Available online at www.sciencedirect.com

ScienceDirect

European Journal of Operational Research 178 (2007) 331–342

EUROPEAN JOURNAL OF OPERATIONAL RESEARCH

www.elsevier.com/locate/ejor

Invited Review

Fixed interval scheduling: Models, applications, computational complexity and algorithms

Mikhail Y. Kovalyov [a,*], C.T. Ng [b], T.C. Edwin Cheng [b]

[a] *Faculty of Economics, Belarus State University, and United Institute of Informatics Problems, National Academy of Sciences of Belarus, Skorini 4, 220050 Minsk, Belarus*

[b] *Department of Logistics, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong*

Received 5 April 2005; accepted 24 January 2006
Available online 7 July 2006

Abstract

The defining characteristic of fixed interval scheduling problems is that each job has a finite number of fixed processing intervals. A job can be processed only in one of its intervals on one of the available machines, or is not processed at all. A decision has to be made about a subset of the jobs to be processed and their assignment to the processing intervals such that the intervals on the same machine do not intersect. These problems arise naturally in different real-life operations planning situations, including the assignment of transports to loading/unloading terminals, work planning for personnel, computer wiring, bandwidth allocation of communication channels, printed circuit board manufacturing, gene identification and examining computer memory structures. We present a general formulation of the interval scheduling problem, show its relations to cognate problems in graph theory, and survey existing models, results on computational complexity and solution algorithms.

Keywords: Interval scheduling; Interval graph; *k*-coloring; *k*-track assignment; Discrete starting times; Maximum weight clique; Maximum weight independent set

1. General formulation of the fixed interval scheduling problem

Various formulations and special cases of the problem to be addressed in this paper have been considered in the literature under different labels, including "(fixed) interval scheduling", "interval selection", "scheduling with discrete starting times", "fixed job scheduling", "channel assignment (reservation)", "bandwidth allocation", "*k*-track assignment", "*k*-coloring of intervals", "finding *K*-independent sets on interval graphs", "on-line interval scheduling", "seat reservation" and "maximizing the number of on-time jobs", among others. We use the term "fixed interval scheduling" because it appears to be the most popular in the literature and adequately reflects the nature of the underlying problems. The results for several variants of the

[*] Corresponding author. Tel.: +375 17 2842125; fax: +375 17 2318403.
E-mail addresses: koval@newman.bas-net.by (M.Y. Kovalyov), lgtctng@polyu.edu.hk (C.T. Ng), lgtcheng@polyu.edu.hk (T.C. Edwin Cheng).

fixed interval scheduling problem have been reinvented or obtained independently for equivalent formulations of the same problem. The aim of this paper is to review the existing models, computational complexity results and solution methods for the fixed interval scheduling problem.

The fixed interval scheduling problem can be stated as follows. There are n independent non-preemptive jobs to be scheduled for processing on m *unrelated parallel machines*. A machine can process at most one job at a time, and a job can be executed by at most one machine at a time. For each machine l, a set of jobs N_l is specified such that no job $j \notin N_l$ can be processed on machine l, $l = 1,\ldots,m$. Furthermore, machine *unavailability intervals* $U_{vl} = (a_{vl}, b_{vl}]$, $a_{vl} < b_{vl}$, $v = 1,\ldots,u_l$, are given such that machine l cannot process any job within these intervals, $l = 1,\ldots,m$. Denote $U_l = U_{1l} \cup \cdots \cup U_{u_l l}$, $l = 1,\ldots,m$. On machine l, job j can be processed in one of the *fixed intervals* $I_{jlk} := (s_{jlk}, d_{jlk}]$, $s_{jlk} < d_{jlk}$, $k = 1,\ldots,n_{jl}$. A weight w_{jlk} is associated with each interval I_{jlk}, which is related to the value derived from processing job j in this interval. Each job can be processed only in one of its intervals, or is not processed at all (rejected). All the data are assumed to be non-negative integers. Note that it can be assumed without loss of generality that each set N_l is determined by $N_l = \{j | n_{jl} \geqslant 1\}$.

