := \Delta - \delta$. Furthermore, for each job j on aircraft type 3 the variable X_{j1} is set equal to the value of the flow in the arc corresponding to job j and the variable X_{j2} is set equal to $1 - X_{j1}$. It is not difficult to prove (see [15]) that in this way a feasible solution of the LP-Relaxation is obtained. This completes the proof of Lemma 2. □

Note that the result of Lemma 2 does not only hold for integer values of Δ but also for fractional ones. The network $N_2(\Delta)$ is constructed analog to the network $N_1(\Delta)$. The difference is that $N_2(\Delta)$ contains arcs for the jobs on aircraft type 2 or 3.

Example. For the set of jobs, which was defined in Figure 1, the network $N_2(\Delta)$ can be represented as in Figure 3.

The lower and the upper capacity of arc 6 are equal to 1. The upper capacities of the arcs 2, 3 and 5 are equal to 1. The straight arcs are the dummy arcs. The upper capacities of these arcs are equal to $\Delta - 2$. The following lemma shows for the general case that the networks $N_1(\Delta)$ and $N_2(\Delta)$ are in some sense 'dual' networks.

Lemma 3. *A flow of δ units from node t_0 to node t_P exists in the network $N_1(\Delta) \Leftrightarrow$ a flow of $\Delta - \delta$ units from node t_0 to node t_P exists in the network $N_2(\Delta)$.*

Proof. If a flow of δ units from node t_0 to node t_P exists in the network $N_1(\Delta)$, then a feasible solution of the LP-Relaxation exists with $Y_1 = \delta$ and $Y_2 = \Delta - \delta$. But then it can be shown in the same way as in Lemma 2 that a flow of $Y_2 = \Delta - \delta$ units from node t_0 to node t_P exists in the network $N_2(Y_1 + Y_2) = N_2(\Delta)$. The converse statement follows by symmetry. This completes the proof of Lemma 3. □

The relations between the LP-Relaxation and the networks $N_1(\Delta)$ and $N_2(\Delta)$ have several implications, which are described next.

Corollary 4. *If (X, Y) is an extremal solution of the* LP*-Relaxation and $Y_1 + Y_2$ is integer, then (X, Y) is all integer.*

Proof. According to Lemma 2, a compatible flow Φ exists in the network $N_1(Y_1 + Y_2)$. Suppose that the solution (X, Y) is fractional. Then it follows that the flow Φ is also fractional. However, as all capacities in the network $N_1(Y_1 + Y_2)$ are integer, it follows that Φ is not an extremal flow. Therefore Φ can be written as a convex combination of extremal compatible flows, which are all integer. According to Lemma 2, these integer compatible flows can be interpreted as integer solutions of the LP-Relaxation.

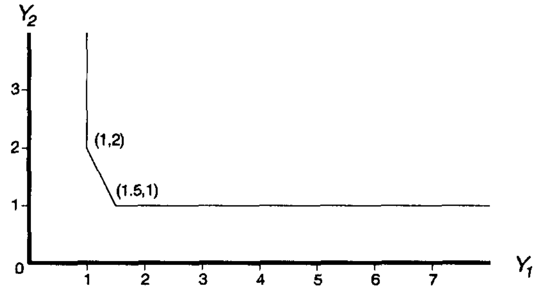


Figure 4

By linearity, it follows that (X, Y) can be written as a convex combination of integer solutions of the LP-Relaxation and hence (X, Y) is not an extremal solution. This contradiction completes the proof of Corollary 4. □

Example. This example shows that an optimal solution of the LP-Relaxation can be fractional. Suppose that the same jobs (s_j, f_j, a_j) have to be carried out as in Figure 1. Then the projection of the feasible region onto the 2-dimensional (Y_1, Y_2)-space can be represented as in Figure 4.

