SABİT İŞ ÇİZELGELEME PROBLEMİ: Literatür Değerlendirmesi

Deniz Türsel Eliiyi, Meral Azizoğlu
Orta Doğu Teknik Üniversitesi, 06531, Ankara

Özet

Sabit iş çizelgelemesi hizmet ve imalat sistemlerinde sıkça rastlanan bir problemdir. Problem, sabit geliş ve termin zamanları olan aktivitelerin paralel özdeş kaynaklara yerleştirilmesi olarak tanımlanabilir. Pratik önemine rağmen, literatürde sabit iş çizelgelemesi üzerine çok az sayıda çalışma bulunmaktadır. Bu çalışmalarda, kaynak kullanım maliyetlerinin minimizasyonu (taktiksel problem) veya aktivitelerin toplam kar maksimizasyonu (operasyonel problem) hedeflenmiştir. Taktik problem tüm işlerin işlenmesini, operasyonel problem ise işlenecek aktivite kümesinin belirlenmesini gerektirmektedir. Biz bu çalışmada, sabit iş çizelgelemesi problemi üzerine literatürde yer alan çalışmaları, çalışma zamanı (working time), yaygınlık zamanı (spread time) veya işleyebilirlik (eligibility) kısıtları altında sınıflandırdık ve karşılaştırmalı değerlendirmeler yaptık. Ayrıca gelecek çalışmalara ışık tutması amacıyla açık araştırma alanlarını belirledik.

***Anahtar Kelimeler*:** *Interval Scheduling, Sabit İş Çizelgelemesi*

THE FIXED JOB SCHEDULING PROBLEM: A Literature Review

Abstract

Fixed Job Scheduling is an important problem in production and service environments. The problem can be defined as scheduling jobs that have fixed ready times and deadlines to identical parallel machines. Despite its practical importance, there are very few studies in literature about this problem. In existing studies, the aim is to minimize the number of machines (tactical problem) or maximize the total profit of jobs (operational problem). In this study, we consider the problem under working time, spread time and eligibility constraints. We review the related literature and state open areas for research.

***Keywords*:** *Interval Scheduling, Fixed Job Scheduling*

1. Introduction

Interval Scheduling (IS) problem is an emerging area of scheduling where tasks, each with specified ready times and deadlines, are to be processed on a number of resources. The IS problem is typical for reservation systems and has many real life applications, such as classroom assignment, transportation systems, and shift scheduling.

Reservation systems may arise in a service environment like hotel room reservations and car rental/repair services, where customers represent tasks and hotel rooms, cars or technicians correspond to resources. They may also take place in production environment, where tasks become parts or jobs, and resources become machines. In either environment, tasks give reserve requests to resources during specified time windows of processing, and the problem may involve the decisions as to which orders to accept, or how many resources to accommodate to serve all orders.

A reservation system typically refers to a parallel resource environment. More specifically, using conventional scheduling terminology (this terminology will be employed throughout the text), there are n independent jobs available to be processed in a non-preemptive fashion on m parallel machines. The time window of job j is specified by a release (ready) time r_j and a due time (deadline) d_j. If the time window of a job is larger than its processing time; that is, if it may be delayed after its start time, this type of IS problem is referred to as the Variable Job Scheduling Problem. On the other hand, if the job cannot be delayed after its ready time, then the IS problem becomes a Fixed Job Scheduling (FJS) problem. The processing time p_j of job j is equal to $(d_j - r_j)$ in the FJS problem, where it is smaller than or equal to $(d_j - r_j)$ in Variable Job Scheduling Problem. If a job is accepted for processing, it will start at r_j and finish at d_j in FJS. However, in variable job scheduling, the start time of the accepted job is a decision variable that takes a value between r_j and $(d_j - p_j)$.

In this study, we consider the FJS problem, which has the following two variants based on objective functions. The first variant is the Operational Fixed Job Scheduling (OFJS) problem. In OFJS problem, each job has a weight w_j that represents its value or relative importance (e.g. profit made by

processing the job) for the decision maker, and maximizing weighted number of processed jobs with a given number of processors is of concern. When all jobs have equal weights, i.e. when $w_j=w$ for all j, the objective reduces to maximizing the number of jobs processed. This problem is also known as the Maximal (Weight) Interval Scheduling Problem. In a more general OFJS model, the weight of job j may also depend on the machine to which it is assigned, then the weight of job j if processed on machine k becomes w_{jk} (the profit depends on the machine as well as the job). The second variant is the Tactical Fixed Job Scheduling (TFJS) problem. In this variant of the FJS problem, there is a fixed cost c_k associated with each machine, and the objective is the minimization of the total cost of the machines needed to process all available jobs. When machine costs are identical, i.e. when $c_k=c$ for all k, associated the objective reduces to minimizing the number of machines necessary. Note that the number of machines m is a parameter for the OFJS problem, while it becomes an upper bound in the TFJS problem.