A schedule is characterized by the set of processed jobs and their assignment to appropriate intervals on the machines. A schedule and the corresponding assignment are feasible if the following constraints are satisfied: (a) if any job j is selected for processing in an interval I_{jlk}, then $j \in N_l$ and $I_{jlk} \cap U_l = \phi$, and (b) the intervals selected for processing the jobs on the same machine do not intersect. The problem is to find a feasible schedule maximizing the total weight of the processed jobs. We denote this problem as *problem P*.

Note that job processing times are not present in the formulation of problem P. If there are job processing times, then the above formulation should be extended by the assumptions that each job can be completely processed within any of its intervals and no two occupied intervals can intersect on the same machine.

The fixed interval scheduling problem and its various variants appear in many applications, including fleet planning [29,45], computer wiring and bandwidth allocation of communication channels [53,51,52,22,11], scheduling of bus drivers [72,39], class scheduling [61,21], retail trade [62], VLSI circuit design [20], assignment of incoming aircraft to gates and work planning for aircraft maintenance personnel [64], planning of satellite photography [42], printed circuit board manufacturing [91], satellite data transmitting [35], genome comparison in molecular biology [93], spectroscopical methods in the identification of protein-encoding genes [24,23], solving problems of disparity between processor and memory speeds in computers [92,17], real-time (or time-constrained) scheduling [9,75], planning cargo loading/unloading processes [78].

There exist results on the fixed interval scheduling problem with the objective to minimize the number of machines that accommodate all the jobs [53,51,61,63,62,39,64], to minimize machine costs, subject to the machines being able to accommodate all the jobs [31,37], and to maximize the total processing time of the assigned jobs [70,68]. We will not review these results.

Let us introduce some notation for special cases of the above formulated interval scheduling problem P.

Problem P1. In this problem, a single time interval is associated with each job, a job can be processed within this interval on any of the machines, and the interval weights are all unit. Thus, $N_l = \{1,\ldots,n\}$, $n_{jl} = 1$, $w_{jl1} = 1$, $s_{jl1} = s_j$, $d_{jl1} = d_j$, $I_j = (s_j, d_j]$ and $U_l = \phi$ for $j = 1,\ldots,n$ and $l = 1,\ldots,m$.

Problem P2. A generalization of problem P1 to the case where $U_l \neq \phi$, $l = 1,\ldots,n$.

Problem P3. A generalization of problem P1 to the case of arbitrary job weights w_j, $j = 1,\ldots,n$.

Problem P4. The job processing intervals are job and machine dependent, and there is at most one interval for each job on each machine. Thus, $n_{jl} \leqslant 1$, $s_{jl1} = s_{jl}$, $d_{jl1} = d_{jl}$, $I_{jl1} = I_{jl} = (s_{jl}, d_{jl}]$ and $w_{jl1} = w_{jl}$ for $j = 1,\ldots,n$, $l = 1,\ldots,m$.

Problem P5. A special case of problem P in which there is a single machine.

Observe that any of the problems P1 and P3 can be viewed as a parallel machine scheduling problem with job release dates s_j, due dates d_j, non-preemptive processing times p_j and additional constraints $p_j = d_j - s_j$, $j = 1,\ldots,n$. We are aware only of the following three papers addressing such problems with the objective of minimizing a weighted deviation of job completion times from their due dates. Heady and Zhu [54] suggested a heuristic algorithm, Balakrishnan et al. [5] proposed a mixed integer programming formulation, and Sivrikaya-Serifoglu and Ulusoy [90] developed two genetic algorithms.

Notice that the constraints $p_j = d_j - s_j$, $j = 1, \ldots, n$, were not considered in these studies. Problems P1 and P3 can be also formulated as parallel machine scheduling problems to maximize the number of *just-in-time jobs* that complete exactly at their due dates. In these problems, there are job processing times and no job release dates. In this paper, we review two papers considering these problems, by Lann and Mosheiov [65] and Hiraishi et al. [55]. Furthermore, problems P1 and P3 fall into the class of *real time scheduling* problems, see [86] for a recent survey. Since the real time scheduling problems are strongly NP-hard, heuristic and enumerative methods have been proposed in the literature for their solution. These methods do not take into account the specificity of the fixed interval scheduling problem that each job occupies its feasible time interval fully. Therefore, these methods are unlikely to be useful in solving problems P1 and P3.

