European Journal of Operational Research 54 (1991) 23–38
North-Holland

On the computational complexity of (maximum) class scheduling

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 September 1989; revised November 1989

Abstract: In this paper we consider several generalizations of the Fixed Job Scheduling Problem (FSP) which appear in a natural way 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, a value representing the job's priority and a job class. For carrying out these jobs a number of machines are available. These machines can be split up into a number of disjoint machine classes. For each combination of a job class and a machine class it is known whether or not it is allowed to assign a job in the job class to a machine in the machine class. Furthermore the jobs must be carried out in a non-preemptive way and each machine can be carrying out at most one job at the same time. Within this setting one can ask for a feasible schedule for all jobs or, if such a schedule does not exist, for a feasible schedule for a subset of the jobs of maximum total value. In this paper we present a complete classification of the computational complexity of two classes of combinatorial problems related this operational job scheduling problem.

Keywords: Scheduling, combinatorial analysis, computational complexity, fixed job intervals

1. Introduction

Between the time of arrival at and the consecutive time of departure from the main airport in the Netherlands an aircraft must be inspected before being allowed to take off again. If the stochastic elements are neglected, then such an inspection can be seen as a job being of a certain job class and having a fixed start time, a fixed finish time and a value representing its priority. The job class of a job is determined by the type of the corresponding aircraft. Examples of job classes are B747, A310, DC10, etc. 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 consequently each job could be assigned to each of them. However, this is not a practical solution, as a governmental rule states that each ground engineer is allowed to have two licenses at most.

In principle all the jobs must be carried out. However, jobs with a low priority may also be carried out at the next stop of the corresponding aircraft at an airport. Hence an operational job scheduling problem that has to be solved within the described context is the following: Suppose that the number of available engineers and the combination of licenses of each engineer are known. Then the operational job scheduling problem asks for a feasible schedule for all the jobs, taking into account the licenses of the engineers, or, if a feasible schedule for all the jobs does not exist, for a feasible schedule for a subset of the jobs of maximum total value.

In this paper we study two classes of mathematical problems related to this operational job scheduling problem. We derive a complete classification of the computational complexity of these classes of problems. In a forthcoming publication we will address the tactical counterpart of these problems, asking for the required number and licenses of the engineers. In order to follow the literature on job scheduling the engineers are addressed as 'machines' in the remainder of this paper.

2. Problem definition

Suppose there are J jobs that have to be carried out. Job j requires continuous processing in the interval (s_j, f_j), has a job class a_j and a value v_j. As it is sufficient to know for each pair of jobs whether or not they are overlapping, it may be assumed without loss of generality that the start and finish times of the jobs are integers (see Kolen, Lenstra and Papadimitriou [11]). Therefore job j can be represented as a quadruple of integers (s_j, f_j, a_j, v_j). Whenever the values of the jobs are not relevant, for example in a feasibility problem, then they are omitted from the notation. Throughout this paper we use the notation A for the number of different job classes and we write T for the integer $\max\{f_j \mid j = 1, 2, \ldots, J\}$.

The jobs must be carried out in a non-preemptive way by a number of machines, each of which is continuously available. The machines can be split up into a number of disjoint machine classes. The number of machine classes is denoted by C. For each combination of a job class and a machine class the feasibility of assigning a job in the job class to a machine in the machine class is represented in the $A \times C$ zero/one matrix L, where the rows of L correspond to the job classes and the columns correspond to the machine classes. The interpretation of the matrix L is as follows: $L_{ac} = 1$ if and only if it is allowed to assign jobs in job class a to machines in machine class c. Examples of matrices L that are studied explicitly in the sequel are the following:

$$L_0 = (1), \quad L_1 = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}, \quad L_2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{pmatrix},$$

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

For example, the matrix L_3 represents a situation with 2 job classes and 3 machines classes and where for $a = 1, 2$ a job in job class a can be carried out by the machines in the machine classes a and 3. Let the $A \times C$ zero/one matrix L be given. Then the problem of finding a feasible schedule for all the jobs is called Class Scheduling with respect to L, or CS(L) for short. This problem can be stated more formally as follows:

Instance of CS(L).

– Jobs (s_j, f_j, a_j) for $j = 1, 2, \ldots, J$, that have to be carried out.

– C nonnegative integers $M_1, M_2, \ldots, M_C$, representing the numbers of machines available in the machine classes $1, 2, \ldots, C$, respectively.

Question Does there exist a feasible non-preemptive schedule for all the jobs?

Note that the matrix L is part of the definition of the problem CS(L) and does not belong to the instances of the problem CS(L). Therefore we have defined a whole class of combinatorial problems, indexed by the zero/one matrices L.

The problem of finding a feasible schedule for a subset of jobs of maximum total value is called Maximum Class Scheduling with respect to L, abbreviated as MCS(L). The formal definition of this problem is as follows:

Instance of MCS(L).

– Jobs (s_j, f_j, a_j, v_j) for $j = 1, 2, \ldots, J$, that have to be carried out.

– C nonnegative integers $M_1, M_2, \ldots, M_C$, representing the numbers of machines available in the machine classes $1, 2, \ldots, C$, respectively.

Question. What is the maximum total value of a subset of the jobs for which there exists a feasible non-preemptive schedule?

Again the matrix L does not belong to the instances of the problem MCS(L) but is part of the definition of the problem MCS(L). Hence we have defined a whole class of combinatorial problems again. Note that for any matrix L the problem MCS(L) is a generalization of the problem CS(L).

A further generalization of the problems CS(L) and MCS(L) is given by Arkin and Silverberg [1]. In [1] the assumption is also that all the jobs have a fixed start time and a fixed finish time. However the assumption in [1] with respect to the feasibility of the assignment of a specific job to a specific machine is different from ours. In [1] for each job j a subset of the machines V_j is given and it is assumed that job j can be carried out by the machines in V_j only. The objective is to find a feasible schedule for all the jobs. In [1] it is shown that this problem is NP-complete. However an algorithm based on Dynamic Programming is presented that can be used for solving in $O(J^{M+1})$ time the problem of finding a feasible schedule for a subset of the jobs of maximum total value. Here M denotes the total number of available machines. This implies that, if the number of machines is fixed, then this optimization problem can be solved in polynomial time. Note that in our context for job j with job class a_j the set V_j can be defined as follows: $V_j = \{$machines belonging to those machine classes c with $L_{a_j c} = 1\}$. Hence for a given matrix L an instance of the problems CS(L) or MCS(L) can be seen as an instance of the problems in [1]. Therefore the $O(J^{M+1})$ time algorithm in [1] can be applied for solving CS(L) and MCS(L) also.