Figure 4 clearly shows that $(Y_1^*, Y_2^*) = (1.5, 1)$ is the unique optimal solution of the LP-Relaxation as long as $(k_1/k_2) < 2$. In that case one obtains $(X_{21}^*, X_{22}^*) = (0.5, 0.5)$, $(X_{31}^*, X_{32}^*) = (0.5, 0.5)$ and $(X_{61}^*, X_{62}^*) = (1, 0)$ for the values of the assignment variables.

Corollary 5. *If (X^*, Y^*) is an optimal solution of the* LP-*Relaxation and $Y_1^* + Y_2^*$ is fractional, then the following statements hold:*

(i) *There exists an optimal integer solution corresponding either to a Maximum Flow in the network $N_1(\lfloor Y_1^* + Y_2^* \rfloor)$ or to a Maximum Flow in the network $N_1(\lceil Y_1^* + Y_2^* \rceil)$.*

(ii) *If $k_1 = k_2 = k$, then there exists an optimal integer solution corresponding to a Maximum Flow in the network $N_1(\lceil Y_1^* + Y_2^* \rceil)$. The value of this optimal integer solution is $k\lceil Y_1^* + Y_2^* \rceil$.*

Proof. (i) First it is shown that an optimal integer solution exists, where the total number of engineers Δ^* satisfies the following inequalities:

$$\lfloor Y_1^* + Y_2^* \rfloor \leqslant \Delta^* \leqslant \lceil Y_1^* + Y_2^* \rceil. \tag{3.5}$$

To establish this result the following parametric Linear Program LP(Δ) is considered, which is obtained from the original LP-Relaxation by the addition of one parametric constraint:

$$\begin{aligned} \min \quad & \mathrm{LP}(\Delta) = k_1 Y_1 + k_2 Y_2 \\ \text{subject to} \quad & Y_1 + Y_2 = \Delta, \\ & (3.2) \text{ and } (3.3), \\ & \text{All variables are non-negative.} \end{aligned}$$

Let the set $\mathscr{D}$ denote the set of Δ's for which LP(Δ) has at least one feasible solution. As LP(Δ) is obtained from the original LP-Relaxation by the addition of one parametric constraint, the following relation between LP(Δ) and the optimal value of the LP-Relaxation is obvious.

$$\mathrm{LP} = \min\{\mathrm{LP}(\Delta) \mid \Delta \in \mathscr{D}\}. \tag{3.6}$$

As (X^*, Y^*) is an optimal solution of the LP-Relaxation, the minimum in (3.6) is obtained for $\Delta = Y_1^* + Y_2^*$. In the same way as in the proof of Corollary 4 it can be shown that each extremal solution of LP(Δ) is an integer solution whenever Δ is an integer. Thus the following relation between LP(Δ) and the optimal value of the Integer Program is obtained.

$$\mathrm{IP} = \min\{\mathrm{LP}(\Delta) \mid \Delta \in \mathscr{D} \text{ and integer}\}. \tag{3.7}$$

From the theory of Parametric Programming it is known that, on its domain, the function LP(Δ) is a convex function of Δ. As the minimum in (3.6) is obtained for $\Delta = Y_1^* + Y_2^*$, it follows that the minimum in (3.7) is obtained for $\Delta^* = \lfloor Y_1^* + Y_2^* \rfloor$ or for $\Delta^* = \lceil Y_1^* + Y_2^* \rceil$. This implies the existence of an optimal integer solution satisfying (3.5). Furthermore, the assumption $0 < k_1 \leqslant k_2$ implies that, given the total number of engineers Δ^*, the number of engineers with license combination 1 must be maximum. From Lemma 2 it follows that this can be accomplished by calculating a Maximum Flow in the network $N_1(\Delta^*)$. This completes the proof of part (i).

(ii) If $k_1 = k_2$, then $Y_1^* + Y_2^*$ is the minimum value of Δ for which a compatible flow exists in the network $N_1(\Delta)$. As a consequence, a compatible flow in the network $N_1(\lfloor Y_1^* + Y_2^* \rfloor)$ does not exist. □

4. Computational complexity

The aim of this section is to provide a complete classification of the computational complexity of the problems LCD(L) and LCDUC(L). The structure of this section is as follows. In Section 4.1 some preliminary remarks are made, which imply that one does not have to consider all zero/one matrices L in order to obtain a complete classification. In Section 4.2 a complete classification of the computational complexity of the problems LCDUC(L) is presented and in Section 4.3 the same is done for the problems LCD(L).