Notice that problem P4 and the general problem P are polynomially reducible to problem P5. Given an instance of problem P4 or problem P, the corresponding instance of problem P5 can be obtained by shifting each job processing interval I_{jlk} and machine unavailability interval U_{vl} to start $(l-1)T$ time units later, $l = 1, \ldots, m$, where T is the length of the planning horizon for the corresponding instance of problem P4 or P.

The next section presents some definitions and notation from graph theory that will be used in the rest of the paper. Sections 3–7 address problems P1–P5, respectively. Section 7 also addresses the most general problem P. The paper concludes with a summary of the results and suggestions for future research.

2. Definitions from graph theory

A *graph* $G = G(V, E)$ is a pair of sets V and E, where V is a set of elements called *vertices* and E is a set of unordered pairs (i,j), $i,j \in V$, called *edges*. We work with finite graphs $G(V, E)$ that have no self-loops or parallel edges. Thus, $(i, i) \notin E$ for any $i \in V$, and all the pairs in E are distinct. If $(i,j) \in E$, then vertices i and j are called *adjacent* vertices *connected* by the edge (i,j). For $Z \subseteq V$, a *vertex subgraph* $G(Z)$ of graph $G(V, E)$ is defined by $G(Z) = G(Z, E')$, where $E' = \{(i,j) | i,j \in Z, (i,j) \in E\}$. Graph G is *complete* if all its vertices are pairwise adjacent.

Let graph $G(V, E)$ be given. A set of vertices $C \subseteq V$ is a *clique* if $G(C)$ is complete. A set of vertices $S \subseteq V$ is an *independent set* if $G(S)$ has no edges (consists of isolated vertices). A *maximal clique* (*independent set*) is a clique (independent set) $X \in V$ such that $X \cup \{i\}$ is not a clique (independent set) for any $i \in V \backslash X$. A *maximum clique* (*independent set*) is a clique (independent set) $X \in V$ of maximum cardinality. The *weight of a clique* (*independent set*) is the total weight of its vertices. The *maximum weight clique (independent set) problem* is to find a maximum weight clique (independent set) in a given graph.

Graph $G(V, \overline{E})$ is a *complement* of graph $G(V, E)$ if $(i, j) \in \overline{E}$ if and only if $(i,j) \notin E$.

The vertices of a graph are called *legally colored* if there are no adjacent vertices colored with the same color. The *chromatic number* of graph G, denoted as $\chi(G)$, is the least number of colors needed to legally color its vertices.

A graph is called *perfect* if for every of its vertex subgraph, the size of its maximum clique is equal to the chromatic number of this subgraph.

Graph $G(V, E)$ is a *cycle* if $V = \{i_1, \ldots, i_k\}$ and $E = \{(i_r, i_{r+1}) | r = 1, \ldots, k, i_{k+1} := i_1\}$.

A *Berge graph* is a graph that has no vertex subgraph being a *cycle* with an odd number of vertices greater than or equal to five or a *complement* of such a cycle.

An *interval graph* is a graph in which the vertices are associated with intervals of a line and there is an edge between two vertices if and only if the corresponding intervals intersect. A *co-interval graph* is a complement of an interval graph. Therefore, there is an edge between two vertices of a co-interval graph if and only if the corresponding intervals do not intersect. Interval and co-interval graphs are perfect, see, for example, [47]. This statement is a consequence of the more general statement that the complement of any perfect graph is perfect.

A *circular-arc graph* is a graph in which the vertices are associated with segments of a circle and there is an edge between two vertices if and only if the corresponding segments intersect.