Similar problems were studied by several authors. Dondeti and Emmons [4] and [5] prove that CS(L_1) and CS(L_2) can be solved in polynomial time. They also show that for the matrices L_1 and L_2 the tactical problem asking for the required number of machines in each machine class can be solved in polynomial time. Furthermore in [5] the computational complexity that is introduced by limited availability of the machines is studied. Kolen, Lenstra and Papadimitriou [11] describe a situation with A job classes and A machines classes, where a job in job class a can be carried out by a machine in machine class c if and only if $a \leq c$. This corresponds in our setting to an $A \times A$ upper triangular matrix L. Complexity results are presented which show that for this kind of matrices L the problem CS(L) is NP-complete if and only if $A \geq 3$. Kolen and Kroon [10] study a situation with A job classes, where each machine has exactly B licenses. One of the main results in [10] is that the feasibility problem is NP-complete if and only if $1 < B < A$. Problems similar to the problems CS(L) and MCS(L) but in the context of the assignment of classes to classrooms at high schools and universities are studied by Carter and Tovey [2].

All the mentioned problems are generalizations of the well known Fixed Job Scheduling Problem (FSP) which has been studied by many authors such as Dantzig and Fulkerson [3], Gertsbakh and Stern [8] and Gupta, Lee and Leung [9]. FSP is a feasibility problem where it is allowed to assign each of the jobs to each of the machines. That is, in our setting FSP can be described as CS(L_0). The following result, which will be used in the sequel, gives a necessary and sufficient condition for the existence of a feasible schedule for an instance of FSP:

Lemma 1. *For an instance of* FSP *a feasible schedule for all the jobs exists if and only if the maximum job overlap is less than or equal to the number of available machines.*

Here the job overlap in the interval $(t-1, t)$, denoted by R_t, is defined as follows:

$$R_t = |\{ j \mid s_j \leq t-1 \text{ and } t \leq f_j \}|.$$

That is, the job overlap is equal to the number of jobs that should be carried out in the time interval $(t-1, t)$. The maximum job overlap R is defined as:

$$R = \max\{ R_t \mid t = 1, 2, \ldots, T \}.$$

As FSP is equivalent to CS(L_0) and calculating the maximum job overlap is easy, the problem CS(L_0) can be solved in polynomial time. More specific, it is well known that the problem CS(L_0) can be solved in O(J log J) time. This is optimal as follows from a result of Fredman and Weide [6]. It is also well known (Arkin and Silverberg [1], Kolen Lenstra and Papadimitriou [11]) that MCS(L_0) can be solved by finding a Minimum Cost Flow in a network with O(J) nodes and O(J) arcs. Hence MCS(L_0) can be solved in polynomial time also.

3. Preliminary remarks with respect to L

In the Sections 4 and 5 a complete classification of the computational complexity of the problems CS(L) and MCS(L) will be presented. An important tool in this classification is Lemma 2, which relates the complexities of two problems to each other. This Lemma can be proved by applying an evident reduction and therefore the proof is omitted.

Lemma 2. *Let L_x and L_y be $A_x \times C_x$ and $A_y \times C_y$ zero/one matrices respectively. Suppose that, after the application of row or column permutations, L_x is a submatrix of L_y. Then the following statements hold:*
- *If* CS(L_x) *is* NP-*complete, then* CS(L_y) *is* NP-*complete too.*
- *If* MCS(L_x) *is* NP-*hard, then* MCS(L_y) *is* NP-*hard too.*

Furthermore it is clear that a complete row or a complete column of zeros in the matrix L is not interesting. A row of zeros corresponds to a job class that cannot be assigned to any machine class and a column of zeros corresponds to a machine class that is not allowed to carry out any jobs. Two identical rows or two identical columns in the matrix L are not interesting either. If, for example, the matrix L contains two identical rows, then this means that the corresponding job classes are equivalent and hence it makes no sense to distinguish between them. Finally it is clear that applying row or column permutations to the matrix L does not change the complexity of CS(L). Hence if the matrix L can be written as

$$L = \begin{pmatrix} L_x & 0 \\ 0 & L_y \end{pmatrix},$$

by applying row or column permutations, then the complexity of the problem CS(L) is completely determined by the complexity of the problems CS(L_x) and CS(L_y). More specific: the problem CS(L) can be solved in polynomial time if and only if both problems CS(L_x) and CS(L_y) can be solved in polynomial time. If at least one of the problems CS(L_x) or CS(L_y) is NP-complete, then CS(L) is NP-complete also. Analogous results also hold with respect to MCS(L). These considerations lead to the following definition:

Definition. 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

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

If none of these conditions is satisfied, then the matrix L is called *irreducible*.

Note that the matrices L_0, L_1, L_2, L_3, L_4 and L_5 which were defined in Section 2 are all irreducible. Now it is clear that a classification of the computational complexity of the problems is the set $\{\text{CS}(L) \mid L$ is an irreducible zero/one matrix$\}$ implies a classification of the computational complexity of the problems in the set $\{\text{CS}(L) \mid L$ is any zero/one matrix$\}$ and that the same holds with respect to the problems MCS(L). Therefore we can restrict our attention to irreducible matrices L. In the following sections the NP-completeness results are established by a reduction from Numerical Three Dimensional Matching (= N3DM). This problem is defined as follows (Garey and Johnson [7]):

Instance of N3DM.

- Integers t, d and a_i, b_i, and c_i for $i = 1, 2, \ldots, t$, satisfying the following relations:

$$\sum_{i=1}^{t} (a_i + b_i + c_i) = td,$$

and

$$0 < a_i,\ b_i,\ c_i < d \quad \text{for } i = 1, 2, \ldots, t.$$

Question. Is it possible to find permutations ρ and σ of $\{1, 2, \ldots, t\}$ which are such that:

$$a_i + b_{\rho(i)} + c_{\sigma(i)} = d \text{ for } i = 1, 2, \ldots, t.$$

It is well known that N3DM is NP-complete. Therefore any problem in NP that is more general than N3DM is NP-complete also.