4.1. Preliminary remarks

In this section it is assumed that an NP-hard problem can not be solved in polynomial time and vice versa. This assumption is justified because it simplifies the notation in several places and because it is in accordance with the general opinion on the relation between the set of polynomially solvable problems and the set of NP-hard problems.

Important concepts in the classification of the computational complexity of the problems LCDUC(L) and LCD(L) are the concepts *irreducibility* of a zero/one matrix and *dominance* of a column of a zero/one matrix. These concepts are defined as follows. An $A \times C$ zero/one matrix L is called *reducible* if at least one of the following conditions is satisfied.

- L contains a complete row or a complete column of zeros.
- L contains two identical rows or two identical columns.
- By applying row or column permutations L can be written as

$$L = \begin{bmatrix} L' & 0 \\ 0 & L'' \end{bmatrix}.$$

If none of these conditions is satisfied, then the matrix L is called *irreducible*. If L contains columns c and d, which are such that $L_{ac} \leqslant L_{ad}$ for $a = 1,2,\ldots,A$, then column c is said to be *dominated* by column d. If L does not contain such columns, then L is said to be a matrix *without dominance*.

Furthermore, important tools in the classification of the computational complexity of the problems LCD(L) and LCDUC(L) are the Lemmas 6 and 7, which relate the complexities of some problems to each other. As these lemmas can be proved by evident reductions, the proofs are omitted.

Lemma 6. *If, by applying row or column permutations, L can be written as*

$$L = \begin{bmatrix} L' & 0 \\ 0 & L'' \end{bmatrix},$$

then the following statement holds: LCD(L) *can be solved in polynomial time* $\Leftrightarrow$ LCD(L') *and* LCD(L'') *can be solved in polynomial time. A similar statement holds for the problems* LCDUC(L), LCDUC(L') *and* LCDUC(L'').

Lemma 6 shows that a complete classification of the computational complexity of the problems in the set {LCD(L) | L is an irreducible zero/one matrix} implies a complete classification of the computational complexity of the problems in the set {LCD(L) | L is any zero/one matrix}. A similar statement holds for the problems LCDUC(L).

Lemma 7. *If the* $A' \times C'$ *zero/one matrix* L' *is a submatrix of the* $A \times C$ *zero/one matrix* L, *then* LCD(L') α LCD(L).

Here the notation LCD(L') α LCD(L) means that there exists a polynomial reduction from LCD(L') to LCD(L) (see [8] for a definition). As in LCDUC(L) the costs of all engineers are equal, the result of Lemma 7 does not hold for the problems LCDUC(L). This is illustrated by the following example. If the matrices L' and L are defined as follows:

$$L' = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}, \qquad L = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{bmatrix},$$

then the problem LCDUC(L') is NP-hard, as will be proved in Section 4.2. On the other hand, in the situation represented by the matrix L the engineers with one of the license combinations 1, 2 or 3 are dominated by the engineers with license combination 4. This implies that for any instance of LCDUC(L) there exists an optimal solution, where all engineers have license combination 4. Hence LCDUC(L) is equivalent to FSP, which can be solved in O(J log J) time.

More generally, if in a given license matrix L the license combination c is dominated by the license combination d and I is an instance of LCDUC(L), then there exists an optimal solution for I, where the number of engineers with license combination c is zero. As a consequence, in the analysis of the problems LCDUC(L) it may be assumed without loss of generality that the matrix L is a matrix without dominance. This implies that for each pair of license combinations c and d an aircraft type $a^{c,d}$ exists, which can be carried out by license combination c and which can not be carried out by license combination d.