A *directed graph* (*digraph*) $G(F, D)$ is determined by the set of vertices F and the set of *arcs* D, where arc $i \to j$ is a directed edge from vertex i to vertex j. A *path* in digraph $G(F, D)$ is a set of its vertices $\{j_1, \ldots, j_k\}$ such that there are arcs $j_i \to j_{i+1} \in D$, $i = 1, \ldots, k-1$. Vertex i is called a *predecessor* of vertex j, and vertex j is called a *successor* of vertex i if there exists a path going from i to j. Vertex i is called an *immediate predecessor* of vertex j and vertex j is called an *immediate successor* of vertex i if there exists an arc $i \to j$. A digraph is *acyclic* if every of its vertices is not a successor of itself.

3. Problem P1

Consider problem P1, in which a single interval is associated with each job, a job can be processed within this interval on any of the machines, and the interval weights are all unit. Thus, $N_l = \{1,\ldots,n\}$, $n_{jl} = 1$, $w_{jl1} = 1$, $s_{jl1} = s_j$, $d_{jl1} = d_j$ and $I_j = (s_j, d_j]$ for $j = 1,\ldots,n$ and $l = 1,\ldots,m$. Let $G(V,E)$ be an interval graph corresponding to problem P1 such that $V = \{I_1,\ldots,I_n\}$ and $(I_i, I_j) \in E$ if and only if intervals I_i and I_j intersect. problem P1 is equivalent to finding a *maximum m-colorable subgraph of graph* $G(V,E)$, which is a vertex subgraph of this graph with the maximum number of vertices that can be legally colored using m colors. The set of vertices colored with the same color corresponds to the set of jobs assigned to the same machine. Yannakakis and Gavril [96] presented an $O(n + |E|)$ time algorithm for the latter problem, assuming that m is a constant. Notice that the number of edges in the interval graph is $O(n^2)$ in general. Furthermore, in order to use the algorithm of Yannakakis and Gavril, the graph $G(V,E)$ must be constructed, which also takes $O(n + |E|)$ time.

Faigle and Nawjin [36], and Bouzina and Emmons [15] suggested the following algorithm for solving problem P1.

Algorithm A1P1 (*Faigle and Nawjin, Bouzina and Emmons*)

Step 1. Number the jobs in nondecreasing order of the interval start times such that $s_1 \leqslant \cdots \leqslant s_n$. Initiate a set of jobs $S = \phi$.

Step 2. For $j = 1,\ldots,n$, compute the following. Add job j to the set S. If there is a machine that can start the processing of job j at time s_j, assign j to this machine. Otherwise, if no machine is free at time s_j, then remove from S the job with the largest ending time d_j.

The time complexity of Algorithm A1P1 is $O(n\max\{\log n, m\})$. Faigle and Nawjin stressed that this algorithm is an optimal on-line algorithm because it assigns a newly arrived job by using only the information about the jobs that have arrived so far. In the considered on-line model, it is assumed that a job that has been started but not completed can be rejected.

The best existing off-line algorithm for problem P1 is due to Carlisle and Lloyd [20] and Faigle and Nawjin [36]. It can be described as follows. Given an instance of problem P1, construct the corresponding co-interval graph $G(V,E)$ such that $V = \{I_j | j = 1,\ldots,n\}$ and $(I_i, I_j) \in E$ if and only if intervals I_i and I_j do not intersect. Further, number the intervals such that $d_1 \leqslant \cdots \leqslant d_n$ and consider an acyclic digraph $G(V,D)$, where $I_i \to I_j \in D$ if and only if $(I_i, I_j) \in E$ and $i < j$. problem P1 can be viewed as the problem of finding a collection of disjoint paths $P_1,\ldots,P_m$ in $G(V,D)$ such that $|P_1 \cup \cdots \cup P_m|$ is maximized. If such a collection is given, the vertices of the same path P_l correspond to the jobs assigned to the same machine l. This problem can be solved by the following algorithm, which is our interpretation of the algorithm given in Faigle and Nawjin [36].

