European Journal of Operational Research 64 (1993) 138–151
North-Holland

Theory and Methodology

On the computational complexity of (Maximum) Shift 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 December 1990

Abstract: In this paper we consider a generalization of the Fixed Job Scheduling Problem (FSP) which appears in a natural way in the aircraft maintenance process at an airport. A number of jobs has to be carried out, where the main attributes of a job are: a fixed start time, a fixed finish time and a value representing the priority of the job. For carrying out these jobs a number of machines is available. These machines are available in specific time intervals (shifts) only. A job can be carried out by a machine only if the interval between the start time and the finish time of the job is a subinterval of the shift of the machine. 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 jobs of maximum total value. In this paper a classification of the computational complexity of two classes of combinatorial problems related to these questions is presented.

Keywords: Air transportation; Computational complexity; Fixed 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. Such an inspection can be seen as a job with a fixed start time, a fixed finish time and a value representing the priority of the inspection. The start time and the finish time of an inspection 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 time and the finish time of each inspection.

The inspections have to be carried out by a number of ground engineers. These ground engineers are available at the airport in specific time intervals (shifts) only. In principle an inspection can be carried out by a ground engineer only if the interval between the start time and the finish time of the inspection is a subinterval of the shift of the engineer.

In the practical situation at the airport still other restrictions should be satisfied. For example, an engineer is allowed to carry out a specific

inspection only if he has a license for the corresponding aircraft type. The problems that are introduced by the licenses of the engineers have been studied by Kolen and Kroon [12]. In this paper we will therefore neglect the restrictions that are imposed by the licenses by assuming that all engineers have licenses for all aircraft types.

Suppose that a reliable estimate of the workload in the near future has been made. Then both tactical and operational questions should be answered in order to realize a sufficiently high service level. Examples of such questions are the following:

- How many shifts are required and how to choose the begin and the end times of the shifts appropriately?
- How many engineers should be available in each shift?
- How many engineers are required in total and how to assign these engineers to the shifts?
- How to assign the inspections to the available engineers if both the begin and the end times of the shifts and the assignments of the engineers to the shifts have been determined?

In this paper we fill focus on the latter (operational) job scheduling problem. The remainder of this paper is organized as follows: In Section 2 we give a formal definition of a class of feasibility problems and a class of optimization problems related to this operational job scheduling problem. In Section 3 some preliminary remarks with respect to the computational complexity of the problems in these classes are made. In Section 4 a detailed analysis of the computational complexity of the class of optimization problems is presented and in Section 5 the same is done for the class of feasibility problems. We finish with some concluding remarks in Section 6. In order to follow the standard literature on job scheduling, in the remainder of this paper the inspection will be addressed as 'jobs' and the engineers will be addressed as 'machines'.

2. Problem definition

Suppose that J jobs have to be carried out, where the main attributes of job j are a fixed (rational) start time s_j, a fixed (rational) finish time f_j and a value v_j representing the priority of the job. The jobs have to be carried out in a non-preemptive way by a number of parallel machines. The total number of machines is denoted by M. These machines are available in specific time intervals (shifts) only. The number of shifts is denoted by Z and shift z has a fixed (rational) begin time b_z and a fixed (rational) end time e_z.

The number of machines available in shift z is denoted by M_z. Note that, if $M_z > J$, then scheduling the jobs in shift z is easy, because in this case each job in shift z can get its own machine. Hence throughout this paper it is assumed that $M_z \leqslant J$ for $z = 1, 2, \ldots, Z$. The shift in which machine m is available is denoted by $z(m)$. Job j can be carried out by machine m if and only if the interval (s_j, f_j) is a subinterval of the shift of machine m which is denoted by $(b_{z(m)}, e_{z(m)})$.

After this description of the relevant notation, the problems Shift Class Scheduling (SCS) and Maximum Shift Class Scheduling (MSCS) can be defined. SCS is a feasibility problem asking whether or not a feasible schedule exists for all jobs. MSCS is an optimization problem asking for a subset of jobs of maximum total value for which a feasible schedule exists. SCS can be defined more formally as follows.

Instance of SCS.

- J jobs (s_j, f_j) to be carried out.
- Z triples (b_z, e_z, M_z), for $z = 1, 2, \ldots, Z$ representing shift z and satisfying $M_z \leqslant J$.

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

SCS is studied by Kolen, Lenstra and Papadimitriou [13], who call this problem Interval Scheduling. This problem is proved to be NP-complete by a straightforward reduction from Circular Arc Colouring (see Garey and Johnson [8]). However, in [13] it is also shown that SCS can be solved in polynomial time if preemption of the jobs (only at the end of the shift intervals) is allowed.

If a feasible schedule for *all* jobs in an instance of SCS does not exist, then it is interesting to find a subset of jobs of maximum total value for which a feasible schedule does exist. This problem is called MSCS and is defined more formally as follows.

Instance of MSCS.

– J jobs (s_j, f_j, v_j) to be carried out.

– Z triples (b_z, e_z, M_z), for $z = 1, 2, \ldots, Z$ representing shift z and satisfying $M_z \leqslant J$.

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

MSCS is NP-hard, as it is evident that MSCS is a generalization of SCS. A further generalization of SCS and MSCS is studied by Arkin and Silverberg [1]. In [1] the assumption is that all 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 machines W_j is given and it is assumed that job j can be carried out by the machines in W_j only. The objective is to find a feasible schedule for all jobs. In [1] it is shown that this problem is NP-complete. Furthermore, a Dynamic Programming formulation is presented that can be used for solving in $O(J^{M+1})$ time the more general problem of finding a subset of jobs of maximum total value for which a feasible schedule exists. This implies that, if the number of machines is fixed beforehand, then this optimization problem can be solved in polynomial time. Note that in our context for job j the set W_j can be defined by

$$W_j = \{m \mid b_{z(m)} \leqslant s_j \text{ and } f_j \leqslant e_{z(m)}\}.$$

Hence SCS and MSCS can be seen as special cases of the problem in [1]. Therefore the $O(J^{M+1})$ time algorithm in [1] can be applied for solving SCS and MSCS also. However, it should be noted that this result in only interesting from a theoretical point of view, due to the size of the state space in the Dynamic Programming formulation.

Note that in the definitions of SCS and MSCS the set of shifts belongs to the instances of the problems. This leaves open the complexity of the scheduling problems when the number of shifts and the begin and end time of each shift are known in advance, as is the case at the operational level in the aircraft maintenance process at the airport. In order to study these problems, let S be a set containing Z shift intervals. That is, S is a set $\{(b_z, e_z) \mid z = 1, 2, \ldots, Z\}$. Then the problems Shift Class Scheduling with respect to S, or SCS(S) for short, and Maximum Shift Class Scheduling with respect to S, abbreviated to MSCS(S), can be defined. For example, for the given set of shifts S the problem SCS(S) is defined as follows.