In this section the results of [13] are used several times, as it turns out that there exist strong connections between the problems LCS(L) and the problems LCD(L) and LCDUC(L). In terms of the above definitions the main results of [13] with respect to the problems LCS(L) can be stated as follows.

Theorem 8. *If* L *is an irreducible zero/one matrix, then the following statement holds*: LCS(L) *is* NP-*complete* $\Leftrightarrow$ L *contains at least* 3 *columns.*

Lemma 9. *If* L *is an irreducible matrix with at least* 3 *columns, then, after a suitable permutation of the rows and the columns, it contains one of the matrices* L_3, L_4 *or* L_5 *as a submatrix.*

4.2. Complexity of LCDUC*(L)*

The aim of this subsection is to provide a complete classification of the computational complexity of the problems LCDUC(L) for irreducible zero/one matrices L without dominance. This classification is expressed in Theorem 10, which reads as follows.

Theorem 10. *If* L *is an irreducible zero/one matrix without dominance, then the following statement holds*: LCDUC(L) *is* NP-*hard* $\Leftrightarrow$ L *contains at least* 3 *columns.*

Proof. The only irreducible matrix without dominance and with one column is the matrix L_0. As was mentioned in Section 2, the problem LCDUC(L_0) is equivalent to the problem FSP and can be solved in polynomial time therefore. Furthermore, the only irreducible matrix without dominance and with two columns is the matrix L_2. As it was proved in Section 3 that LCDUC(L_2) can be solved in polynomial time, this completes the 'only if'-part of Theorem 10.

The 'if'-part is proved by a reduction from LCS(L). Note that it follows from Theorem 8 that LCS(L) is NP-complete. Let I_1 be an instance of LCS(L) containing J jobs (s_j, f_j, a_j) to be carried out and C integers E_c, for $c = 1,2,\ldots,C$ representing the number of engineers with license combination c and satisfying $E_c \leqslant J$. Let the number T be defined by: $T = 1 + \max\{f_j \mid j = 1,2,\ldots,J\}$. As the matrix L is a matrix without dominance, for each pair of different license combinations c and d there exists an aircraft type $a^{c,d}$ which can be carried out by license combination c and which cannot be carried out by license combination d. Now an instance I_2 of LCDUC(L) is defined as follows. In I_2 the following jobs have to be carried out.

- J jobs (s_j, f_j, a_j) from the instance I_1.
- E_c times the dummy job $(T+d-1,\ T+d,\ a^{c,d})$ for $c,d = 1,2,\ldots,C$ and $c \neq d$.

As C is fixed and $E_c \leqslant J$ for $c = 1,2,\ldots,C$, the size of I_2 is polynomial in the size of I_1. Now the following statement will be proved: I_1 is a yes-instance $\Leftrightarrow$ all jobs in I_2 can be carried out by $\sum_{c=1}^{C} E_c$ engineers. Suppose that all jobs in I_2 can be carried out by $\sum_{c=1}^{C} E_c$ engineers. Let Y_c be the number of engineers with license combination c in an optimal solution for I_2. Then one has:

$$\sum_{c=1}^{C} Y_c = \sum_{c=1}^{C} E_c. \tag{4.1}$$

Looking at the interval $(T+d-1,\ T+d)$ one sees that all jobs in this interval cannot be carried out by engineers with license combination d. The number of jobs in this interval is equal to $\sum_{c \neq d} E_c$. Therefore one has:

$$\sum_{c \neq d} Y_c \geqslant \sum_{c \neq d} E_c \quad \text{for } d = 1,2,\ldots,C. \tag{4.2}$$