There are many practical applications of TFJS and OFJS cited in the literature. In a study by Kroon (1990), the TFJS problem is used as the core model in tactical capacity planning of aircraft maintenance personnel for an airline company. An aircraft arriving at the airport requires a number of short maintenance inspections, which are specified by strict maintenance norms. Hence, the timetables of airline companies and the maintenance norms determine the fixed intervals in which the inspections have to be carried out in order to avoid aircraft delays. The problem is further complicated by a safety rule that gives license to each engineer to carry out inspections on at most two different aircrafts. A later study by Kroon *et al.* (1995) addresses the OFJS variant of the same case, i.e. for a given number of maintenance engineers, where the priorities are defined for maintenance inspections.

Another application of the TFJS problem is the Bus Driver Scheduling Problem (see Fischetti *et al.*, 1987). Given a transit company's bus schedule for a particular day, the Bus Driver Scheduling Problem finds a set of driver duties that covers the schedule at minimum cost, while satisfying a set of constraints laid down by the union contract and company regulations.

The OFJS problem may as well be observed in scheduling earth-observing satellites (EOS), as reported by Wolfe and Sorensen (2000). Hundreds of orbiting satellites are used to observe the earth and transmit data to the ground for further processing. A satellite occasionally passes through positions where specific data can be collected, and it needs a link to a relay satellite for transmission to the ground. However, there are very few relay satellites and they do not have enough capacity to guarantee a link. The priority for any observation is set beforehand, and this helps resolve satellite conflicts.

In the most basic setting of the FJS problem, each job needs to be processed on at most one machine. Individual machines are able to perform all operations, i.e. all machines are eligible for processing all jobs. In addition, the processing time requirement of a job does not depend on the machine that it is assigned to, that is, all machines are identical. Preemption and job splitting are not allowed. There are no restrictions as to the capacity of a machine, and all machines are available at all times.

In practice, the cases are often more sophisticated so that additional constraints and parameters that are more general are needed. Some variations of the FJS problem are the following:

- **Uniform machines:** As opposed to having identical machines, when the machine speed factors determine the processing times of the jobs, the problem is referred to as FJS problem with uniform machines. More formally, processing time of a job j on machine k (p_{jk}) is expressed as the product of p_j and a speed factor s_k in FJS with uniform machines. As an example, the relay satellites in the EOS problem may have different transmission speeds, and the overall speed of transmission to the ground may depend on both the observing satellite and the relay satellite that it links to. There is also a more general case of the FJS problem with unrelated machines, where the speed factor is defined relative to jobs and machines, and s_{jk} denotes the speed factor of machine k for job j. The processing time of job j on machine k is then p_j*s_{jk}.
- **Eligibility constraints:** When each machine is eligible to process a subset of jobs rather than the whole set, the problem becomes FJS with eligibility constraints. In this situation, there may be several job and machine classes based on their similarities, and for each combination of a job class and a machine class, the feasibility of assigning a job to a machine is established in advance. Aircraft and Classroom Scheduling are two problems where eligibility constraints find application.
- **Operating time constraints:** As a generalization of the FJS problem, there may be some constraints on the operating times of the machines.
  - ⇒ **Spread time constraints:** One such generalization includes spread time constraints, where an upper bound S is imposed on the total time between the start and the finish times of the operations on any machine. In particular, the spread time of a machine is defined as the time between the earliest ready time and the latest deadline among all jobs that are processed on that machine. Note that there may be idle times of the machine between these two time points, yet

these times are included by definition in the spread time. In the Bus Driver Scheduling Problem example, such a constraint is imposed on the duty of any driver. Thus, for example, if S=8 hours, a driver who sets out his duty with a job starting at 10 a.m. should stop working at 6 p.m., and cannot be assigned any job with deadline later than 6 p.m. Spread time bound may be the same for all machines, or arbitrary for each machine.

⇒ **Working time constraints:** Another generalization includes working time constraints where the actual time that the machine operates is bounded; that is, no machine is allowed to operate for more than a given total working time T. Consequently, working time of a machine is the total processing times (p_j s) of jobs assigned to that machine, and in contrast to spread time, it does not include idle times. For example, when T=8 for the Bus Driver Scheduling Problem instance above, a job with deadline 10 p.m. may well be assigned to the driver if there is much idle time in between. Moreover, in production environments, it may not be economical to operate some high precision machines more than their preset allowable time, T. Working time bound may be the same or different for different machines, as in the case of spread time.

- **Availability constraints:** When any of the machines is available only for a specified time interval rather than being continuously available, the resulting problem is FJS with availability constraints. In the airline company example, there are different shifts for the engineers, since the arrivals and departures of aircrafts take place continuously. Teams of engineers are available at the airport during specific time intervals (shifts) of some consecutive hours. In such a situation, a job can be carried out in a shift by a machine if the time between the ready time and the deadline of the job forms a subinterval of the machine's shift. Different shifts may have different costs, as well.

The rest of this report is organized as follows. In the next section, we present the mathematical formulation for the basic models of the FJS problem and review the related literature. In Section 3, the FJS problem with working time constraints is examined in detail. Sections 4 and 5 we cover FJS with spread time constraints and FJS with eligibility constraints. Finally, in Section 6, we introduce some topics for the OFJS problem that are worth studying further.

2. Basic Models of the FJS Problem

In this section, we present the formulations for the two variants of the FJS problem, namely the OFJS problem and the TFJS problem. In the formulation of the basic model of the FJS problem, the following conventional scheduling assumptions are made:

- All machines are identical.
- A machine can process at most one job at a time.
- All machines can process all jobs, i.e., there is no concern of eligibility.
- There are no machine breakdowns, and no interval of machine unavailability.
- A job should be processed without any interruption once it is started, i.e. no preemption is allowed.
- A job can be processed by at most one machine at a time, i.e. no job splitting is allowed.