Instance of SCS(S).

– J jobs (s_j, f_j) to be carried out.

– Z integers M_z, for $z = 1, 2, \ldots, Z$ representing the number of machines in shift z and satisfying $M_z \leqslant J$.

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

Now the set of shifts does not belong to the instances of SCS(S) but to the type of the problem. Therefore we have defined a whole class of job scheduling problems, indexed by the sets of shifts S. Note that, if the integers M_z representing the numbers of machines in each of the Z shifts are also taken away from the instances of SCS(S), then the results of Arkin and Silverberg [1] show that the obtained problem can be solved in polynomial time. MSCS(S) can be defined in a similar way as SCS(S).

It is evident that the computational complexities of the problems SCS(S) and MSCS(S) depend on the size and the structure of the set of shifts S. In Section 4 a complete classification of the computational complexity of the problems MSCS(S) is presented and in Section 5 a classification of the computational complexity of a large subset of the problems SCS(S) is presented. It turns out that it is not difficult to create a set of shifts S such that SCS(S) is NP-complete or such that MSCS(S) is NP-hard, but polynomially solvable cases occur frequently as well.

SCS(S) and MSCS(S) are closely related to the job scheduling problems that are introduced by taking into account the licenses of the engineers. These problems have been studied extensively by Kolen and Kroon [12]. Special cases of these problems were considered by Carter and Tovey [2], by Dondeti and Emmons [4], [5], and by Kolen, Lenstra and Papadimitriou [13].

3. Preliminary remarks

All problems that were mentioned in Section 2 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 [9] and Gupta, Lee and Leung [10].

In FSP the jobs have a fixed start time and a fixed finish time and the machines are continuously available. Therefore FSP is equivalent to SCS(S) if the set of shifts S contains only one shift. The following result 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 jobs exists* $\Leftrightarrow$ *the maximum job overlap is less than or equal to the number of available machines.*

Here for a given set of jobs the job overlap at instant t, denoted by D_t, and the maximum job overlap, denoted by D, are defined as follows:

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

As calculating the maximum job overlap is easy, FSP can be solved in polynomial time. More specific, it is well known that FSP can be solved in $O(J \log(J))$ time on a sequential processor. This is optimal, as follows from a result of Fredman and Weide [6]. However, FSP can be solved in $O(\log(J))$ time by $O(J)$ parallel processors (Kindervater [11]).

The Maximum Fixed Job Scheduling Problem MFSP is the optimization version of FSP. That is, in MFSP all machines are continuously available and the problem is to find a feasible schedule for a subset of jobs of maximum total value. As is shown by Arkin and Silverberg [1], by Kolen, Lenstra and Papadimitriou [13] and by Kroon [14], MFSP can be solved by finding a Minimum Cost Flow of M units of flow in a directed network with $O(J)$ nodes and $O(J)$ arcs. Consequently, MFSP can be solved in polynomial time.

For a given set of shifts S the shift overlap at instant t, denoted by A_t, and the maximum shift overlap, denoted by A, are defined in the same way as the (maximum) job overlap. That is:

- $A_t = |\{z \mid b_z < t < e_z\}|$
- $A = \max\{A_t \mid -\infty < t < \infty\}$

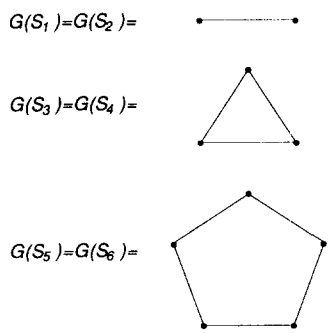

Figure 1. Example of sets of shifts S and graphs $G(S)$

$S_1 = \{(0, 6), (1, 5)\}$
$S_2 = \{(0, 6), (1, 7)\}$
$S_3 = \{(0, 12), (1, 11), (2, 7)\}$
$S_4 = \{(0, 12), (1, 13), (2, 14)\}$
$S_5 = \{(0, 5), (0, 8), (1, 2), (4, 8), (6, 7)\}$
$S_6 = \{\{0, 2\}. (0, 4), (1, 4), (1, 5), (3, 5)\}$

Furthermore, the undirected graph $G(S)$, which will play an important role in the remainder of this paper, is defined as follows:

- $G(S)$ contains one node for every shift.
- Two nodes in $G(S)$ are connected $\Leftrightarrow$
  - the corresponding shifts are overlapping and
  - the corresponding shifts have different begin times and different end times.

An example of sets of shifts that will be analyzed in the remainder of this paper and the corresponding graphs $G(S)$ is represented in Figure 1.

The aim of this paper is to provide an analysis of the computational complexity of the problems SCS(S) and MSCS(S). In this paper we make the assumption that P $\neq$ NP. This assumption is justified by the fact that it simplifies the notation in several places and by the fact that it is in accordance with the general opinion on the relation of P and NP. Important tools in the analysis of the computational complexity of the problems SCS(S) and MSCS(S) are Lemmas 2 and 3 which relate the complexities of two problems to each other. As these lemmas can be proved by elementary reductions, the proofs are omitted.

Lemma 2. *Let* S_x *and* S_y *be two sets of shifts. If* S_x *is a subset of* S_y, *then the following statements hold*:

- $\text{SCS}(S_x) \propto \text{SCS}(S_y)$.
- $\text{MSCS}(S_x) \propto \text{MSCS}(S_y)$.

Here the notation $X \propto Y$ means that a polynomial reduction exists from problem X to problem

Y. We refer to Garey and Johnson [8] for a definition of this concept. It turns out that the complexity of the problems SCS(S) and MSCS(S) is completely determined by the overlap structure of the shifts and that the absolute begin times and the absolute end times of the shifts are of minor importance. This is expressed in Lemma 3 which uses the following definition. If $f: R \to R$ is a strictly increasing bijection and S is a set of shifts, then S^f denotes another set of shifts which is defined as follows:

$$S^f := \{(f(b_z), f(e_z)) \mid z = 1, 2, \ldots, Z\}.$$

Now it is evident that the following relation on the pairs of sets of shifts is an equivalence relation.

$S_x \sim S_y \Leftrightarrow \exists f: R \to R$, f is a strictly increasing bijection and $S_y = S_x^f$.