Combining results (4.1) and (4.2) one obtains:

$$Y_d \leqslant E_d \quad \text{for } d = 1,2,\ldots,C. \tag{4.3}$$

However, equality (4.1) must be satisfied and therefore one finds:

$$Y_d = E_d \quad \text{for } d = 1,2,\ldots,C. \tag{4.4}$$

As a consequence, the schedule for the J regular jobs (s_j, f_j, a_j) of I_2 is a feasible schedule for the J jobs (s_j, f_j, a_j) of I_1 and hence I_1 is a yes-instance.
Conversely, if I_1 is a yes-instance, then one can construct a schedule for I_2 with $\sum_{c=1}^{C} E_c$ engineers by taking E_c engineers with license combination c, for $c = 1,2,\ldots,C$. The J regular jobs (s_j, f_j, a_j) of I_2 can be scheduled in the same way as the J jobs (s_j, f_j, a_j) in the feasible schedule for I_1. The dummy jobs $(T+d-1,\ T+d,\ a^{c,d})$ can be carried out by engineers with license combination c. This holds for $c,d = 1,2,\ldots,C$ and $c \neq d$.
Thus the following statement has been proved. I_1 is a yes-instance $\Leftrightarrow$ all jobs in I_2 can be carried out by $\sum_{c=1}^{C} E_c$ engineers. As LCS(L) is NP-complete, it follows that LCDUC(L) is NP-hard. This completes the 'if'-part of Theorem 10. □

For any zero/one matrix L the problem LCDUC(L) is an integer-valued minimization problem and for any instance of LCDUC(L) the minimum number of required engineers is less than or equal to the number of jobs. Furthermore, if LCDUC(L) is NP-hard, then it is NP-hard in the strong sense. This follows from the fact that in that case LCS(L) is NP-hard in the strong sense [13]. These remarks imply that, assuming P $\neq$ NP, a fully polynomial approximation scheme for LCDUC(L) does not exist if LCDUC(L) is NP-hard [8].

4.3. *Complexity of* LCD*(L)*

The aim of this paragraph is to provide a complete classification of the computational complexity of the problems LCD(L) for irreducible zero/one matrices L. This classification is expressed in Theorem 11, which reads as follows.

Theorem 11. *If L is an irreducible zero/one matrix, then the following statement holds:* LCD(L) *is* NP*-hard* $\Leftrightarrow$ *L contains at least* 3 *columns.*

The greatest irreducible matrix with two columns is the matrix L_2. As it was proved in Section 3 that LCD(L_2) can be solved in polynomial time by a combination of Linear Programming and Network Flow algorithms, this provides the 'only if'-part of Theorem 11. The 'if'-part of Theorem 11 will be established by proving that the problems LCD(L_3), LCD(L_4) and LCD(L_5) are NP-hard and by applying the Lemmas 7 and 9 next.

Lemma 12. LCD(L_3) *is* NP*-hard.*

Proof. The proof of this lemma uses a reduction from LCS(L_3). Note that it follows from Theorem 8 that LCS(L_3) is NP-complete. Let I_1 be an instance of LCS(L_3) containing J jobs (s_j, f_j, a_j) to be carried out and 3 integers E_1, E_2 and E_3 representing the numbers of engineers with the license combinations 1, 2 and 3 respectively. Let the number T be defined by: $T = 1 + \max\{f_j \mid j = 1,2,\dots,J\}$. Now an instance I_2 of LCD(L_3) is constructed. The following jobs of the form (s_j, f_j, a_j) have to be carried out.

- J jobs (s_j, f_j, a_j) from instance I_1.
- E_2 times the dummy job $(T, T+1, 2)$.
- $E_1 + E_3$ times the dummy job $(T, T+1, 1)$.
- $E_2 + E_3$ times the dummy job $(T+1, T+2, 2)$.