Job-related parameters of the problem are:

- r_j: ready time of job j $\quad j=1,\ldots,n$
- d_j: deadline of job j $\quad j=1,\ldots,n$
- p_j: processing time of job j, $p_j = d_j - r_j$ $\quad j=1,\ldots,n$
- w_{jk}: weight of job j if it is processed on machine k $\quad j=1,\ldots,n \quad k=1,\ldots,m$

Without loss of generality, we assume that all numerical data are positive integers. The time is divided into periods that are not necessarily equal in length. This is achieved by forming a chronological sequence of ready times and deadlines of all jobs. Namely, let $\{t_1, t_2,\ldots\}$ be the sorted sequence of the r_js and d_js in chronological order with duplicates removed. Let P_a be the set of jobs that need to be processed in the interval $[t_a, t_{a+1})$ for a=1, 2,...

Our binary decision variable is defined as follows:

$$x_{jk} = \begin{cases} 1, & \text{if job } j \text{ is processed on machine } k, \\ 0, & \text{otherwise.} \end{cases} \qquad \forall j,k.$$

The fundamental constraints that are common for both variants of the FJS problem are the following:

- Each job j is processed on only one machine:

$$\sum_{k=1}^{m} x_{jk} \le 1 \qquad j = 1,...,n \qquad (1)$$

- No machine can process more than one job at a time:

$$\sum_{j \in P_a} x_{jk} \le 1 \qquad k = 1,...,m \qquad \forall a \qquad (2)$$

- Integrality constraints for job-machine assignments:

$$x_{jk} \in \{0,1\} \qquad k = 1,...,m \qquad j = 1,...,n \qquad (3)$$

2.2. The OFJS Model

The common constraints defined in the previous section define the feasible region for the OFJS problem. The complete model for OFJS with the most general objective function becomes:

$$\text{Maximize } \sum_{k=1}^{m} \sum_{j=1}^{n} w_{jk} x_{jk} \qquad (4)$$

subject to (1), (2), and (3).

The objective function expressed in (4) maximizes the total weighted number of jobs to be processed, in other words it maximizes total profit. When the weights are identical for all jobs and machines, i.e. when $w_{jk} = w$, $\forall j,k$, the problem reduces to the maximization of the number of processed jobs.

Bouzina and Emmons (1996) provide an algorithm for solving the OFJS problem with the objective of the maximization of the number of processed jobs, i.e. the Maximal IS problem. The algorithm is stated below.

Algorithm 2.2.1:
S1. Set $S = \varnothing$
S2. Index the jobs in chronological order of arrivals (ready times).
S3. Consider the jobs sequentially. At each ready time, add the arriving job to set *S*. If no machine is available at the ready time of a job, remove the job with the latest deadline from set *S*.

Bouzina and Emmons (1996) also formulate the total weight maximization problem where $w_{jk} = w_j$, i.e. the Maximal Weight IS problem, as a Minimum Cost Network Flow (MCNF) problem with n+1 nodes and 2n arcs. Hence, the OFJS problem is polynomially solvable, since there exist polynomial time algorithms for MCNFP (Orlin, 1993). Below is the description of the MCNF algorithm specified for solving this OFJS problem.

Algorithm 2.2.2:
S1. Index jobs in chronological order of ready times.
S2. Create nodes $s = V_1, V_2,\ldots, V_n$ for each job, and a dummy node $t = V_{n+1}$. Connect V_j to V_{j+1} with arc cost zero and capacity m, j=1,…,n.
S3. Create arc (V_j, V_k), where V_k is the first job not overlapping with job j, j=1,…,n. If no such job exists, create arc (V_j, t). Each of these arcs has cost $-w_j$ and capacity 1.
S4. Require a flow of m from s to t and solve the resulting minimal cost flow problem.

We illustrate the network structure through the following example problem.

Example 1: Consider the jobs in Figure 1 to be scheduled on 2 parallel identical machines. Labels are the job numbers. Assume $w_j = p_j$, for j=1,…,8, i.e. the profit is charged per hour.

Using Algorithm 2.2.1, the maximum number of jobs is found as 6, where the processed job set becomes {1,3,4,5,6,7}.

The network structure for Algorithm 2.2.2 is given in Figure 2, where each arc is labeled with its cost $-w_j$, and capacity 1, except for the arcs generated in Step 2 (those on the straight line from s to t), whose costs and capacities are 0 and 2, respectively. The minimal cost becomes –35 and utilizes the arcs corresponding to the job set {4,5,6,7,8}.