Lemma 3 shows that for all sets of shifts S in one equivalence class of this relation the computational complexity of the problem SCS(S) is the same and that an analogous result also holds for MSCS(S).

Lemma 3. *Let S be a set of shifts and let $f: R \to R$ be a strictly increasing bijection. Then the following statements hold:*
- SCS(S) $\propto$ SCS(S^f).
- MSCS(S) $\propto$ MSCS(S^f).

Note that, if f is a strictly increasing bijection, then the same holds for f^{-1}. It follows that we can restrict ourselves to the representatives of the equivalence classes of the equivalence relation. Furthermore, the following lemma will be used several times in the Sections 4 and 5, where the computational complexity of the job scheduling problems is analyzed.

Lemma 4. *Let S be a set of shifts containing a pair of shifts x and y with $e_x = b_y$. Then a set of shifts S' exists with the following properties:*
- SCS(S) $\propto$ SCS(S').
- *$e'_x \neq b'_y$ for all shifts x and y in S'.*
- *The graphs $G(S)$ and $G(S')$ are isomorphic.*

Proof. Let us first suppose that S is such that exactly one time instant t exists such that $t = e_x = b_y$ for some pair of shifts x and y in S. Then the set of shifts S' is defined as follows:

$$b'_z = \begin{cases} b_z & \text{if } b_z < t \\ b_z + 1 & \text{if } b_z \geqslant t \end{cases} \quad \text{for } z = 1, 2, \ldots, Z.$$

$$e'_z = \begin{cases} e_z & \text{if } e_z \leqslant t \\ e_z + 1 & \text{if } e_z > t \end{cases} \quad \text{for } z = 1, 2, \ldots, Z.$$

As it was assumed that t is the only time instant such that $t = e_x = b_y$ for some pair of shifts x and y in S, it follows that $e'_x \neq b'_y$ for all shifts x and y in S'. Furthermore, overlap of shifts and equality of begin times of shifts and equality of end times of shifts are preserved by the transformation. Therefore the graphs $G(S)$ and $G(S')$ are isomorphic. Moreover, if I is an instance of SCS(S) containing J jobs (s_j, f_j) and Z integers M_z, then an instance I' of SCS(S') is constructed as follows:

$$s'_j = \begin{cases} s_j & \text{if } s_j < t \\ s_j + 1 & \text{if } s_j \geqslant t \end{cases} \quad \text{for } j = 1, 2, \ldots, J.$$

$$f'_j = \begin{cases} f_j & \text{if } f_j \leqslant t \\ f_j + 1 & \text{if } f_j > t \end{cases} \quad \text{for } j = 1, 2, \ldots, J.$$

$$M'_z = M_z \quad \text{for } z = 1, 2, \ldots, Z.$$

It is not difficult to see that I is a yes-instance of SCS(S) if and only if I' is a yes-instance of SCS(S'). This proves Lemma 4 in the case that exactly one time instant t exists such that $t = e_x = b_y$ for some pair of shifts x and y in S. The general case can be proved by repeating the above argument as often as necessary. □

It is evident that an analogous result also holds with respect to the problems MSCS(S). The Lemmas 7 and 12 in the following sections are established by a reduction from an adapted version of Numerical Three Dimensional Matching or N3DM for short. This problem is defined as follows:

Instance of N3DM.

- A positive integer t and $3t$ rational numbers a_i, b_i and c_i for $i = 1, 2, \ldots, t$ satisfying $0 < a_i, b_i, c_i < 1$ and

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

Question. Is it possible to find permutations ρ and σ of $\{1, 2, \ldots, t\}$ such that: $a_i + b_{\rho(i)} + c_{\sigma(i)} = 1$ for $i = 1, 2, \ldots, t$?

It is well known that N3DM is NP-complete (see Garey and Johnson [8]). Therefore any problem in NP that is more general than N3DM is NP-complete also. The proofs of the Lemmas 7 and 12 are illustrated by the following instance of N3DM with $t = 3$:

$(a_1, a_2, a_3) = (\frac{1}{8}, \frac{1}{4}, \frac{3}{8})$,

$(b_1, b_2, b_3) = (\frac{1}{8}, \frac{1}{4}, \frac{1}{2})$

and

$(c_1, c_2, c_3) = (\frac{1}{4}, \frac{3}{8}, \frac{3}{4})$.

Note that this instance is a yes-instance of N3DM, as

$$a_1 + b_1 + c_3 = a_2 + b_3 + c_1 = a_3 + b_2 + c_2 = 1.$$

4. Complexity results for MSCS(S)

The classification of the computational complexity of the problems MSCS(S) is a complete classification, whereas the classification of the computational complexity of the problems SCS(S) is only a classification of a (large) subset of the problems SCS(S). Therefore we will first present the classification of the computational complexity of the problems MSCS(S). This classification is expressed in Theorem 5 which reads as follows:

Theorem 5. MSCS(S) *can be solved in polynomial time* $\Leftrightarrow$ *the graph* $G(S)$ *consists of isolated nodes only.*

Theorem 5 is proved in several steps. First we will prove in Lemma 6 that MSCS(S) can be solved in polynomial time if $G(S)$ consists of isolated nodes only. Conversely, if $G(S)$ contains at least one edge, then S contains a subset of two overlapping shifts with different begin times and different end times and belonging to the same equivalence class as S_1 or S_2. Here the sets of shifts S_1 and S_2 are the same as in Section 3. As it will be shown in Lemmas 7 and 8 that MSCS(S_1) and MSCS(S_2) are NP-hard, it follows that MSCS(S) is NP-hard by applying the Lemmas 2 and 3.

Lemma 6. *If the graph* $G(S)$ *consists of isolated nodes only, then* MSCS(S) *can be solved in polynomial time.*

Proof. This lemma is proved by establishing the following reduction: MSCS(S) α MFSP. As MFSP can be solved in polynomial time, the result follows. Let the set of shifts S be such that the graph $G(S)$ consists of isolated nodes only. Note that, according to Lemma 4, it may be assumed that

$$e_x \neq b_y \text{ for all shifts } x \text{ and } y \text{ in } S. \qquad (*)$$

Let I_1 be an instance of MSCS(S) containing J jobs (s_j, f_j, v_j) to be carried out and Z integers M_z, for $z = 1, 2, \ldots, Z$ representing the number of machines in shift z and satisfying $M_z \leqslant J$. Now an instance I_2 of MFSP is constructed as follows: The number of machines is equal to $M = \sum_{z=1}^{Z} M_z$. Furthermore, let

$$B = \min\{b_z, s_j \mid z = 1, 2, \ldots, Z \text{ and } j = 1, 2, \ldots, J\} - 1,$$

and let

$$E = \max\{e_z, f_j \mid z = 1, 2, \ldots, Z \text{ and } j = 1, 2, \ldots, J\} + 1.$$

Finally, let V be a big number with $V > \sum_{j=1}^{J} v_j$. Note that an appropriate value for V can be calculated in polynomial time. Then the following jobs have to be carried out:

- J jobs (s_j, f_j, v_j) from the instance I_1.
- M_z times the dummy job (B, b_z, V) for $z = 1, 2, \ldots, Z$.
- M_z times the dummy job (e_z, E, V) for $z = 1, 2, \ldots, Z$.