Furthermore, three integers k_1, k_2 and k_3 are chosen which represent the costs of the engineers and which satisfy the following relation: $\max\{k_1, k_2\} < k_3 < k_1 + k_2$. As $E_c \leqslant J$ for $c = 1,2,3$, the size of I_2 is polynomial in the size of I_1. Let the total costs of the engineers in an optimal solution for I_2 be denoted by B. Then the following statement will be proved: I_1 is a yes-instance $\Leftrightarrow B \leqslant k_1E_1 + k_2E_2 + k_3E_3$.
Suppose that I_1 is a yes-instance. Then in I_2 one can choose E_1, E_2 and E_3 engineers with the license combinations 1, 2 and 3, respectively. The regular jobs in I_2 can be scheduled in the same way as in I_1 and it is evident that there exists a feasible schedule for the dummy jobs of I_2. As the total costs for these engineers are equal to $k_1E_1 + k_2E_2 + k_3E_3$, it follows that $B \leqslant k_1E_1 + k_2E_2 + k_3E_3$.
Conversely, suppose that in I_2 an optimal solution exists satisfying $B \leqslant k_1E_1 + k_2E_2 + k_3E_3$. Let Y_1, Y_2 and Y_3 denote the numbers of engineers with the license combinations 1, 2, and 3, respectively, in this optimal solution. Then one has the following inequalities:

$$\begin{array}{ll}
Y_1 + Y_3 \geqslant E_1 + E_3 & \text{because of the interval } (T, T+1), \\
Y_2 + Y_3 \geqslant E_2 + E_3 & \text{because of the interval } (T+1, T+2), \\
Y_1 + Y_2 + Y_3 \geqslant E_1 + E_2 + E_3 & \text{because of the interval } (T, T+1), \\
k_1Y_1 + k_2Y_2 + k_3Y_3 \leqslant k_1E_1 + k_2E_2 + k_3E_3. &
\end{array}$$

Using these inequalities and the relations between k_1, k_2 and k_3, one obtains:

$$\begin{aligned}
&k_1Y_1 + k_2Y_2 + k_3Y_3 \\
&\quad \leqslant k_1E_1 + k_2E_2 + k_3E_3 \\
&\quad = (k_3 - k_2)(E_1 + E_3) + (k_3 - k_1)(E_2 + E_3) + (k_1 + k_2 - k_3)(E_1 + E_2 + E_3) \\
&\quad \leqslant (k_3 - k_2)(Y_1 + Y_3) + (k_3 - k_1)(Y_2 + Y_3) + (k_1 + k_2 - k_3)(Y_1 + Y_2 + Y_3) \\
&\quad = k_1Y_1 + k_2Y_2 + k_3Y_3
\end{aligned}$$

As the first and the last part of this chain are identical, it follows that all inequalities are equalities and therefore one has

$$Y_1 = E_1, \qquad Y_2 = E_2 \text{ and } Y_3 = E_3.$$

Consequently I_1 is a yes-instance. As LCS(L_3) is NP-complete, it follows that LCD(L_3) is NP-hard. This completes the proof of Lemma 12. □

The NP-hardness of LCD(L_4) and LCD(L_5) can be proved in the same way as the NP-hardness of LCD(L_3). Therefore the results of the Lemmas 7 and 9 can be applied to establish the validity of Theorem 11. This completes the classification of the computational complexity of the problems LCD(L).

5. Concluding remarks

In this paper the problems LCDUC(L) and LCD(L), which appear within the aircraft maintenance process at an airport, were described in a formal way. A complete classification of the computational complexity of these problems was presented and it was shown that the polynomially solvable cases can be solved by a combination of Linear Programming and Network Flow algorithms.

In the problems LCD(L) it is assumed that the costs per engineer are given by the instances of LCD(L) and in the problems LCDUC(L) it is assumed that the costs per engineer are independent of the license combinations. However, one can also assume that the costs per engineer depend on the license combinations, but are part of the type of the problem at hand. In that case a cost vector K with C entries $k_1, k_2, \ldots, k_C$ for $c = 1,2,\ldots,C$ representing the costs of one engineer with license combination c would belong to the type of the problem. The resulting problem could be called LCD(L, K). In this case the computational complexity of the problem LCD(L, K) depends both on the matrix L and on the cost vector K.