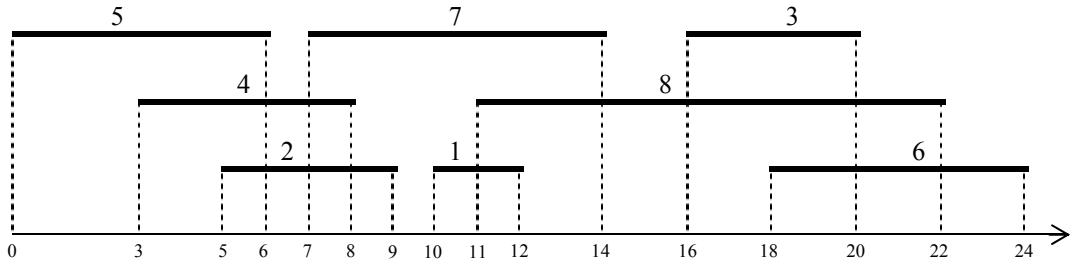

Figure 1. Instance of the OFJS problem

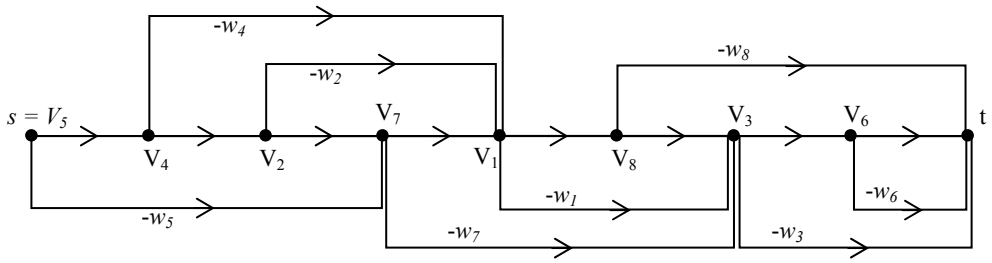

Figure 2. Network representation of the instance

2.3. The TFJS Model

In formulating the TFJS problem, a decision variable for machines is also necessary. For this purpose, we let;

$$y_k = \begin{cases} 0, & \text{if no job is assigned to machine } k, \\ 1, & \text{otherwise.} \end{cases} \qquad \forall k.$$

Note that, in the case of the TFJS problem, the parameter m is an upper bound on the number of machines. One such valid upper bound is the number of jobs, n. Given any upper bound, the following additional constraints are necessary to represent the TFJS problem:

$$x_{jk} \le y_k \qquad k = 1,...,m \qquad j = 1,...,n \qquad (5)$$

$$y_k \in \{0,1\} \qquad k = 1,...,m \qquad (6)$$

Constraint set (5) prevents the assignment of the jobs to unutilized machines. Constraint set (6) guarantees integrality for machine variables. Then, the complete model for the TFJS problem becomes:

$$\text{Minimize} \sum_{k=1}^{m} c_k y_k \qquad (7)$$

subject to (1'), (2), (3), (5), and (6),

where constraint (1') is the equality form of constraint (1), and c_k is the cost of operating machine k. The objective function expressed in (7) minimizes the total cost combination of machines. When $c_k = c$, the problem becomes the minimization of the total number of machines.

The TFJS problem with $c_k = c$ was studied by Dantzig and Fulkerson (1954), and by Gertsbakh and Stern (1978) in the context of fleet planning. Hashimoto and Stevens (1971), and Gupta *et al.* (1979) also studied this problem in the context of computer wiring. An optimal solution to this problem is described by the following lemma.

Lemma 2.3.1: The minimum number of machines required to carry out all jobs of an instance of the TFJS problem equals the maximum job overlap of the jobs.

This lemma is a direct consequence of Dilworth's theorem on partially ordered sets, stating that in any partially ordered set, the minimum number of chains required for covering all elements is equal to the size of a maximum anti-chain (Dilworth, 1950). An $O(n \log n)$ algorithm for determining the maximum job overlap of the jobs is described by Hashimoto and Stevens (1971) and by Gupta *et al.* (1979). Note that according to our notation, the maximum job overlap is $\text{Max}_a\{|P_a|\}$, where $|P_a|$ is the

cardinality of set P_a. Examine Figure 1 of Example 1. As the maximum job overlap equals 3, the minimum number of machines required to carry out all jobs equals 4 as well.

3. The FJS Problem with Working Time Constraints

When there is a limit imposed on the total working time of each machine, the following additional constraint becomes necessary for both TFJS and OFJS models:

$$\sum_{j=1}^{n} p_j x_{jk} \le T_k \qquad k = 1,...,m, \qquad (8)$$

where T_k is the processing capacity of machine k. It is assumed that no job j has p_j greater than any $T_k, j = 1,\ldots,n, k = 1,\ldots,m$. Throughout the study, we assume that

$\sum_{j=1}^{n} p_j > T_k, \forall k$, so that the working time constraints are not redundant.

As to the best of our knowledge, the only study in literature regarding the TFJS problem with working time constraints is that of Fischetti, Martello and Toth's (1989). In their study, the authors deal in particular with the Bus Driver Scheduling Problem mentioned in section 1, the tactical version of the FJS problem, where the objective is minimizing the number of drivers necessary to perform all duties (they denote the problem as FSW). The duty of a driver/crew is defined as the set of tasks assigned to him. A crew cannot work for more than a given working time T in a day.