4. Complexity of MCS(L)

In this section we derive a classification of the computational complexity of the problems MCS(L). In this classification we pay attention to irreducible matrices L only. The main theorem in the classification is Theorem 3 which reads as follows:

Theorem 3. *Let L be an irreducible zero/one matrix. Then the problem* MCS(L) *is* NP-*hard if and only if L has at least* 2 *columns.*

The ‘only if’-part of this theorem can be proved by noting that, if the irreducible matrix L has only 1 column, then it must be the matrix L_0. As MCS(L_0) can be solved in polynomial time, the ‘only if’-part of Theorem 3 is clear. On the other hand, if the irreducible matrix L has at least 2 columns, then, after applying row or column permutations, it contains the matrix L_1 as a submatrix. Therefore the 'if'-part of Theorem 3 follows from the following theorem and Lemma 2:

Theorem 4. MCS(L_1) *is* NP-*hard.*

Proof. This theorem is proved by a reduction from N3DM. Let I_1 be an instance of N3DM containing integers t, d and a_i, b_i and c_i for $i = 1, 2, \ldots, t$. Then the following quantities can be defined:

$$A_i := i \quad \text{for } i = 1, 2, \ldots, t,$$

$$B_j := t + j \quad \text{for } j = 1, 2, \ldots, t,$$

$$X_{ij} := 2t + (i-1)t + j \quad \text{for } i,\ j = 1, 2, \ldots, t,$$

$$S := t^2 + 2t,$$
$$T := S + 2d + 1.$$

Note that the numbers A_i, B_j and X_{ij} are all different for i, $j = 1, 2, \ldots, t$. Now an instance I_2 of MCS(L_1) can be constructed as follows: there are t machines in machine class 1 and $t^2 - t$ machines in machine class 2. Furthermore the following jobs have to be carried out:
Jobs in job class 1:

$$(0,\ A_i) \quad \text{for } i = 1, 2, \ldots, t,$$

$$(S + d - c_k,\ T) \quad \text{for } k = 1, 2, \ldots, t,$$

Jobs in job class 2:

$$t-1 \text{ times } (0,\ B_j) \quad \text{for } j = 1, 2, \ldots, t,$$

$$(A_i,\ X_{ij}) \quad \text{for } i,\ j = 1, 2, \ldots, t,$$

$$(B_j,\ X_{ij}) \quad \text{for } i,\ j = 1, 2, \ldots, t,$$

$$(X_{ij},\ S + a_i + b_j) \quad \text{for } i,\ j = 1, 2, \ldots, t,$$

$$(S + a_i + b_j,\ T - 1) \quad \text{for } i,\ j = 1, 2, \ldots, t,$$

$$t^2 - t \text{ times } (T-1,\ T).$$

The value of a job is equal to it's length (that is: $v_j = f_j - s_j$). Figure 1 gives an example of an instance I_2 of MCS(L_1) that has been constructed from an instance I_1 of N3DM. In this example I_1 is a yes-instance and in I_2 there exists a feasible schedule for a subset of the jobs which is such that all the machines are occupied uninterruptedly. Now we will show for the general case that I_1 is a yes-instance if and only if in I_2 the maximum total value of the jobs for which a feasible schedule exists is equal to t^2T. The latter is equivalent to the fact that the schedule is such that all the machines are occupied uninterruptedly during the interval (0, T).

Suppose that there exists a feasible schedule for a subset of the jobs which is such that all the machines are occupied uninterruptedly during the interval (0, T). As there are exactly t^2 jobs that start at 0, all of them must be carried out. Therefore the job (0, A_i) in job class 1 is carried out by a machine in machine class 1 and it is followed directly by one of the jobs (A_i, X_{ij}) in job class 2. This statement holds for $i = 1, 2, \ldots, t$. Now machine class 1 is occupied completely in the interval (0, 1). Therefore the $t - 1$ jobs (0, B_j) in job class 2 are carried out by machines in machine class 2 and each of them is followed directly by one of the jobs (B_j, X_{ij}) in job class 2. This statement holds for $j = 1, 2, \ldots, t$.

As there are exactly t^2 jobs that finish at T, all of them must be carried out. The jobs ($S + d - c_k$, T) in job class 1 are carried out by machines in machine class 1. Therefore machine class 1 is occupied completely in the interval ($T - 1$, T) and hence the jobs ($T - 1$, T) in job class 2 are carried out by machines in machine class 2.

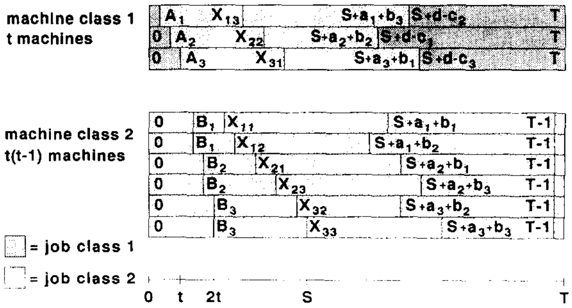

Figure 1

As all the machines must be occupied uninterruptedly and the jobs $(X_{ij}, S+a_i+b_j)$ are the only jobs which have an overlap with the interval $(S, S+1)$, all these jobs must be carried out. Therefore we get schedules of the form

$$(0, A_i)(A_i, X_{ij})(X_{ij}, S+a_i+b_j)$$

on machines in machine class 1, where each i occurs exactly once and we get schedules of the form

$$(0, B_j)(B_j, X_{ij})(X_{ij}, S+a_i+b_j)$$

on machines in machine class 2, where each j occurs exactly $t-1$ times. Hence among the jobs $(X_{ij}, S+a_i+b_j)$ that are carried out by machines in machine class 1 each i and each j occurs exactly once.

Furthermore on machines in machine class 1 a job $(X_{ij}, S+a_i+b_j)$ in job class 2 is followed by a job $(S+d-c_k, T)$ in job class 1 in such a way that $S+a_i+b_j=S+d-c_k$, which also means that $a_i+b_j+c_k=d$. So if we define $\rho(i)=j$ and $\sigma(i)=k$, whenever job $(X_{ij}, S+a_i+b_j)$ is combined with job $(S+d-c_k, T)$, then ρ and σ are the required permutations for I_1.

Conversely, given a feasible solution for I_1, the construction can be reversed to find in I_2 a feasible schedule for a subset of the jobs of total value equal to t^2T, which is clearly optimal. Note that the jobs $(S+a_i+b_j, T-1)$ in job class 2 can be used to fill the 'gap' on the machines in machine class 2 completely. Hence I_1 is a yes-instance if and only if in I_2 the maximum total value of the jobs for which a feasible schedule exists is equal to t^2T. It follows that MCS(L_1) is NP-hard. □

Figure 1 gives an example of an instance I_2 of MCS(L_1) that has been constructed from an instance I_1 of N3DM as described in the proof of Theorem 4.