Algorithm A2P1 (*Carlisle and Lloyd, Faigle and Nawjins*)

Step 1. Number the jobs in nondecreasing order of the interval ending times such that $d_1 \leqslant \cdots \leqslant d_n$. Initiate the sets of jobs $P_l = \phi$, $l = 0,1,\ldots,m$. Set P_0 will include the intervals of the rejected jobs and set P_l will include the intervals of the jobs assigned to machine l, $1 \leqslant l \leqslant m$.

Step 2. For $j = 1,\ldots,n$, compute the following. Try to find an interval I_i of the maximal index $i < j$ such that
  (a) $I_i \in P_1 \cup \cdots \cup P_m$,
  (b) I_i has the maximal index in a set P_l it belongs to,
  (c) $I_i \to I_j$.

If such an interval I_i exists, then assign I_j to P_l if $I_i \in P_l$. If no such interval I_i exists and there is a set $P_l = \phi$, $1 \leqslant l \leqslant m$, then set $P_l = \{I_j\}$. If no such interval I_i exists and $P_l \neq \phi$, $l = 1,\ldots,m$, then assign I_j to P_0.

Carlisle and Lloyd [20] described an implementation of Algorithm A2P1 such that Step 2 requires $O(m + n)$ time. Therefore, it solves problem P1 in $O(n\log n)$ time.

Other polynomial time algorithms for this problem can be found in Frank [40], Arkin and Silverberg [2], Yannakakis and Gavril [96], Lee and Sarrafzadeh [67], Hsiao et al. [56], Harms [52], and Lann and Mosheiov [65]. Note that the algorithms of Harms, and Lann and Mosheiov are not optimal. We give a counter example, where there are three jobs and two machines. The interval starting and ending times are given by $(s_1, s_2, s_3) = (2,1,0)$ and

$(d_1, d_2, d_3) = (3, 2, 3)$. The mentioned algorithms output a solution where job 1 is processed on machine 1 and job 2 on machine 2. Job 3 is rejected. This solution is not optimal because jobs 1 and 2 can be processed on machine 1 and job 3 on machine 2.

4. Problem P2

Problem P2 generalizes problem P1 such that there are machine unavailability intervals. Brucker and Nordmann [18] studied a special case of problem P2, in which there are two unavailability intervals on each machine – at the beginning and at the end of the planning horizon. This problem is called the k-track assignment problem ($k = m$ in our notation). Brucker and Nordmann [18] showed an equivalence of this problem to finding k disjoint independent sets of the maximum total cardinality in a circular-arc graph. The problem of determining whether there exist k disjoint independent sets that cover all the vertices of a circular-arc graph, or in other words, if the vertices of a circular-arc graph can be legally colored with k or less colors, is strongly NP-complete [43]. Therefore, the k-track assignment problem, as well as problem P2, is strongly NP-hard.

Brucker and Nordmann [18] and Hsu and Tsai [57] provided $O(n)$ time algorithms for the 2-track assignment problem. Furthermore, Brucker and Nordmann [18] presented an $O(n^{m-1}m!m^{m+1})$ time algorithm for the general m-track assignment problem and an $O(n^m m!m^m)$ algorithm for problem P2. Descriptions of all these algorithms are rather lengthy and will not be presented here.

An on-line algorithm for problem P2 was suggested by Faigle et al. [35]. Their on-line model coincides with that of Faigle and Nawjin [36], in which a started but non-completed job can be rejected. The algorithm constructs a schedule with an absolute deviation at most $m - 1$ from the off-line optimum.

5. Problem P3

Problem P3 generalizes problem P1 to the case of arbitrary weights. Arkin and Silverberg [2] reformulated problem P3 in terms of maximizing the total weight of legally colored vertices in the interval graph. Using a polyhedral approach (see [48,84] for the relevant polyhedral approaches), Arkin and Silverberg reduced the latter problem to a binary integer linear program (BILP) with a feasible domain being an integral polyhedron, i.e., a polyhedron with integral vertices. Due to the integrality of the feasible domain, the integrality constraints in BILP can be omitted and the problem can be solved by one of the linear programming algorithms (for example, the polynomial time algorithm of Khachiyan [60], or the strongly polynomial time algorithm of Vavasis and Ye [81]). Arkin and Silverberg further provided a more efficient $O(n^2 \log n)$ algorithm by reformulating problem P3 as a minimum cost flow problem (see [66] for the definition), where arcs represent the maximal cliques of the interval graph.