The authors prove that the FSW problem is NP-hard in the strong sense through transformation from Bin Packing Problem. The Bin Packing Problem is shown to be NP-hard in the strong sense by Garey and Johnson (1979). Fischetti *et al.* (1989) provide the mathematical formulation of the problem, and present two graph theoretical models based on vertex coloring and digraphs, respectively. They present an $O(n^2)$ algorithm that optimally solves the preemptive version of the problem. They also prove that the optimal objective function value can be found in $O(n \log n)$ time. Such an algorithm can be of value when the optimal schedule is not needed. Some other polynomially solvable special cases are analyzed in the study, and some lower bounds are developed and incorporated in a Branch and Bound algorithm. The computational results show that their algorithm's performance is quite satisfactory for the data sets generated close to real-life situations.

In a later study by the same authors (Fischetti *et al*., 1992), some polynomial time approximation algorithms for the FSW problem are provided. One such algorithm is based on the preemptive relaxation and runs in $O(n^2)$ time. Two other greedy $O(n \log n)$ algorithms and a two-phase $O(n^2 \log n)$ algorithm are presented, and all algorithms are compared in terms of their running times and quality of the solutions.

The only study in literature related with the OFJS problem with working time constraints is that of Bouzina and Emmons' (1996). The authors deal with the preemptive version of the problem with the objective of maximizing the number of jobs processed. They provide a polynomial time algorithm that solves this problem optimally by solving a MCNF problem repeatedly using Algorithms 2.2.1 and 2.2.2. Algorithm 3.1 below is a restatement of their algorithm.

Algorithm 3.1:
S1. Index jobs in their nondecreasing order of p_j values.
S2. Initialize $S = \varnothing$
S3. Repeat
$j = j+1$
If $m+1$ jobs in $S \cup \{j\}$ overlap at the same time point, then
Apply Algorithm 2.2.1 to the jobs in $\{1,\ldots,j\}$ → Resulting schedule S'
If $|S'| = S+1$, then

Set w_j = M-p_j where $M \ge \sum_{j=1}^{n} p_j$ (a large number) for jobs in $\{1,\ldots,j\}$,

Solve the resulting problem using Algorithm 2.2.2 → Resulting schedule S^*

If $\sum_{i \in S^*} p_i \le mT$, then set $S = S^*$, else, set $S = S \cup \{j\}$, until $j = n$, or $p_{j+1} + \sum_{i \in S} p_i > mT$.

Bouzina and Emmons (1996) show that the feasibility problem of OFJS where the number of jobs is maximized, i.e. $w_j = 1$, is NP-complete through transformation from the Bin Packing Problem.

Thus, the associated optimality problem is NP-hard in the strong sense, as the Bin Packing Problem is known to be NP-hard in the strong sense (Garey and Johnson, 1979). The authors also prove that the problem remains NP-hard, even when preemptions are allowed.

4. The FJS Problem with Spread Time Constraints

Recall that the spread time for a machine k is defined as $(d_j - r_i)$ where j is the last job and i is the first job processed by machine k, and the jobs are assumed to be indexed in nondecreasing order of their ready times. A limit imposed on the total spread time of machines necessitates an extra set definition and a constraint for both TFJS and OFJS models. We define a set I_j as the incompatibility set for job j as the following:

$I_j = \{ i > j : r_i < d_j \text{ or } d_i - r_j > S \}$.

Note that the assignment of two jobs i and j such that $j \in I_i$ or $i \in I_j$ to the same machine violates the spread time constraint. To impose spread time, the following constraint should be added to the formulation of TFJS and OFJS models:

$$x_{ik} + x_{jk} \le 1 \qquad k = 1,...,m,\ j = 1,...,n-1,\ i \in I_j \tag{9}$$

As in the case of the FJS problem with working time constraints, the only study in FJS literature with working time constraints is that of Fischetti, Martello and Toth's (1987). In their study, the authors deal with the Bus Driver Scheduling Problem with an additional constraint on spread time of the drivers.

The authors prove that TFJS with spread time constraints is NP-hard in the strong sense through transformation from Circular Arc Coloring Problem, which is known to be NP-hard in the strong sense (Garey and Johnson, 1979). Fischetti *et al.* provide the mathematical formulation of the problem, and present a graph theoretical model based on vertex coloring problem. They provide an $O(n \log n)$ algorithm that optimally solves the preemptive version of the problem. Some lower bounds with nice performance ratios are developed, and some dominance criteria that help to reduce the problem size are defined as well. A Branch and Bound algorithm is described for solving the problem, and computational results are provided.

In a later study (Fischetti *et al*., 1992), some polynomial time approximation algorithms for the problem are provided. One such algorithm is a greedy one that runs in $O(n \log n)$ time. Two other $O(n \log n)$ time algorithms that are based on the preemptive relaxation of the model are presented. The authors also present two other greedy algorithms and an algorithm that is based on a longest path solution. Worst case performance analyses of algorithms are also presented and all algorithms are compared in terms of their running times and quality of solutions.