In I_1 we have $t=3$ and $d=12$. Furthermore we have $(a_1, a_2, a_3)=(3, 4, 6)$, $(b_1, b_2, b_3)=(5, 3, 7)$ and $(c_1, c_2, c)=(5, 2, 1)$. Note that I_1 is a yes-instance, as

$$a_1+b_3+c_2=a_2+b_2+c_1=a_3+b_1+c_3=12.$$

Figure 1 shows that there exists a feasible schedule for a subset of the jobs which is such that all the machines are occupied uninterruptedly.

5. Complexity of CS(L)

The aim of this section is to provide a complete classification of the computational complexity of the problems CS(L). In this classification we pay attention to irreducible matrices L only, as the general case can be deduced from this easily using the remarks in Section 3. The main theorem in this classification is Theorem 5 which reads as follows:

Theorem 5. *Let L be an irreducible zero/one matrix. Then the problem* CS(L) *is* NP-*complete if and only if L has at least* 3 *columns.*

The proof of this theorem is rather long and has been split up into several parts therefore. In order to prove Theorem 5 we adopt the following strategy: First the problem CS(L_2) will be shown to be polynomially solvable. As L_2 is, apart from row or column permutations, the greatest irreducible matrix with at most 2 columns, this proves the 'only if'-part of Theorem 5. This part of the proof of Theorem 5 can be found in subsection 5.1. Next the problems CS(L_3), CS(L_4) and CS(L_5) will be shown to be NP-complete. This part of the proof of Theorem 5 is described in subsection 5.2. Finally it will be shown that the problem CS(L) can be reduced from at least one of the problems CS(L_3), CS(L_4) or CS(L_5), if the matrix L is irreducible and has at least 3 columns. This proves the 'if'-part of Theorem 5. This part of the proof of Theorem 5 is described in subsection 5.3.

5.1. Complexity of CS(L_2)

In this subsection we will prove that CS(L_2) can be solved in polynomial time. Let I be an instance of CS(L_2) consisting of a set of jobs in 3 job classes and two integers M_1 and M_2 representing the numbers of available machines in the two machine classes. Using the result of Lemma 1 necessary conditions for the existence of a feasible schedule can be derived as follows. If we define R_{at} to be the job overlap of the jobs in job class a in the interval $(t-1, t)$ for $a = 1, 2, 3$, and $t = 1, 2, \dots, T$, then these conditions can be stated as

$$R_{1t} \le M_1 \quad \text{for } t = 1, 2, \dots, T, \tag{1}$$

$$R_{2t} \le M_2 \quad \text{for } t = 1, 2, \dots, T, \tag{2}$$

$$R_{1t} + R_{2t} + R_{3t} \le M_1 + M_2 \quad \text{for } t = 1, 2, \dots, T. \tag{3}$$

In the remainder of this subsection we assume that these conditions hold. Note that for $t = 1, 2, \dots, T$, we can add $M_1 + M_2 - R_{1t} - R_{2t} - R_{3t}$ dummy jobs in job class 3 in the interval $(t-1, t)$ to the set of jobs of I without changing the feasibility of I. Therefore the inequalities (3) can be replaced by the equalities (4):

$$R_{1t} + R_{2t} + R_{3t} = M_1 + M_2, \quad \text{for } t = 1, 2, \dots, T. \tag{4}$$

Next we construct the following network N with O(J) nodes and O(J) arcs. N contains the nodes $\{t \mid t = 0, 1, 2, \dots, T\}$. For each job j in job class 1 we have an arc from node s_j to node f_j. The lower and upper capacity of such an arc are both equal to 1. For each job j in job class 3 we have an arc from node s_j to node f_j. The lower capacity of such an arc is equal to 0 and the upper capacity is equal to 1. Now the following lemma holds:

Theorem 6. *There exists a feasible schedule for I if and only if there exists a compatible flow of* M_1 *units from node* 0 *to node T in the network N.*

Proof. Suppose that there exists a feasible schedule for I. Then we can assign 1 unit of flow to each arc in N corresponding to a job that is carried out by a machine in machine class 1. The equalities (4) guarantee that all machines, and in particular the machines in machine class 1, are continuously busy. Therefore the jobs that are carried out by one machine in machine class 1 represent 1 unit of flow from node 0 to node T. As there are M_1 machines in machine class 1, this gives a flow of M_1 units from node 0 to node T. It is evident that this flow is compatible with all the capacity constraints.

Conversely, suppose that there exists a compatible flow of M_1 units from node 0 to node T. Then a feasible schedule for I can be constructed as follows: the jobs corresponding to an arc containing 1 unit of flow are carried out by machines in machine class 1 and the other jobs are carried out by machines in machine class 2. Note that the definition of N guarantees that each job in job class 1 is carried out by a machine in machine class 1 and that none of the jobs in job class 2 is carried out by a machine in machine class 1. For $t = 1, 2, \ldots, T$, we have M_1 units of flow across the cut $\{(0, \ldots, t-1), (t, \ldots, T)\}$. Therefore the job overlap of the jobs that have to be carried out by the machines in machine class 1 is equal to M_1. Hence Lemma 1 implies that there exists a feasible schedule for these jobs. According to the equalities (4), the job overlap of the jobs that are *not* carried out by the machines in machine class 1 is equal to M_2. Hence Lemma 1 implies that there exists a feasible schedule for these jobs on machine class 2. □

Lemma 6 implies that CS(L_2) can be solved in polynomial time, as the existence of a compatible flow of M_1 units from node 0 to node T in the *directed* network N can be checked in polynomial time (see Gondran and Minoux [9]). The same result has also been obtained by Dondeti and Emmons [5]. They use almost the same construction.

5.2. *Complexity of CS(L_3), CS(L_4) and CS(L_5)*

In this subsection we prove the NP-completeness of the problems CS(L_3), CS(L_4) and CS(L_5). The NP-completeness of these problems is proved by a reduction from N3DM. In Theorem 10 the problem CS(L_3) is proved to be NP-complete. As the proof of this theorem is rather complex, it is preceded by several lemmas. These lemmas deal with a scheduling problem which is a generalization of FSP. In this scheduling problem the job classes and the machine classes are the same as in CS(L_3). However the machines are not available over the whole interval $(0, T)$. An instance of this scheduling problem is given by:

- Nonnegative integers T_1, T_2 and t.
- $2t$ integers $B_1, E_1, B_2, E_2, \ldots, B_t, E_t$, with

$$T_1 = B_1 < E_1 < B_2 < E_2 < \cdots < B_t < E_t = T_2.$$

- $t^2 - t$ machines in machine class 1. For $j = 1, 2, \ldots, t$, there are $t-1$ machines available in the interval (T_1, E_j).
- $t^2 - t$ machines in machine class 2. For $j = 1, 2, \ldots, t$, there are $t-1$ machines available in the interval (B_j, T_2).
- t machines in machine class 3. These machines are available in the interval (T_1, T_2).
- t^2 jobs (T_1, f_j) in job class 1. This set of jobs can be partitioned into t subsets $V_1, V_2, \ldots, V_t$ each one containing t jobs, where for $j = 1, 2, \ldots, t$, a job in V_j has a finish time greater than B_j and less than E_j.
- t^2 jobs (s_j, T_2) in job class 2. This set of jobs can be partitioned into t subsets $W_1, W_2, \ldots, W_t$ each one containing t jobs, where for $j = 1, 2, \ldots, t$, a job in W_j has a start time greater than B_j and less than E_j.