As $M_z \leqslant J$ for $z = 1, 2, \ldots, Z$ and Z is fixed, the size of I_2 is polynomial in the size of I_1. Let V_1 denote the maximum value of a feasible schedule in I_1 and let V_2 denote the maximum value of a feasible schedule in I_2. We will show that $V_2 = V_1 + 2\ V\ M$. It is easy to see that $V_2 \geqslant V_1 + 2\ V\ M$, because an optimal solution for I_1 can be transformed into a solution for I_2 by addition of all dummy jobs.

Now we will show that $V_1 \geqslant V_2 - 2\ V\ M$ by showing that in any optimal solution for I_2 each machine is carrying out two dummy jobs (B, b_x, V) and (e_y, E, V) which are such that the interval (b_x, e_y) corresponds to some shift interval. Therefore we obtain a solution for I_1 by deletion of all dummy jobs.

From the definition of the number V it follows that in any optimal solution for I_2 all dummy jobs are scheduled. Therefore each machine is carrying out exactly one dummy job (B, b_x, V) and exactly one dummy job (e_y, E, V). The dummy jobs (B, b_x, V) and (e_y, E, V) are said to be *wrongly coupled* if they are carried out by the same machine and if the interval (b_x, e_y) does not correspond to any shift interval. Note that wrongly coupled jobs always come in pairs. Now suppose that dummy jobs exist which are wrongly coupled. Then the number b^* is defined as follows:

$$b^* = \min\{b_z \mid \text{one of the jobs } (B, b_z, V) \text{ is wrongly coupled}\}.$$

As one of the dummy jobs (B, b^*, V) is wrongly coupled, it follows that a shift y exists with $b_y = b^*$ for which one of the dummy jobs (e_y, E, V) is wrongly coupled also. Let this wrongly coupled job be coupled with a dummy job (B, b_x, V) corresponding to shift x. Then the assumption $(*)$ implies that we can not have $b_x = e_y$ and thus $b_x < e_y$.

Furthermore, the fact that the jobs (B, b_x, V) and (e_y, E, V) are wrongly coupled implies $b_x \neq b_y$ and $e_x \neq e_y$. As the graph $G(S)$ does not contain any edge, it follows that the shifts x and y are not overlapping. The fact that the shifts x and y are not overlapping implies that we have

$$b_x < e_x < b^* = b_y < e_y.$$

However, these inequalities contradict the definition of b^*, as the dummy job (B, b_x, V) is wrongly coupled. Therefore dummy jobs which are wrongly coupled do not exist. □

Lemma 7. MSCS(S_1) *is* NP-*hard.*

Proof. This lemma is proved by a reduction from N3DM. Hence let I_1 be an instance of N3DM containing the integer t and the rational numbers a_i, b_i and c_i for $i = 1, 2, \ldots, t$.

Now an instance I_2 of MSCS(S_1) is constructed as follows: in shift (0, 6) we have t machines and in the shift (1, 5) we have $t^2 - t$ machines. Furthermore, let for $i, j = 1, 2, \ldots, t$ the rational numbers A_i, B_j and X_{ij} be chosen in such a way that all these numbers are different and that for $i, j = 1, 2, \ldots, t$ we have $1 < A_i < B_j < 2 < X_{ij} < 3$. Then the jobs that have to be carried out in I_2 are the following:

- $(0, A_i)$ for $i = 1, 2, \ldots, t$.
- $t - 1$ times $(1, B_j)$ for $j = 1, 2, \ldots, t$.
- (A_i, X_{ij}) for $i, j = 1, 2, \ldots, t$.
- (B_j, X_{ij}) for $i, j = 1, 2, \ldots, t$.
- $(X_{ij}, 3 + a_i + b_j)$ for $i, j = 1, 2, \ldots, t$.
- $(3 + a_i + b_j, 5)$ for $i, j = 1, 2, \ldots, t$.
- $(4 - c_k, 6)$ for $k = 1, 2, \ldots, t$.

The value of each job is equal to the length of the job. That is, $v_j = f_j - s_j$. An example of an instance I_2 constructed from the instance I_1 of N3DM that was defined in Section 3 is presented in Figure 2. Note that in this example a schedule exists which is such that all machines are uninterruptedly busy. Now we will prove the following statement for the general case: I_1 is a yes-instance if and only if in I_2 the maximum value of a subset of jobs for which a feasible schedule exists is equal to $6t + 4(t^2 - t) = 4t^2 + 2t$, which is equal to the total available processing time of the machines.

Suppose that in I_2 the maximum value of a subset of jobs for which a feasible schedule exists is equal to $4t^2 + 2t$. As the value of each job is equal to its length and $4t^2 + 2t$ is equal to the total available processing time of the machines, it follows that all machines must be uninterrupted busy.

The jobs $(0, A_i)$ are the only jobs starting at the instant 0. Therefore all these jobs are carried out by the machines in the shift (0, 6). An analogous argument shows that the jobs $(1, B_j)$ are carried out by the machines in the shift (1, 5) and that the jobs $(4 - c_k, 6)$ are carried out by the machines in the shift (0, 6). For $i = 1, 2, \ldots, t$ the job $(0, A_i)$ is followed directly by one of the jobs (A_i, X_{ij}). For $j = 1, 2, \ldots, t$ the $t - 1$ jobs $(1, B_j)$ are followed directly by one of the jobs (B_j, X_{ij}).

As all machines must be uninterruptedly busy and the jobs $(X_{ij}, 3 + a_i + b_j)$ are the only jobs overlapping the instant 3, all these jobs must be carried out. Therefore, we get schedules of the form

$$(0, A_i)(A_i, X_{ij})(X_{ij}, 3 + a_i + b_j)$$

on machines in the shift (0, 6), where each i occurs exactly once and we get schedules of the form

$$(1, B_j)(B_j, X_{ij})(X_{ij}, 3 + a_i + b_j)$$

on machines in the shift (1, 5), where each j occurs exactly $t - 1$ times. Hence among the jobs $(X_{ij}, 3 + a_i + b_j)$ that are carried out by the machines in the shift (0, 6) each i and each j occurs exactly once.