5. The FJS Problem with Eligibility Constraints

When each machine is eligible to process only a subset of jobs in FJS, several job and machine classes may be formed based on similarities. For each combination of a job class and a machine class, the feasibility of assigning a job to a machine is established in advance, either by using zero/one matrices or by some set definitions. In addition to the parameters defined in Section 2.1, class parameters for jobs and machines are defined as follows:

- e_j: the job class of job j $\qquad j=1,\ldots,n$
- f_k: the machine class of machine k $\qquad k=1,\ldots,m.$

The number of different job classes and machine classes are denoted by E and F, respectively. For each combination of a job and a machine class, an (E x F) zero/one matrix, L, represents the feasibility of assigning a job in a job class to a machine in a specific machine class. Rows of matrix L denote job classes, while columns denote machine classes. L_{ef} takes the value of one if and only if it is allowed to assign jobs in job class e to machines in machine class f. An example L matrix, L_1 is given below:

$$L_1 = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}.$$

Note that L_1 represents a special case with two machine classes having a hierarchical eligibility structure. Machine class 2 can process only the jobs in job class 2, while machine class 1 can process all jobs. We may illustrate this eligibility structure schematically by Figure 3. In the figure, while any machine can process the jobs in job class 2, job class 1 can only be processed by a subset of the machines, namely machine class 1.

With this matrix definition, the additional condition $Le_j f_k = 1$ should be added to constraint sets (1), (2), and (5) to complete the formulation of the FJS problem with eligibility constraints. The assignment variable x_{jk} will assume zero value if the corresponding matrix entry $Le_j f_k$ is zero.

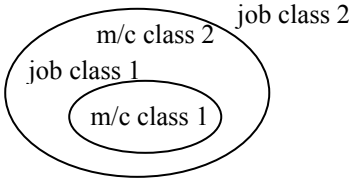

Figure 3. The hierarchical eligibility structure imposed by L1

Another way to assure the satisfaction of eligibility constraints is to redefine the job weights as follows:

$$w_{jk} = \begin{cases} w_{e_j}, & \text{if machine } k \text{ can process job } j \\ -\infty, & \text{otherwise.} \end{cases}$$

There are a number of studies in literature on FJS with eligibility constraints. The problem takes different names based on the inherent eligibility structure. Each of these problems will be discussed separately.

The TFJS problem with eligibility constraints:

The TFJS problem with identical machine classes and one job class is considered by Hashimoto and Stevens (1971), Gertsbakh and Stern (1978), and Gupta *et al*. (1979). This problem is equivalent to the basic TFJS problem, which deals with one job and machine class, and can be solved in $O(n \log n)$ time. Arkin and Silverberg (1987) make an alternative representation of eligibility by letting a V_j to represent the subset of machines by which job j can be processed. Note that, in our context for job j with job class e_j, the set V_j corresponds to the set of machines belonging to those machine classes f with $Le_j f = 1$. The authors show that the corresponding feasibility problem that aims to find a feasible schedule that processes all jobs is NP-complete.

Similar problems were studied by several authors. Dondeti and Emmons (1992) prove that the TFJS problem with minimization of cost objective with L matrices L_1 and L_2 (L_2 given below) can be solved in polynomial time by repeatedly solving a Maximum Network Flow Problem, changing upper bounds on arcs iteratively.

$$L_2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{pmatrix}.$$

In L_2, the machines in machine class i can process the jobs in job classes i and 3. We may illustrate this eligibility structure schematically by the following figure. In the figure, while any machine can process the jobs in job class 3, job class 1 can only be processed by machine class 1, and similarly job class 2 can only be processed by machine class 2.

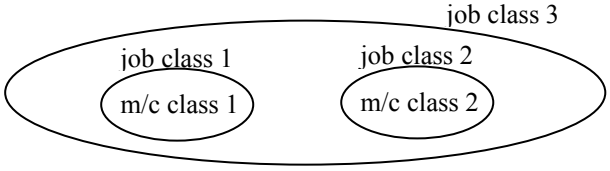

Figure 4. The hierarchical eligibility structure imposed by L2

Dondeti and Emmons (1992) also show that, for these special matrices, the tactical version of the problem that finds the required number of machines in each machine class can be solved in polynomial time.

Kolen and Kroon (1992) study the TFJS problem that minimizes the total cost of machines with the five specific L matrices, including L_1 and L_2. They show that the problem with L_2 can be solved in polynomial time by a combination of Linear Programming and Network Flow algorithms. They prove that the feasibility problem is NP-complete if L has at least 3 columns, i.e. when the number of machine classes (F) is greater than 2. They also show that the optimality problem with $F > 2$ machine classes is NP-hard.

Dondeti and Emmons (1993) refer to the TFJS problem with eligibility and minimization of cost objective as the Exclusive Class Scheduling (ECS) problem. They study the preemptive Exclusive Class Scheduling (ECS) problem. They formulate the preemptive feasibility problem as a set of transportation problems that are solved in polynomial time. However, they conjecture that the preemptive optimization problem is NP-hard.

Kroon *et al.* (1997) present a complete classification for the TFJS problem with eligibility constraints in terms of complexity. When the number of machine classes, F, is variable, the preemptive and nonpreemptive versions of the problem become NP-hard in the strong sense. On the other hand, when F is fixed and smaller than 3, both versions are polynomially solvable. When F is fixed and greater than or equal to 3, only the nonpreemptive version becomes NP-hard. The authors develop some lower and upper bounds for the problem, incorporate these into Branch and Bound algorithm and Lagrangean Relaxation based procedures. Their computations show that the approaches are effective for medium-sized problems.