The question is of course, whether or not there exists a feasible schedule for all the jobs. Figure 2 gives an example of an instance of this scheduling problem with $t = 3$. The instance in Figure 2 is a yes-instance.

Suppose that there exists a feasible schedule for all the jobs. As all the jobs in job class 1 are overlapping, all these jobs are scheduled on different machines. As the maximum number of available machines in machine class 1 is $t^2 - t$, exactly t of the jobs in job class 1 are scheduled on machines in machine class 3. The same is true for the jobs in job class 2. Therefore each machine in machine class 3 is carrying out exactly 1 job in job class 1 and 1 job in job class 2. More details with respect to the distribution of the jobs are derived in the Lemmas 7, 8 and 9:

Lemma 7. *Let p be fixed with $1 \le p \le t$. Then the following statements hold*:

(i) *the number of jobs in the sets V_q with $p \le q$ which are scheduled on a machine in machine class 3 is greater than or equal to $t - p + 1$,*

(ii) *the number of jobs in the sets W_q with $p \le q$ which are scheduled on a machine in machine class 3 is greater than or equal to $t - p + 1$.*

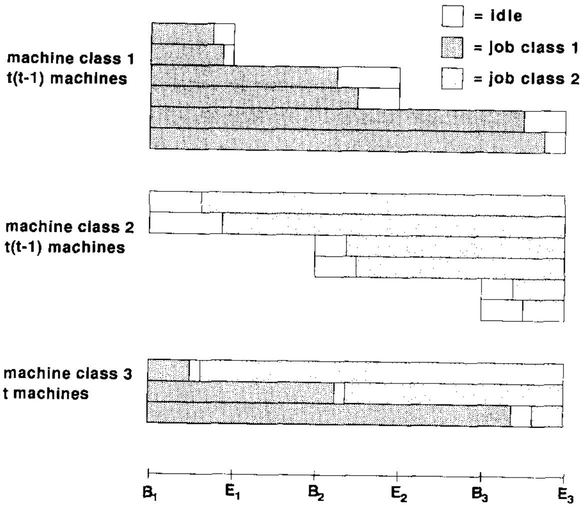

Figure 2

Proof. (i) There are $t(t-p+1)$ jobs in the sets V_q with $p \le q$. As the finish times of these jobs are greater than E_{p-1}, these jobs cannot be scheduled on a machine that is available in (T_1, E_r) with $r < p$. Therefore at most $(t-1)(t-p+1)$ machines in machine class 1 are available for carrying out these jobs. Hence at least

$$t(t-p+1)-(t-1)(t-p+1)=t-p+1$$

jobs in the sets V_q with $p \le q$ are scheduled on a machine in machine class 3.

(ii) As the finish time of a job in V_q exceeds the start time of a job in W_r if $q > r$, a job in V_q can only be combined with a job in W_r on a machine in machine class 3 if $q \le r$. □

Lemma 8. *Let p be fixed with $1 \le p \le t$. Then the following statements hold*:

(i) *the number of jobs in the sets W_q with $q \le p$ which are scheduled on a machine in machine class* 3 *is greater than or equal to p_j*

(ii) *the number of jobs in the sets V_q with $q \le p$ which are scheduled on a machine in machine class* 3 *is greater than or equal to p.*

Proof. (i) There are tp jobs in the sets W_q with $q \le p$. As these jobs start before B_{p+1}, these jobs cannot be scheduled on a machine that is available in (B_r, T_2) with $p < r$. Therefore at most $(t-1)p$ machines in machine class 2 are available for carrying out these jobs. Hence at least $tp-(t-1)p=p$ jobs in the sets W_q with $q \le p$ are scheduled on the set of machines in machine class 3.

(ii) As the finish time of a job in V_r exceeds the start time of a job in W_q if $r > q$, a job in W_q can only be combined with a job in V_r on a machine in machine class 3 if $r \le q$. □

Lemma 9. *Let p be fixed with $1 \le p \le t$. Then the following statements hold*:

(i) *exactly* 1 *job in V_p is scheduled on a machine in machine class* 3;

(ii) *exactly* 1 *job in W_p is scheduled on a machine in machine class* 3;

(iii) *the job in V_p that is scheduled on a machine in machine class* 3 *and the job in W_p that is scheduled on a machine in machine class* 3 *are scheduled on the same machine.*

Proof. (i) According to Lemma 7 at least $t-p+1$ jobs in the sets V_q with $p \le q$ are scheduled on a machine in machine class 3. According to Lemma 8 at least p jobs in the sets V_q with $q \le p$ are scheduled on a machine in machine class 3. Hence if non of the jobs in V_p would be scheduled on a machine in machine class 3, then at least $p + (t - p + 1) = t + 1$ jobs in the sets V_q with $q \neq p$ would be scheduled on machine class 3. However this is impossible as all the jobs in job class 1 are overlapping. Therefore at least 1 job in V_p is scheduled on a machine in machine class 3. There cannot be more than 1 such job as otherwise the job overlap of the jobs on machine class 3 would be greater than t.

(ii) This statement can be proved in the same way as statement (i).