Bouzina and Emmons [15] and Carlisle and Lloyd [20] suggested improved minimum cost flow algorithms for problem P3 with the same computational complexity of $O(mn \log n)$. The algorithm of Bouzina and Emmons appears to be more efficient. It is described as follows.

Algorithm AP3 (*Bouzina and Emmonss*)

Step 1. Number the jobs in nondecreasing order of the interval start times such that $s_1 \leqslant \cdots \leqslant s_n$.

Step 2. Construct a digraph $G(F, D)$ with the set of vertices $F = \{I_1, \ldots, I_n, I_{n+1}\}$, where I_{n+1} is an artificial vertex, and the set of arcs $D = D_1 \cup D_2$, where $D_1 = \{I_j \to I_{j+1} | j = 1, \ldots, n\}$. Set D_2 is constructed as follows. For each interval I_j, $j = 1, \ldots, n$, create arc $I_j \to I_k \in D_2$, where I_k is the interval with the smallest index that does not intersect with I_j, $k = j + 1, \ldots, n$. If no such interval I_k exists, create arc $I_j \to I_{n+1} \in D_2$. The arcs from set D_1 have zero cost and capacity m. Each arc $I_j \to I_k \in D_2$ has cost $-w_j$ and capacity 1.

Step 3. Find the minimum cost flow of value m from vertex I_1 to vertex I_{n+1} in the digraph $G(F, D)$. Determine a set of arcs $A^* \subseteq D_2$ such that a flow on them is equal to 1 in the optimal flow. If $I_j \to I_k \in A^*$, then job j is not rejected in an optimal solution to problem P2. Set $X^* = \{j | I_j \to I_k \in A^*\}$.

Step 4. Solve problem P1 with the set of jobs X^* to find a feasible assignment of the selected jobs to the machines. Algorithm A2P1 in Section 3 can be applied for these purposes.

Sarrafzadeh and Lou [83], Pal and Bhattacharjee [79] and Saha and Pal [82] suggested less efficient algorithms for problem P3 based on finding m disjoint independent sets of the maximum total weight in an interval graph. Hiraishi et al. [55] reduced problem P3 to a transshipment problem in a 0–1 network, which is solvable in polynomial time, see [1]. This approach is less efficient than the algorithm AP3 as well.

An on-line version of problem P3, in which there is a single machine and a started but non-completed job can be (irrevocably) rejected, has been studied by several authors. We call this problem P3-1-On-Line.

In the literature, *deterministic* and *randomized on-line algorithms* are distinguished and a *competitive analysis* is used to evaluate their performance, see [89,10] for details. A deterministic on-line algorithm A is called *ρ-competitive* for a maximization problem if it delivers a feasible solution satisfying $F^A \geqslant \rho F^*$ for any problem instance, where F^A and F^* are the objective value delivered by the algorithm and the optimal objective value, respectively. A randomized on-line algorithm makes random choices with a certain probability, and the term F^A in the above definition is replaced by the *expected objective value*. The value of ρ is called *competitive ratio*.

Woeginger [95] proved that no deterministic on-line algorithm with a constant competitive ratio exists for the general problem P3-1-On-Line. Canetti and Irani [19] proved the same statement for randomized on-line algorithms. Woeginger also proposed an 1/4-competitive deterministic algorithm for a special case of problem P3-1-On-Line that includes the cases of identical length intervals and *monotone intervals* such that if job i arrives before job j, then not only $s_i < s_j$ but also $d_i \leqslant d_j$ is satisfied. For the same special case, Seiden [85] presented a randomized $\left(\frac{1}{2+\sqrt{3}} > \frac{1}{3.73206}\right)$-competitive algorithm. For problem P3-1-On-Line with monotone intervals, Miyazawa and Erlebach [73] suggested a randomized 1/3-competitive algorithm and proved that no randomized algorithm can achieve a competitive ratio strictly larger than 4/5.