Furthermore, on a machine in the shift (0, 6) a job $(X_{ij}, 3 + a_i + b_j)$ is followed by a job $(4 - c_k, 6)$ in such a way that $3 + a_i + b_j = 4 - c_k$, which means that $a_i + b_j + c_k = 1$. So if we define $\rho(i) = j$ and $\sigma(i) = k$, whenever job $(X_{ij}, 3 + a_i + b_j)$ is combined with job $(4 - c_k, 6)$, 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 a subset of jobs of I_2 of total value $4t^2 + 2t$ for which a feasible schedule exists, which is clearly optimal. Note that the jobs $(3 + a_i + b_j, 5)$ can be used to fill the gap on the machines in the shift (1, 5). As N3DM is NP-complete, it follows that MSCS(S_1) is NP-hard. □

Figure 2 gives an example of an instance I_2 constructed from the instance I_1 of N3DM that was defined in Section 3. The schedule in Figure 2 is an optimal schedule, as all machines are uninterruptedly busy.

Lemma 8. MSCS(S_2) *is* NP-*hard*.

Proof. This lemma is proved by a reduction from MSCS(S_1). Hence let I_1 be an instance of MSCS(S_1) containing J jobs (s_j, f_j, v_j) to be carried out and 2 integers M_1 and M_2 representing the numbers of machines in the shifts (0, 6) and (1, 5) respectively and satisfying $M_1 \leqslant J$ and $M_2 \leqslant J$. Furthermore, let V denote a big number satisfying $V > \sum_{j=1}^{J} v_j$. Then an instance I_2 of MSCS(S_2) is constructed as follows: The numbers of machines in the shifts (0, 6) and (1, 7) are equal to M_1 and M_2 respectively and the following jobs have to be carried out:

- J jobs (s_j, f_j, v_j) from the instance I_1.
- M_2 times the dummy job (5, 7, V).

As $M_2 \leqslant J$, the size of I_2 is polynomial in the size of I_1. From the definition of V it follows that in any optimal solution for I_2 all jobs (5, 7, V) are carried out. Hence if V_1 and V_2 represent the values of the optimal solutions for I_1 and I_2 respectively, then it is evident that $V_2 = V_1 + V M_2$ and that a close connection exists between the optimal solutions for I_1 and I_2. As MSCS(S_1) is NP-hard, it follows that MSCS(S_2) is NP-hard also. □

5. Complexity results for SCS(S)

In this section a classification of the computational complexity of the problems SCS(S) is presented. The classification is complete for the subset of problems in which the begin times of all shifts are different and the end times of all shifts are different.

We will start with a sufficient condition on the set of shifts S guaranteeing that SCS(S) can be solved in polynomial time. In the proof of this condition we use the following result of Dondeti

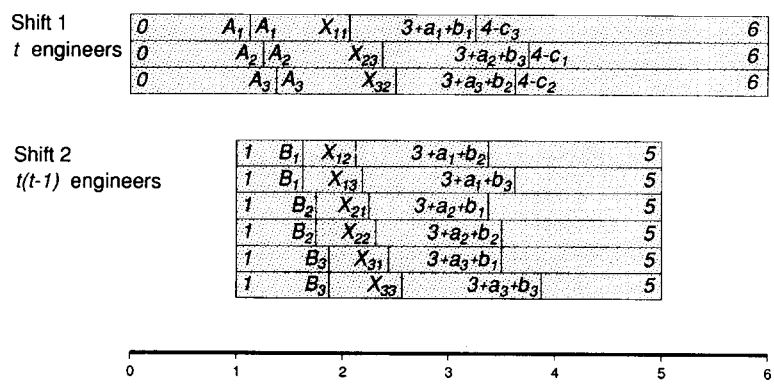

Figure 2. Optimal solution for an instance of MSCS(S_1)

and Emmons [5] and Kolen, Lenstra and Papadimitriou [13]: Suppose that J jobs have to be carried out, where each job has a fixed start and finish time and where each job belongs to one of two job classes. Suppose that M machines are continuously available, where each machine belongs to one of two machine classes. Furthermore, the jobs in job class 2 can be carried out by all machines, but the jobs in job class 1 can be carried out by the machines in machine class 1 only. The problem whether or not a feasible schedule for all jobs exists is called Class Scheduling, or CS for short. In [5] and [13] it is shown that a feasible schedule for an instance of CS corresponds to a compatible flow of M units in a directed network with $O(J)$ nodes and $O(J)$ arcs. As a consequence, CS can be solved in polynomial time.

Lemma 9. *If $G(S)$ is a bipartite graph, then* SCS(S) *can be solved in polynomial time.*

Proof. This lemma is proved by establishing the following reduction: SCS(S) α CS. As CS can be solved in polynomial time, the result follows. Let the set S be such that the graph $G(S)$ is bipartite. Note that, according to Lemma 4, it may be assumed that

$$e_x \neq b_y \text{ for all shifts } x \text{ and } y \text{ in } S. \qquad (**)$$

Let I_1 be an instance of SCS(S) containing J jobs (s_j, f_j) that have to be carried out and Z integers M_z, for $z = 1, 2, \ldots, Z$ representing the number of machines in shift z and satisfying $M_z \leqslant J$. As the graph $G(S)$ is bipartite, the set of shifts in I_1 can be split into two sets Z_1 and Z_2 such that all edges in $G(S)$ connect a node corresponding to a shift in Z_1 with a node corresponding to a shift in Z_2. Furthermore, let B denote the number $\min\{b_z, s_j \mid z = 1, 2, \ldots, Z \text{ and } j = 1, 2, \ldots, J\} - 1$ and let E denote the number $\max\{e_z, f_j \mid z = 1, 2, \ldots, Z \text{ and } j = 1, 2, \ldots, J\} + 1$.

Now an instance I_2 of CS is constructed as follows. The number of machines in machine class 1 is equal to $\sum_{z \in Z_1} M_z$ and the number of machines in machine class 2 is equal to $\sum_{z \in Z_2} M_z$. All machines are continuously available. Furthermore, the following jobs have to be carried out in the instance I_2:

Jobs in job class 1.

- M_z times the dummy job (B, b_z) for all z in Z_1.
- M_z times the dummy job (e_z, E) for all z in Z_1.

Jobs in job class 2.

- M_z times the dummy job (B, b_z) for all z in Z_2.
- M_z times the dummy job (e_z, E) for all z in Z_2.
- J jobs (s_j, f_j) from the instance I_1.