(iii) This statement follows from the fact that a job in V_q can only be combined with a job in W_r if $q \leq r$. Therefore the job in V_t that is scheduled on a machine in machine class 3 is combined with a job in W_t, the job in V_{t-1} that is scheduled on a machine in machine class 3 is combined with a job in W_{t-1}, etc. □

Theorem 10. CS(L_3) *is* NP-*complete.*

Proof. This theorem is proved by a reduction from N3DM. Hence let I_1 be an instance of N3DM containing integers t, d and a_i, b_i and c_i for $i = 1, 2, \ldots, t$. Then the following quantities can be defined:

$$X_i := t + i \quad \text{for } i = 1, 2, \ldots, t,$$

$$Y_j := 3t + 3dj \quad \text{for } j = 1, 2, \ldots, t,$$

$$Z_k := 3t + 3d + 3dt + k \quad \text{for } k = 1, 2, \ldots, t,$$

$$T := 6t + 3d + 3dt.$$

Now an instance I_2 of CS(L_3) can be constructed as follows: there are $t^2 - t$ machines in machine class 1, $t^2 - t$ machines in machine class 2 and t machines in machine class 3. Furthermore the following jobs have to be carried out:

Jobs in job class 1:

$$t^2 \text{ times } (T - t, T),$$

$$(t, X_i) \quad \text{for } i = 1, 2, \ldots, t,$$

$$t - 1 \text{ times } (0, X_i) \quad \text{for } i = 1, 2, \ldots, t,$$

$$t - 1 \text{ times } (Y_j + 2d, T - t) \quad \text{for } j = 1, 2, \ldots, t,$$

$$(X_i, Y_j + a_i + b_j) \quad \text{for } i, j = 1, 2, \ldots, t.$$

Jobs in job class 2:

$$t^2 \text{ times } (0, t),$$

$$(Z_k, T - t) \quad \text{for } k = 1, 2, \ldots, t,$$

$$t - 1 \text{ times } (Z_k, T) \quad \text{for } k = 1, 2, \ldots, t,$$

$$t - 1 \text{ times } (t, Y_j) \quad \text{for } j = 1, 2, \ldots, t,$$

$$(Y_j + d - c_k, Z_k) \quad \text{for } j, k = 1, 2, \ldots, t.$$

Figure 3 gives an example of an instance I_2 of CS(L_3) that has been constructed from an instance I_1 of N3DM. In this example both I_1 and I_2 are yes-instances. Now we will show for the general case that I_1 is a yes-instance if and only if I_2 is a yes-instance.

Suppose that I_2 is a yes-instance. Then the jobs $(0, t)$ in job class 2 fill up the machine classes 2 and 3 in the interval $(0, t)$. Therefore the jobs $(0, X_i)$ in job class 1 are carried out by machines in machine class 1. This implies that the jobs (t, X_i) in job class 1 must be carried out by machines in machine class 3 and that the jobs (t, Y_j) in job class 2 are carried out by machines in machine class 2.

The jobs $(T - t, T)$ in job class 1 fill up the machine classes 1 and 3 in the interval $(T - t, T)$. Therefore the jobs (Z_k, T) in job class 2 are carried out by machines in machine class 2. This implies that the jobs $(Z_k, T - t)$ in job class 2 are carried out by machines in machine class 3 and that the jobs $(Y_j + 2d, T - t)$ in job class 1 are carried out by machines in machine class 1.

All the jobs $(X_i, Y_j + a_i + b_j)$ in job class 1 have an overlap with the interval $(2t, 3t)$. Therefore these jobs are all carried out by different machines. This implies that t of these jobs are carried out by machines in machine class 3 and that the other $t^2 - t$ are carried out by machines in machine class 1. An analogous reasoning can be applied to show that t of the jobs $(Y_j + d - c_k, Z_k)$ in job class 2 are carried out by machines in machine class 3 and that the other $t^2 - t$ are carried out by machines in machine class 2.

In the interval $(0, 2t)$ the total available machine time is equal to $2t(2t^2 - t)$. In the same interval the total required machine time is also equal to $2t(2t^2 - t)$. Therefore in the interval $(0, 2t)$ all the machines are occupied uninterruptedly. This holds in particular for the machines in machine class 3.

As the jobs (t, X_i), among which each i occurs exactly once, are carried out by machines in machine class 3, among the jobs $(X_i, Y_j + a_i + b_j)$ on machine class 3 each i occurs exactly once. (5)

In the same way it can be proved that, among the jobs $(Y_j + d - c_k, Z_k)$ on machine class 3 each k occurs exactly once. (6)

Now the following definitions are used:

$$T_1 := 3t + 3d,$$

$$T_2 := 3t + 2d + 3dt,$$

$$B_j := Y_j \quad \text{for } j = 1, 2, \ldots, t,$$

$$E_j := Y_j + 2d \quad \text{for } j = 1, 2, \ldots, t,$$

$$V_j := \{(X_i, Y_j + a_i + b_j) \mid i = 1, 2, \ldots, t\} \quad \text{for } j = 1, 2, \ldots, t,$$

$$W_j := \{(Y_j + d - c_k, Z_k) \mid k = 1, 2, \ldots, t\} \quad \text{for } j = 1, 2, \ldots, t,$$

These definitions create an instance of the situation that was described in the Lemmas 7, 8 and 9. This can be checked easily. From Lemma 9 and (5) it follows that among the jobs $(X_i, Y_j + a_i + b_j)$ on machine class 3 each i and each j occurs exactly once. Therefore a permutation ρ of the set $\{1, 2, \ldots, t\}$ can be defined by: $\rho(i) := j$ if the job $(X_i, Y_j + a_i + b_j)$ is scheduled on a machine in machine class 3.

Furthermore, using Lemma 9 and (6) it can be concluded that among the jobs $(Y_j + d - c_k, Z_k)$ on machine class 3 each j and each k occurs exactly once. Therefore a permutation τ of the set $\{1, 2, \ldots, t\}$ can be defined by: $\tau(j) := k$ if the job $(Y_j + d - c_k, Z_k)$ is scheduled on machine class 3. Hence if the permutation σ of the set $\{1, 2, \ldots, t\}$ is defined by: $\sigma := \tau \circ \rho$, then it follows from Lemma 9 that for each i job $(X_i, Y_{\rho(i)} + a_i + b_{\rho(i)})$ is combined with job $(Y_{\rho(i)} + d - c_{\sigma(i)}, Z_{\sigma(i)})$ on a machine in machine class 3.

Therefore it follows that $a_i + b_{\rho(i)} \leq d - c_{\sigma(i)}$, for $i = 1, 2, \ldots, t$. As $\Sigma_{i=1}^{t}(a_i + b_i + c_i) = td$, it follows that these inequalities are equalities in fact. Therefore the permutations ρ and σ are the required permutations of the set $\{1, 2, \ldots, t\}$ and hence I_1 is a yes-instance.