Jansen (1994) provides an approximation algorithm for the TFJS problem with eligibility and availability constraints, with a specific 3x3 square L matrix, where machine class 1 can process all jobs, machine class 2 can process job classes 1 and 2, and machine class 3 can process job classes 1 and 3. His algorithm has a complexity of $O(F.Z.n^2)$, where Z is the number of shifts. A genetic algorithm using a reinforcement learning system is developed by Santos and Zhong (2001) for the TFJS problem with eligibility constraints and minimum total cost objective. Their computational experiments reveal that their approach is robust in terms of problem structure, and promising with respect to catching a fast local minimum.

The OFJS problem with eligibility constraints:

Arkin and Silverberg (1987) show that the feasibility problem for the OFJS problem with eligibility is NP-complete, when $F > 2$. They also present an $O(n^{m+1})$ time Dynamic Programming based algorithm for the OFJS problem with eligibility constraints. The existence of the algorithm implies that, the OFJS problem can be solved in polynomial time if the number of machines is fixed. Kolen and Kroon (1991) study five specific L matrices for the OFJS problem with different eligibility structures and at most 3 job and machine classes. They prove that the OFJS problem with at least 2 machine classes ($F > 1$) is NP-hard in the strong sense, through transformation from the Numerical Three Dimensional Matching Problem, which is known to be NP-hard in the strong sense (Garey and Johnson, 1979). They also show that the associated feasibility problem is NP-complete if and only if L has at least 3 columns.

Kroon *et al.* (1995) provide a Lagrangean relaxation based approximation algorithm for the OFJS problem with eligibility constraints. Their algorithm is based on the observation that the OFJS problem can be modeled as a MCNF problem when all machines are identical. They relax the restriction that each job should be processed at most once. For each machine class, they solve a MCNF problem having some side constraints. Their computational results based on hypothetical and real data reveal that their algorithm is satisfactory for practical purposes.

The Hierarchical Class Scheduling (HCS) problem:

The Hierarchical Class Scheduling (HCS) problem is defined as a problem with a upper triangular (or lower triangular) square L matrix, which implies that a machine in class f can process jobs in classes e such that $e \leq f$ (or $e \geq f$). Note that the matrix L_1 represents such an eligibility structure. Kroon *et al.* (1997) show that the feasibility problem for the tactical HCS problem is NP-complete when $F \geq 4$. Dondeti and Emmons (1992) show that the tactical HCS problem can be solved in polynomial time if $F \leq 2$. Dondeti and Emmons (1993) propose a greedy algorithm for the optimal preemptive schedule for the tactical HCS problem with $F > 2$ and with cost minimization objective. They also show that the feasibility problem associated with the nonpreemptive version is NP-complete.

For the operational version of the problem, Kolen and Kroon (1991) prove that the problem is NP-hard in the strong sense, when $F > 1$. They also show that the associated feasibility problem is NP-

complete when $F \geq 3$. Bouzina and Emmons (1995) study the operational HCS problem with L_1, and conjecture that the problem of maximizing the number of jobs is NP-hard. They provide a polynomial time algorithm for the preemptive version of this problem, and a polynomial time algorithm for the case having additional machine availability constraints.

The One-or-all Class Scheduling (OCS) problem:

The One-or-all Class Scheduling (OCS) problem is defined as follows: There are E machine classes and E+1 job classes, and a job in job class e can only be processed by a machine of class e, e=1,2,…,E, and jobs in class *zero* can be processed by all machines. Note that the matrix L_2 represents such an eligibility structure. Dondeti and Emmons (1993) prove that the feasibility problem for the tactical OCS problem is NP-complete. The authors provide an algorithm for the preemptive tactical OCS problem with the minimization of total cost objective. The algorithm provides an optimal preemptive solution, where only class-zero jobs are preempted. They also show that the feasibility problem associated with the nonpreemptive version is NP-complete.

Table 1 combines the complexity results in literature on the FJS problem with eligibility constraints.

Table 1. Results for the FJS problem with eligibility constraints

Objective Functions	
Minimize Total Cost (Minimize number of machines)	**Maximize Total Weight (Maximize number of jobs)**
The feasibility problem is NP-complete. (Arkin & Silverberg, '87)	The feasibility problem is NP-complete for F >2. (Arkin & Silverberg, '87)
F = 2→ Polynomially solvable. (Dondeti, Emmons '92) F > 2 → NP-hard. (Kolen, Kroon '92)	F > 1 → NP-hard. (Kolen, Kroon '91)
Preemptive version: Feasibility problem is polynomially solvable. (Dondeti, Emmons '93) Optimality problem: F fixed → polynomially solvable. F not fixed → Strongly NP-hard. (Dondeti, Emmons '93)	
HCS: Feasibility problem is NP-complete for $F \geq 3$. (Kroon et. al. '97) F =2 → Polynomially solvable. (Dondeti, Emmons '92)	**HCS:** HCS with F = 2 is NP-hard. (Kolen, Kroon '91) Feasibility problem with $F \geq 3$ is NP-complete. (Kroon et. al. '97)
Preemptive HCS: Polynomially solvable. (Dondeti, Emmons '93)	**Preemptive HCS:** F = 2→ Polynomially solvable (Bouzina, Emmons '95)
OCS: Feasibility problem is NP-complete. Preemptive OCS is polynomially solvable. (Dondeti, Emmons '93)	

6. Further Topics in FJS

When any of the machines is available only for a specified time interval, the resulting problem is FJS with availability constraints. The availability constraints usually appear as shifts of consecutive hours for each machine, or sets of machines.