Lemma 13. *If there are* 3 *license combinations and the cost vector K representing the costs per engineer is* (k_1, k_2, k_3), *then the following statements are valid*:

(i) LCD(L_3, K) *is* NP-*hard* $\Leftrightarrow \max\{k_1, k_2\} < k_3 < k_1 + k_2$.
(ii) LCD(L_4, K) *is* NP-*hard* $\Leftrightarrow k_3 < \min\{k_1, k_2\}$.
(iii) LCD(L_5, K) *is* NP-*hard* $\Leftrightarrow k_3 < k_2 < k_1$.

The 'if'-part of (i) follows from the proof of Lemma 12. The 'only if'-part of (i) follows from the observation that at least one of the license combinations is dominated by the others if the condition $\max\{k_1, k_2\} < k_3 < k_1 + k_2$ is not satisfied. In that case there are at most two relevant license combinations and therefore the problem LCD(L_3, K) can be solved in polynomial time then. The statements (ii) and (iii) can be proved in a similar way. Although for such individual problems necessary and sufficient conditions on the cost vector K can be stated guaranteeing the NP-hardness of LCD(L, K), it is difficult to state such conditions in general. This is a topic for further research. Furthermore, we are currently looking for optimization methods to be used for calculating optimal or satisfying solutions for the cases that have been classified as NP-hard in this paper.

References

[1] Arkin, E.M., Silverberg, E.L., "Scheduling jobs with fixed start and end times", *Discrete Applied Mathematics* 18 (1987) 1–8.
[2] Carter, M.W., and Tovey, C.A., "When is classroom assignment hard?", Working paper 89-02, University of Toronto, Department of Industrial Engineering, February 1989, to appear in *Operations Research*.
[3] Dantzig, G.L., and Fulkerson, D.R., "Minimizing the number of tankers to meet a fixed schedule", *Naval Research Logistics Quarterly* 1 (1954) 217–222.
[4] Dilworth, R.P., "A decomposition principle for partially ordered sets", *Annals of Mathematics* 51 (1950) 161–166.

[5] Dondeti, V.R., and Emmons, H., "Resource requirements for scheduling with different processor sizes, Parts I and II", Technical Memoranda 579 and 589, Department of Operations Research, Case Western Reserve University, Cleveland OH, 1986.

[6] Dondeti, V.R., and Emmons, H., "Interval scheduling with processors of two types", April 1989, To appear in *Operations Research.*

[7] Fredman, M.L., and Weide, L., "On the complexity of computing the measure of $U[a_i,b_i]$", *Communications of the ACM* 21 (1978) 540–544.

[8] Garey, M.R., and Johnson, D.S., *Computers and Intractability: A Guide to the Theory of NP-completeness*, Freeman, San Francisco, CA, 1979.

[9] Gertsbakh, I., and Stern, H.I., "Minimal resources for fixed and variable job schedules", *Operations Research* 18 (1978) 68–85.

[10] Gupta, U.L., Lee, D.T., and Leung, J.Y.-T., "An optimal solution to the channel assignment problem", *IEEE Transactions on Computers* 28 (1979) 807–810.

[11] Hashimoto, A., and Stevens, J., "Wire routing by optimizing channel assignments within large apertures", in: *Proceedings of the 8th Design Automation Workshop*, 1971, 155–169.

[12] Kolen, A.W.J., and Kroon, L.G., "On the computational complexity of (A, L) class scheduling", Report 34 of the Rotterdam School of Management, 1989.

[13] Kolen, A.W.J. and Kroon, L.G., "On the computational complexity of (maximum) class scheduling", *European Journal of Operational Research* 54 (1991) 23–38.

[14] Kolen, A.W.J., and Kroon, L.G., "On the computational complexity of (maximum) shift class scheduling", Report 48 of the Rotterdam School of Management, 1989, To appear in the *European Journal of Operational Research.*

[15] Kolen, A.W.J., and Kroon, L.G., "More concerning license class design", Report 64 of the Rotterdam School of Management, 1989.

[16] Kolen, A.W.J., Lenstra, J.K., and Papadimitriou, C.H., "Interval scheduling problems", Unpublished manuscript, 1987.