Conversely, given a feasible solution for I_1, the construction can be reversed to find a feasible schedule for I_2. It follows that I_1 is a yes-instance if and only if I_2 is a yes-instance. As it is clear that CS(L_3) belongs to NP and it is well known that N3DM is NP-complete, it follows that CS(L_3) is NP-complete also. □

Figure 3 gives an example of an instance I_2 of CS(L_3) that has been constructed from an instance I_1 of N3DM. In I_1 we have $t = 3$ and $d = 6$. Furthermore we have $(a_i, a_2, a_3) = (1, 2, 2)$, $(b_1, b_2, b_3) = (3, 1, 2)$ and $(c_1, c_2, c_3) = (4, 2, 1)$. Note that I_1 is a yes-instance, as

$$a_1 + b_2 + c_1 = a_2 + b_1 + c_3 = a_3 + b_3 + c_2 = 6$$

The figure shows that there exists a feasible schedule for all the jobs.

Next we will prove that the problems CS(L_4) and CS(L_5) are NP-complete. These proofs are essentially the same as the proof of Theorem 10, especially the combinatorial argument. Therefore the details of the proofs have been omitted, as they can be found in the proof of Theorem 10. Although the definition of the jobs might have been simplified we have defined the jobs as much as possible the same as in the proof of Theorem 10.

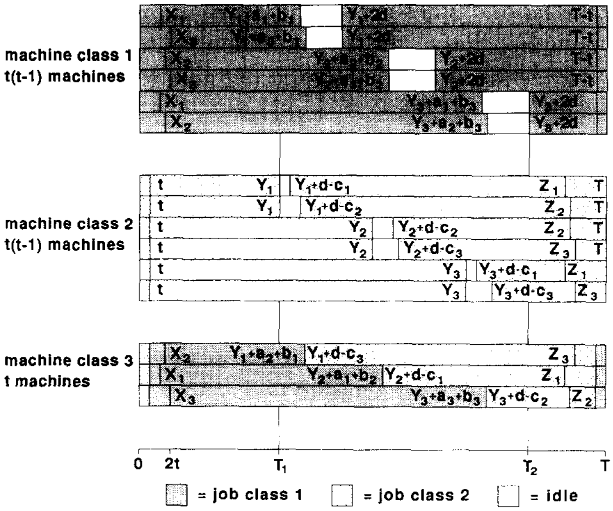

Figure 3

Theorem 11. CS(L_4) *is* NP-*complete.*

Proof. The notation in this proof is the same as in the proof of Theorem 10. Let I_1 be an instance of N3DM containing integers t, d and a_i, b_i and c_i for $i = 1, 2, \ldots, t$. Then an instance I_2 of CS(L_4) can be constructed as follows: There are $t^2 - t$ machines in machine class 1, $t^2 - t$ machines in machine class 2 and t machines in machine class 3. Furthermore the following jobs have to be carried out:
Jobs in job class 1:

$$t - 1 \text{ times } (0, X_i), \quad \text{for } i = 1, 2, \ldots, t,$$

$$t - 1 \text{ times } (Y_j + 2d, T), \quad \text{for } j = 1, 2, \ldots, t,$$

Jobs in job class 2:

$$t - 1 \text{ times } (0, Y_j), \quad \text{for } j = 1, 2, \ldots, t,$$

$$t - 1 \text{ times } (Z_k, T), \quad \text{for } k = 1, 2, \ldots, t,$$

Jobs in job class 3:

$$(0, X_i), \quad \text{for } i = 1, 2, \ldots, t,$$

$$(Z_k, T), \quad \text{for } k = 1, 2, \ldots, t,$$

$$(X_i, Y_j + a_i + b_j), \quad \text{for } i, j = 1, 2, \ldots, t,$$

$$(Y_j + d - c_k, Z_k), \quad \text{for } j, k = 1, 2, \ldots, t.$$

Now we will prove that I_1 is a yes-instance if and only if I_2 is a yes-instance. Suppose that I_2 is a yes-instance. Then it is clear that the jobs in job class 1 are carried out by machines in machine class 1 and that the jobs in job class 2 are carried out by machines in machine class 2. Now the machines classes 1 and 2 are completely occupied at the beginning and at the end of the interval (0, T) and therefore the jobs (0, X_i) and (Z_k, T) in job class 3 are carried out by machines in machine class 3. As all the jobs (X_i, $Y_j + a_i + b_j$) in job class 3 have an overlap in the interval ($2t$, $3t$) with all the jobs (0, Y_j) in job class 2, it follows that the jobs (X_i, $Y_j + a_i + b_j$) are carried out by machines in the machine classes 1 and 3. As all the jobs ($Y_j + d - c_k$, Z_k) in job class 3 have an overlap in the interval ($3t + 3dt + 2d$, $3t + 3dt + 3d + 1$) with all the jobs ($Y_j + 2d$, T) in job class 1, it follows that the jobs ($Y_j + d - c_k$, Z_k) are carried out by machines in the machine classes 2 and 3. Therefore the distribution of the 'essential' jobs in this instance of CS(L_4) is the same as the distribution of the 'essential' jobs in the instance of CS(L_3) that was created in the proof of Theorem 10. Hence the same combinatorial argument as in the proof of Theorem 10 can be used to show that I_1 is a yes-instance.

Conversely, if I_1 is a yes-instance, then the construction can be reversed to find a feasible schedule for I_2. Therefore I_1 is a yes-instance if and only if I_2 is a yes-instance. As is it clear that CS(L_4) belongs to NP and it is well known that N3DM is NP-complete, it follows that CS(L_4) is NP-complete too. □

Theorem 12. CS(L_5) *is* NP-*complete.*

Proof. This theorem has been proved by Kolen, Lenstra and Papadimitriou [12]. However, this theorem can also be proved by applying the proof of Theorem 11. One only has to note that in the proof of Theorem 11 the machines in machine class 1 are completely occupied at the beginning and at the end of the interval (0, T). Hence although in CS(L_5) it is allowed to schedule the jobs in job class 2 on machine classes 1 or 2, they are scheduled on machine class 2 only. Therefore the situation is analogous to the situation in the proof of Theorem 11. Further details of this proof are omitted. □

5.3. Completion of the proof of Theorem 5.

In this subsection the proof of Theorem 5 is completed by linking the results of the Theorems 6, 10, 11 and 12 together. The required connections between these results are provided by the Lemmas 2, 13 and 14.

In this subsection the following notation is used: For job class a the set C_a denotes the set of machine classes that can be used for carrying out jobs in job class a. That is: $C_a = \{c \mid L_{ac} = 1\}$.