As $M_z \leqslant J$ for $z = 1, 2, \ldots, Z$ and Z is fixed, the size of I_2 is polynomial in the size of I_1. Now we will prove the following statement: I_1 is a yes-instance if and only if I_2 is a yes-instance. The 'only if' part of this statement is evident, because any feasible schedule for I_1 can be transformed into a feasible schedule for I_2 by addition of all dummy jobs in an obvious way.

In order to prove the 'if' part of the statement it is sufficient to prove that in any feasible schedule for I_2 the dummy jobs (B, b_x) and (e_y, E) are carried out by the same machine if and only if the interval (b_x, e_y) corresponds to some shift interval. A feasible solution for I_1 can be obtained then by deletion of all dummy jobs. The argument that we use here is similar to the argument that was used in the proof of Lemma 6.

Suppose a feasible schedule exists for I_2. Then it is evident that each machine is carrying out exactly one dummy job (B, b_x) and exactly one dummy job (e_y, E). The dummy jobs (B, b_x) and (e_y, E) are said to be *wrongly coupled* if they are carried out by the same machine and if the interval (b_x, e_y) does not correspond to any shift interval.

The dummy jobs (B, b_x) and (e_y, E) in job class 1 can be scheduled only on the machines in machine class 1. This implies that the machines in machine class 1 are occupied during the intervals $(B, B + 1)$ and $(E - 1, E)$. As a consequence, the dummy jobs (B, b_x) and (e_y, E) in job class 2 are scheduled on machines in machine class 2. This implies that a dummy job (B, b_x) in job class 1 and a dummy job (e_y, E) in job class 2 can not be carried out by the same machine. The same holds for a dummy job (B, b_x) in job class 2 and a dummy job (e_y, E) in job class 1. Now suppose that dummy jobs in job class 1 exist, which are

wrongly coupled. Then the number b^* is defined as follows:

$$b^* = \min\{b_z \mid \text{one of the jobs } (B, b_z) \text{ in job class 1 is wrongly coupled}\}.$$

As one of the dummy jobs (B, b^*) in job class 1 is wrongly coupled, it follows that a shift y in Z_1 with $b_y = b^*$ exists for which one of the dummy jobs (e_y, E) in job class 1 is wrongly coupled also. Let this wrongly coupled job be coupled with a dummy job (B, b_x) in job class 1 corresponding to shift x in Z_1. Then the assumption $(**)$ implies that we can not have $b_x = e_y$ and thus $b_x < e_y$. Furthermore, the fact that the jobs (B, b_x) and (e_y, E) in job class 1 are wrongly coupled implies $b_x \neq b_y$ and $e_x \neq e_y$. As the graph $G(S)$ does not contain any edge between any pair of shifts in Z_1, it follows that the shifts x and y are not overlapping. In the same way as in the proof of Lemma 6 we can conclude that:

$$b_x < e_x < b^* = b_y < e_y.$$

However, these inequalities contradict the definition of b^*, as the dummy job (B, b_x) in job class 1 is wrongly coupled. Therefore dummy jobs in job class 1 which are wrongly coupled do not exist. The same argument can be used to show that wrongly coupled dummy jobs in job class 2 do not exist either. □

Lemmas 6 and 9 are special cases of the results of Kroon [14], who relates the complexities of the problems MSCS(S) and SCS(S) to the node colouring number of the graph $G(S)$.

It is well known that Interval Graphs are triangulated, which means that in an Interval Graph any cycle of length greater than 3 has a chord (see Golumbic [7]). This property implies that the graph $G(S)$ does not contain any cycle if the maximum shift overlap of the shifts in S is less than or equal to 2. Hence we have the following corollary:

Corollary 10. *If the maximum shift overlap of the shifts in S is less than or equal to* 2, *then* SCS(S) *can be solved in polynomial time.*

Unfortunately the condition of Lemma 9 is a sufficient condition but it is not a necessary condition for SCS(S) to be solvable in polynomial time. This is illustrated by the set of shifts S_5 which was defined in Section 3. The graph $G(S_5)$ is a cycle of length 5, which is clearly not bipartite. However, SCS(S_5) can be solved in polynomial time by cutting the time axis at the instant $t = 3$. The resulting sets of shifts are S_5' and S_5''. The construction of the sets of shifts S_5' and S_5'' is illustrated by Figure 3.

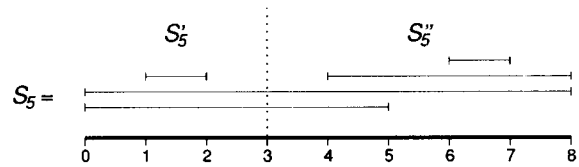

Figure 3. Cutting the time axis at the instant $t = 3$

$S_5' = \{(0, 3), (0, 3), (1, 2)\}$
$S_5'' = \{(3, 5), (3, 8), (4, 8), (6, 7)\}$

Note that S_5' contains only two different shifts in fact. Both SCS(S_5') and SCS(S_5'') can be solved in polynomial time by applying Lemma 9. If I is an instance of SCS(S_5), then it may be assumed that all jobs in I are contained in the interval (0, 8), as otherwise I is a no-instance. Now I can be transformed into an instance I' of SCS(S_5') and an instance I'' of SCS(S_5'') by defining the numbers of engineers in the shifts in I' and I'' to be equal to the numbers of engineers in the corresponding shifts in I and by 'cutting' the jobs overlapping the instant $t = 3$. Thus we obtain the following sets of jobs:

– The set of jobs in $I' =$

$$\{(s_j, f_j) \mid (s_j, f_j) \in I \text{ and } 0 \leqslant s_j < f_j \leqslant 3\}$$
$$\cup \{(s_j, 3) \mid (s_j, f_j) \in I \text{ and } 0 \leqslant s_j < 3 < f_j \leqslant 8\}.$$

– The set of jobs in $I'' =$

$$\{(s_j, f_j) \mid (s_j, f_j) \in I \text{ and } 3 \leqslant s_j < f_j \leqslant 8\}$$
$$\cup \{(3, f_j) \mid (s_j, f_j) \in I \text{ and } 0 \leqslant s_j < 3 < f_j \leqslant 8\}.$$

These definitions guarantee that any feasible schedule for I can be transformed into a feasible schedule for I' and a feasible schedule for I''. Conversely, the shifts (0, 5) and (0, 8) are the only shifts in S overlapping the instant $t = 3$. As these shifts have the same begin time, a feasible schedule for I' and a feasible schedule for I'' can be 'pasted together' into a feasible schedule for I. Therefore I is a yes-instance if and only if both

I' and I'' are yes-instances. As SCS(S_5') and SCS(S_5'') can be solved in polynomial time, it follows that SCS(S_5) can be solved in polynomial time also.