The FJS problem with eligibility constraints, and the FJS problem with availability constraints are closely related. To see the relation, recall the definition of Arkin and Silverberg (1987) for the OFJS problem with eligibility constraints. Hence, the OFJS problem with availability constraints can be seen as

a special case of the OFJS problem with eligibility constraints. Therefore, the $O(n^{m+1})$ time algorithm of Arkin and Silverberg (1987) can solve the OFJS problem with availability constraints, as well.

The problem with availability constraints seems also related with the OFJSS problem (the OFJS problem with spread time constraints). Note that, the OFJSS problem is a more general case where the beginning times of the shifts of the machines are not predetermined, but are decision variables.

As opposed to the situation where all machines are identical in terms of processing speeds, the machine speed factors may determine the processing times of the jobs, in which case the problem is referred to as FJS problem with uniform machines. The processing time of a job j on machine k is expressed as the product of p_j and a speed factor s_k. All the problems analyzed in the previous chapters for identical machines can be extended to the uniform parallel machine case. Most of these problems are NP-hard even when the machines are identical. Exact algorithms may be generated for some special cases for the uniform machine environments. Detecting these special cases and proposing solution approaches for the general case can open new research avenues.

References

Arkin A.M., Silverberg E.L., 1987. Scheduling Jobs with Fixed Start and End Times, *Discrete Applied Mathematics*, 18, 1-8.

Bouzina K.I., Emmons H., 1995. "Interval Scheduling with Shift and 2-Level Hierarchy Constraints", presentation in: *Informs Meeting*, Los Angeles.

Bouzina K.I., Emmons H., 1996. Interval Scheduling on Identical Machines, *Journal of Global Optimization*, 9, 379-393.

Dantzig G.L., Fulkerson D.R., 1954. Minimizing the Number of Tankers to Meet a Fixed Schedule, *Naval Research Logistics Quarterly*, 1, 217-222.

Dondeti V.R., Emmons H., 1992. Fixed Job Scheduling with Two Types of Processors, *Operations Research*, 40, S76-S85.

Dondeti V.R., Emmons H., 1993. Algorithms for Preemptive Scheduling of Different Classes of Processors to do Jobs with Fixed Times, *European Journal of Operational Research*, 70, 316-326.

Dilworth R.P., 1950. A Decomposition Principle for Partially Ordered Sets, *Annals of Mathematics*, 51, 161-166.

Fischetti M., Martello S., Toth P., 1987. The Fixed Job Schedule Problem with Spread-Time Constraints, *Operations Research*, 35, 849-858.

Fischetti M., Martello S., Toth P., 1989. The Fixed Job Schedule Problem with Working-Time Constraints, *Operations Research*, 37, 395-403.

Fischetti M., Martello S., Toth P., 1992. Approximation Algorithms for Fixed Job Schedule Problems, *Operations Research*, 40, S96-S108.

Gertsbakh I., Stern H.I., 1978. Minimal Resources for Fixed and Variable Job Schedules, *Operations Research*, 26, 68-85.

Gupta U.L., Lee D.T., 1979. Leung J.Y.-T., An Optimal Solution to the Channel Assignment Problem, *IEEE Transactions on Computers*, 28, 807-810.

Hashimoto A., Stevens J.E., 1971. "Wire Routing by Optimizing Channel Assignments within Large Apertures", in: *Proceedings of the 8th Design Automation Workshop*, 155-169.

Jansen K., 1994. An Approximation Algorithm for the License and Shift Class Design Problem, *European Journal of Operational Research*, 73, 127-131.

Kolen A.J.W., Kroon L.G., 1991. On the Computational Complexity of (Maximum) Class Scheduling, *European Journal of Operational Research*, 54, 23-38.

Kolen A.J.W., Kroon L.G., 1992. License Class Design: Complexity and Algorithms, *European Journal of Operational Research*, 63, 432-444.

Kroon L.G., 1990. "Job Scheduling and Capacity Planning in Aircraft Maintenance", PhD Thesis, Rotterdam School of Management, Erasmus University, The Netherlands.

Kroon L.G., Salomon M., Van Wassenhove L.N., 1995. Exact and Approximation Algorithms for the Operational Fixed Interval Scheduling Problem, *European Journal of Operational Research*, 82, 190-205.

Kroon L.G., Salomon M., Van Wassenhove L.N., 1997. Exact and Approximation Algorithms for the Tactical Fixed Interval Scheduling Problem, *Operations Research*, 4, 624-638.

Orlin J.B., 1993. A Faster Strongly Polynomial Minimum Cost Flow Algorithm, *Operations Research*, 41, 338-350.

Santos E.J., Zhong X., 2001. Genetic Algorithms and Reinforcement Learning for the Tactical Fixed Interval Scheduling Problem, *International Journal on Artificial Intelligence Tools*, 10, 23-38.
Wolfe W.J., Sorensen S.E., 2000. Three Scheduling Algorithms Applied to the Earth Observing Systems Domain, *Management Science*, 46, 148-168.