Lemma 13. *Let L be an irreducible $A \times C$ matrix with the property that for all job classes a and b we have:*

$$(C_a \cap C_b \neq \emptyset) \Rightarrow ((C_a \subset C_b) \vee (C_b \subset C_a)).$$

Then the following statements hold:

(i) *for every job class a there is a job class a' with $C_{a'} \subset C_a$ and $|C_{a'}| = 1$,*

(ii) *there exists a job class a with $C_a = \{1, 2, \ldots, C\}$.*

Proof. (i) Let a be a job class. Then either we have

$$\forall b((C_a \cap C_b = \emptyset) \vee (C_a \subset C_b))$$

or we have

$$\exists b((C_a \cap C_b \neq \emptyset) \wedge (C_a \not\subset C_b)).$$

The latter condition implies $C_b \subset C_a$ and in that case we can continue with the job class b. By repeating this argument several times we will find a job class a' with

$$\forall b((C_{a'} \cap C_b = \emptyset) \vee (C_{a'} \subset C_b)).$$

Now we will show that a' contains exactly one element. Choose c, $d \in C_{a'}$ and let b be a job class. If $C_{a'} \cap C_b = \emptyset$, then $L_{bc} = L_{bd} = 0$, and if $C_{a'} \subset C_b$, then $L_{bc} = L_{bd} = 1$. Hence the columns c and d are identical. As the matrix L is irreducible it follows that $c = d$ and therefore $|C_{a'}| = 1$.

(ii) Analogously as above we can find a job class a with

$$\forall b((C_a \cap C_b = \emptyset) \vee (C_b \subset C_a)).$$

However, if there exists a job class b with $C_a \cap C_b = \emptyset$, then the matrix L can be transformed into block diagonal form with at least two blocks and hence it is reducible. This contradiction implies that $\forall b(C_b \subset C_a)$. As L does not contain a complete column of zeros, it follows that $C_a = \{1, 2, \ldots, C\}$. □

Lemma 14. *If L is an irreducible matrix with at least 3 columns, then after a suitable permutation of the rows and the columns, it contains at least one of the matrices L_3, L_4 or L_5 as a submatrix.*

Proof. The following two cases can be distinguished:

Case 1. There exist job classes a and b such that

$$(C_a \cap C_b \neq \emptyset) \wedge (C_a \not\subset C_b) \wedge (C_b \not\subset C_a).$$

Now the columns of L can be rearranged in such a way that the rows of L corresponding to the job classes a and b are as follows:

$$\begin{array}{c} a \\ b \end{array}\begin{pmatrix} 0 & 0 & \cdots & 0 & 1 & 1 & \cdots & 1 & 0 & 0 & \cdots & 0 & 1 & 1 & \cdots & 1 \\ 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 1 & 1 & \cdots & 1 & 1 & 1 & \cdots & 1 \end{pmatrix}.$$

In this case, after a suitable permutation of the rows and the columns, L contains L_3 as a submatrix.

Case 2. For all job classes a and b we have

$$(C_a \cap C_b \neq \emptyset) \Rightarrow ((C_a \subset C_b) \vee (C_b \subset C_a)).$$

According to Lemma 13 there is a job class r that can be carried out by all the machine classes. Now there must be at least two other job classes a and b, as otherwise L would contain 2 identical columns. Again 2 cases can be distinguished:

Case 2a. $C_a \cap C_b = \emptyset$. According to Lemma 13 there are job classes a' and b', which can be carried out by exactly 1 machine class and such that $C_{a'} \subset C_a$ and $C_{b'} \subset C_b$. Now the columns of L can be rearranged in such a way that the rows corresponding to the job classes a', b' and r are as follows:

$$\begin{array}{c} a' \\ b' \\ r \end{array}\begin{pmatrix} 1 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & 0 & \cdots & 0 \\ 1 & 1 & 1 & 1 & \cdots & 1 \end{pmatrix}.$$

In this case, after a suitable permutation of the rows and the columns, L contains L_4 as a submatrix.

Case 2b. $C_a \cap C_b \neq \emptyset$. This implies $(C_a \subset C_b) \vee (C_b \subset C_a)$. It may be assumed that $C_a \subset C_b$. Then the columns of L can be reordered in such a way that the rows corresponding to a, b and r are as follows:

$$\begin{array}{c} a \\ b \\ r \end{array}\begin{pmatrix} 1 & 1 & \cdots & 1 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ 1 & 1 & \cdots & 1 & 1 & 1 & \cdots & 1 & 0 & 0 & \cdots & 0 \\ 1 & 1 & \cdots & 1 & 1 & 1 & \cdots & 1 & 1 & 1 & \cdots & 1 \end{pmatrix}.$$

In this case, after a suitable permutation of the rows and the columns, L contains L_5 as a submatrix. □

Now the results of the Lemmas 2 and 14 provide the required connections between the results of the Theorems 6, 10, 11 and 12. This completes the proof of Theorem 5 and the classification of the computational complexity of the problems CS(L).

6. Concluding remarks

In this paper the problems CS(L) and MCS(L) which appear in a natural way in the aircraft maintenance process at an airport were described in a formal way. We have presented a complete classification of the computational complexity of these problems. In this paper we did not look at optimization methods for calculating optimal or satisfying solutions. These aspects are a topic for further research.

Until now we have focused mainly on the operational questions that should be answered within the aircraft maintenance process. However in Section 1 we mentioned already that tactical questions with respect to the required number and qualifications of the engineers should be answered also. These tactical problems, which we have called Class Design with respect to the matrix L, or CD(L) for short, can be described more formally in terms of jobs and machines as follows:

Instance of CD(L).
- Jobs (s_j, f_j, a_j) for $j = 1, 2, \ldots, J$, that have to be carried out.

Question. How to choose the numbers of machines in each of the machine classes in order to guarantee the existence of a feasible schedule for all the jobs with a minimum total number of machines?

It is clear that these problems can be seen as generalizations of FSP also. In a forthcoming publication we will present a classification of the computational complexity of the problems CD(L), more or less analogous to the classification that we have presented in this paper.

References

[1] Arkin, E.M., and 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, February 1989, University of Toronto, Department of Industrial Engineering, and *Operations Research*, to appear.

[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] 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.

[5] Dondeti, V.R., and Emmons, H., "Interval scheduling with processors of two types", *Operations Research*, to appear.

[6] 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.

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

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

[9] Gondran, M., and Minoux, M., *Graphs and Algorithms*. Wiley-Interscience, New York, 1984.

[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] Kolen, A.W.J., and Kroon, L.G., "On the computational complexity of (A, L)-Class Scheduling", Report Nr. 34 of the Rotterdam School of Management, 1989.

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