A decomposition as described can be applied if there exists an instant $t \notin \{b_z, e_z \mid z = 1, 2, \ldots, Z\}$ which is such that all shifts overlapping t have either the same begin time or the same end time, there is at least one shift finishing before t and there is at least one shift starting after t. Unfortunately such an instant t does not exist for the set of shifts S_6 which was defined in Section 3. The graph $G(S_6)$ is again a cycle of length 5. The computational complexity of SCS(S_6) is still an open problem. The complicating factor in this example is the fact that many shifts exist with equal begin or end times.

These considerations show that the definition of the graph $G(S)$ may be not sufficiently sophisticated for expressing a complete classification of the computational complexity of the problems SCS(S), as the graph treats two non-overlapping shifts in the same way as two overlapping shifts with the same begin or end times. However, a useful alternative has not been found yet. Therefore in the remainder of this section we will assume that the begin times of all shifts are different and that the end times of all shifts are different. The aim of the remainder of this section is to provide a classification of the computational complexity of the problems SCS(S) satisfying this assumption.

Theorem 11. *If the set of shifts S is such that all shifts in S have different begin times and different end times, then the following statement holds:* SCS(S) *can be solved in polynomial time* $\Leftrightarrow$ *the maximum shift overlap is less than or equal to* 2.

Proof. The 'if'-part of this theorem follows directly from Corollary 10. In order to prove the 'only if'-part suppose that the set of shifts S contains three overlapping shifts x, y and z with different begin times and different end times. If the begin times of these shifts satisfy $b_x < b_y < b_z$, then the end times of these shifts satisfy one of the following inequalities.

(i) $e_z < e_y < e_x$,
(ii) $e_z < e_x < e_y$,
(iii) $e_y < e_z < e_x$,
(iv) $e_y < e_x < e_z$,
(v) $e_x < e_z < e_y$,
(vi) $e_x < e_y < e_z$.

According to Lemma 3, in case (i) the set of shifts $\{x, y, z\}$ is equivalent to the set of shifts

$$S_3 = \{(0, 12), (1, 11), (2, 7)\}.$$

In Lemma 12 it is shown that SCS(S_3) is NP-complete. Furthermore, in case (vi) the set of shifts $\{x, y, z\}$ is equivalent to the set of shifts

$$S_4 = \{(0, 12), (1, 13), (2, 14)\}.$$

In Lemma 13 it is shown that SCS(S_4) is NP-complete by a straightforward reduction from SCS(S_3). Combining these results with the Lemmas 2 and 3 it follows that SCS(S) is NP-complete in the cases (i) and (vi). A similar trick as in the proof of Lemma 13 can be applied to show that SCS(S) is NP-complete in the cases (ii) to (v) as well. □

Lemma 12. SCS(S_3) *is* NP-*complete.*

Proof. This lemma is proved by a reduction from N3DM. Hence let I_1 be an instance of N3DM containing the integer t and the rational numbers a_i, b_i and c_i for $i = 1, 2, \ldots, t$.

Now an instance I_2 of SCS(S_3) is constructed as follows. In the shift (0, 12) we have t machines, in the shift (1, 11) we have $t^2 - t$ machines and in the shift (2, 7) we have t^2 machines. Furthermore, let for $i, j = 1, 2, \ldots, t$ the rational numbers A_i, B_j and X_{ij} be chosen in such a way that all these numbers are different and that for $i, j = 1, 2, \ldots, t$ we have $3 < A_i < B_j < 4 < X_{ij} < 7$. Then the jobs that have to be carried out in I_2 are the following:

- $(0, A_i)$ for $i = 1, 2, \ldots, t$.
- $t - 1$ times $(1, B_j)$ for $j = 1, 2, \ldots, t$.
- $t - 1$ times $(2, A_i)$ for $i = 1, 2, \ldots, t$.
- $(2, B_j)$ for $j = 1, 2, \ldots, t$.
- (A_i, X_{ij}) for $i, j = 1, 2, \ldots, t$.
- (B_j, X_{ij}) for $i, j = 1, 2, \ldots, t$.
- $(X_{ij}, 7)$ for $i, j = 1, 2, \ldots, t$.
- $(X_{ij}, 9 + a_i + b_j)$ for $i, j = 1, 2, \ldots, t$.
- $(10 - c_k, 12)$ for $k = 1, 2, \ldots, t$.

An example of an instance I_2 constructed from the instance I_1 of N3DM that was defined in Section 3 is presented in Figure 4. Note that in this example a feasible schedule for all jobs exists. Now we will prove the following statement for the general case: 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 $(0, A_i)$ are carried out by the machines in the shift (0, 12), the jobs $(1, B_j)$ are carried out by the machines in the shift (1, 11) and the jobs $(2, A_i)$ and $(2, B_j)$ are carried out by the machines in the shift (2, 7). Furthermore, the jobs $(10 - c_k, 12)$ are carried out by the machines in the shift (0, 12). As the jobs $(X_{ij}, 9 + a_i + b_j)$ finish later than the instant 7, they are carried out by the machines in the shifts (0, 12) and (1, 11). This implies that the jobs $(X_{ij}, 7)$ are carried out by the machines in the shift (2, 7).

It is not difficult to see that in the interval (0, 7) the total required processing time equals the total available processing time$(= 11t^2 + t)$. Hence as far as a machine is available during the interval (0, 7), it is uninterruptedly busy. This implies that each job $(0, A_i)$ is combined with a job (A_i, X_{ij}) on one machine and that the same holds for each job $(2, A_i)$. Each job $(1, B_j)$ is combined with a job (B_j, X_{ij}) on one machine and the same holds for each job $(2, B_j)$. The schedules on the ma-
chines in the shift (0, 12) are of the form

$$(0, A_i)(A_i, X_{ij})(X_{ij}, 9 + a_i + b_j),$$

where each i occurs exactly once. The schedules on the machines in the shift (1, 11) are of the form

$$(1, B_j)(B_j, X_{ij})(X_{ij}, 9 + a_i + b_j),$$

where each j occurs exactly $t - 1$ times. Hence among the jobs $(X_{ij}, 9 + a_i + b_j)$ that are scheduled on the machines in the shift (0, 12) each i and each j occurs exactly once.

From the fact that $\sum_{i=1}^{t}(a_i + b_i + c_i) = t$ it follows that in the interval (9, 12) the total required processing time on the machines in the shift (0, 12) equals the total available processing time on the machines in the shift $(0, 12)(= 3t)$. Hence the machines in the shift (0, 12) are uninterruptedly busy during the interval (9, 12). This implies that on a machine in the shift (0, 12) a job $(X_{ij}, 9 + a_i + b_j)$ is combined with a job $(10 - c_k, 12)$ in such a way that $9 + a_i + b_j = 10 - c_k$. This means that $a_i + b_j + c_k = 1$. So if we define $\rho(i) = j$ and $\sigma(i) = k$ whenever job $(X_{ij}, 9 + a_i + b_j)$ is combined with job $(10 - c_k, 12)$, then ρ and σ are the required permutations for I_1 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

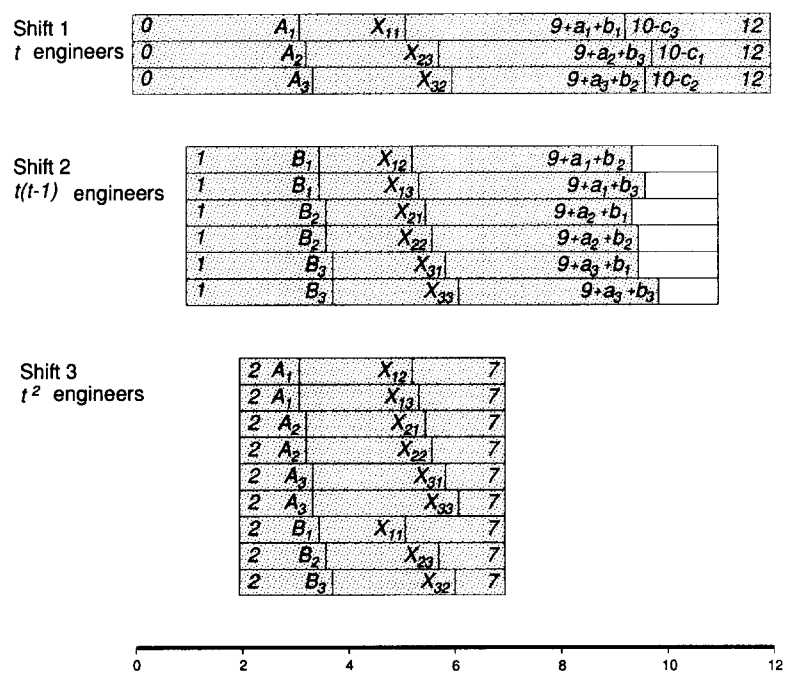

Figure 4. Feasible schedule for an instance of SCS(S_3)

schedule for I_2. As N3DM is NP-complete and it is clear that SCS(S_3) is in NP, it follows that SCS(S_3) is NP-complete also. □

Figure 4 gives an example of an instance I_2 constructed from the instance I_1 of N3DM that was defined in Section 3. The schedule in Figure 4 is a feasible schedule for all jobs.

Lemma 13 SCS(S_4) *is* NP-*complete.*

Proof. This lemma is proved by a reduction from SCS(S_3). Let I_1 be an instance of SCS(S_3) containing J jobs (s_j, f_j) that have to be carried out and 3 integers M_1, M_2 and M_3 representing the numbers of machines in the shifts (0, 12), (1, 11) and (2, 7) respectively and satisfying $M_z \leqslant J$ for $z = 1, 2, 3$.

Then an instance I_2 of SCS(S_4) is constructed as follows. In the shifts (0, 12), (1, 13) and (2, 14) we have M_1, M_2 and M_3 machines available. Furthermore, in I_2 the following jobs have to be carried out:

- J jobs (s_j, f_j) from the instance I_1.
- M_2 times the dummy job (11, 13).
- M_3 times the dummy job (7, 14).

As $M_2 \leqslant J$ and $M_3 \leqslant J$, the size of I_2 is polynomial in the size of I_1. Moreover, it is evident that I_1 is a yes-instance if and only if I_2 is a yes-instance. As SCS(S_3) is NP-complete and it is clear that SCS(S_4) is in NP, it follows that SCS(S_4) is NP-complete also. □

6. Concluding remarks

In this paper the problems SCS(S) and MSCS(S), 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 the problems MSCS(S) and a classification of the computational complexity of a large subset of the problems SCS(S). At this moment we are trying to extend the classification of the problems SCS(S) into a complete classification. Any suggestions that will lead us into this direction will be appreciated.

In this paper we did not look at optimization methods for calculating optimal or satisfying solutions. However, some preliminary experiments have shown that Linear Programming can be useful as a kernel for optimization algorithms or heuristics. These aspects of the problems SCS(S) and MSCS(S) will be a topic for further research.

Until now we have focused mainly on the operational questions that should be answered in the aircraft maintenance process. However, in Section 1 we mentioned already that tactical questions with respect to the required number of engineers in each of the shifts should be answered also. A subset of these tactical problems, which we have called Shift Class Design with respect to the set of shifts S or SCD(S) for short, can be described more formally in terms of jobs and machines as follows:

Instance of SCD(S).

- J jobs (s_j, f_j) that have to be carried out.
- Z integers c_z representing the costs per machine in each of the Z shifts.

Question. What are the minimum total costs for hiring machines if all jobs have to be carried out in a non-preemptive way?

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 SCD(S), more or less analogous to the classifications 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. University of Toronto, Department of Industrial Engineering, Toronto, 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] Dondeti, V.R., and Emmons, H., "Resource requirements for scheduling with different processor sizes, Parts I and II", Technical memoranda 579 and 589, Case Western Reserve University, Department of Operations Research, Cleveland, 1986.

[5] Dondeti, V.R., and Emmons, H., "Interval scheduling with processors of two types", Case Western Reserve University, Department of Operations Research, Cleveland (1989), to appear in *Operations Research*.

[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] Golumbic, M.C., *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, New York, 1980.

[8] Garey, M.R., and Johnson, D.S., *Computers and Intractability: A guide to the Theory of NP-Completeness*, Freeman, San Fransisco, 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] Kindervater, G.A.P., "Excercises in parallel computing", Ph.D. Thesis, Centre of Mathematics and Computer Science, Amsterdam, 1989.

[12] Kolen, A.W.J., and Kroon, L.G., "On the computational complexity of (maximum) class scheduling", *European Journal of Operational Research* 54/1 (1991) 23–38.

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

[14] Kroon, L.G., "Job scheduling and capacity planning in aircraft maintenance", Ph.D. Thesis, Erasmus University Rotterdam, 1990.