An on-line version of problem P3 with m machines, in which a job once assigned to a machine cannot be rejected, was studied under the name "the seat reservation problem" by Boyar and Larsen [16] and Bach et al. [4]. In this problem, there is a train with m seats traveling through stations $1,\ldots,k$. Seat reservations can be made for any trip from station s to station d if $1 \leqslant s < d \leqslant k$. The reservation cannot be refused if there is a seat available for the entire requested trip. An algorithm that fulfills this requirement is called *fair*. Requests for seat reservation arrive over time and the problem is to handle them so as to maximize the sum of the prices of the tickets sold. Boyar and Larsen proved that any fair deterministic or randomized on-line algorithm for the unit price seat reservation problem is at least 1/2-competitive. Bach et al. [4] showed that the upper bound of 1/2 is asymptotically reachable for any fair deterministic algorithm and that 7/9 is an asymptotic upper bound for the competitive ratio of any fair randomized algorithm for the seat reservation problem.

6. Problem P4

In problem P4, the job processing intervals are job and machine dependent, and there is at most one interval for each job on each machine. Therefore, we can omit the index k in the notation. Observe that if $I_{jl} \cap U_l \neq \phi$, then job j cannot be processed on machine l. Therefore, all such jobs can be removed from the set N_l in problem P4. Let us make such a removal for each set N_l. After this modification, the relation $I_{jl} \cap U_l = \phi$ is satisfied for each $j \in N_l$, $l = 1,\ldots,m$.

From now on, we assume without loss of generality that there is no unavailability interval on each machine in problem P4, i.e., $U_l = \phi$, $l = 1,\ldots,m$.

Problem P4 with unit weights and $m = 2$ is strongly NP-hard even if the length of each interval I_{jl} is equal to 2, and at most two intervals intersect at each time instant, see Section 7.

Arkin and Silverberg [2] studied a special case of problem P4, where the job processing intervals are the same on all the machines, i.e., $I_{jl} = I_j = (s_j, d_j]$ and $w_{jl} = w_j$, $l = 1,\ldots,m$, $j = 1,\ldots,n$. We denote this special case as problem P4-I. Arkin and Silverberg proved that problem P4-I is NP-hard in the strong sense for a variable number of machines m, and it is solvable in $O(mn^{m+1})$ time and space by a reduction to the problem of finding a longest path in a specifically designed acyclic digraph with $O(mn^{m+1})$ arcs. Their reduction can be described as follows. Denote the mentioned digraph as $G(F,D)$. Number the interval starting and ending times in nondecreasing order, where a t_j precedes an s_i in case of a tie, and other ties are broken arbitrarily. This yields a nondecreasing sequence $u_1 \leqslant \cdots \leqslant u_{2n}$.

There are two artificial vertices S and T in F. The set of the remaining vertices is partitioned into $2n$ *layers* numbered $1,\ldots,2n$. Consider layer r. Let $u_r \in \{s_j, t_j\}$. The vertices of layer r correspond to the event that job j starts (if $u_r = s_j$) or completes (if $u_r = t_j$) on a machine. The vertices of layer r, $u_r \in \{s_j, t_j\}$, are associated with m-tuples $(x_1, \ldots, x_m)$ representing the possible schedules at time u_r. We shall not distinguish a vertex and the m-tuple associated with it. In an m-tuple $(x_1, \ldots, x_m)$, values $x_l \in \{1, \ldots, n, \phi\}$, where $x_l = i$ means that job i is processed on machine l and $x_l = \phi$ means that machine l is idle. If $u_r = s_j$, then $x_l = j$, and if $u_r = t_j$, then $x_l = \phi$ for $l \in \{1, \ldots, m\}$ such that $j \in N_l$ in each m-tuple of layer r. It is assumed that vertices S and T are associated with identical m-tuples $(\phi, \ldots, \phi)$ of layers 0 and $2n+1$, respectively. The set of arcs D is constructed as follows. There is an arc between m-tuples $(z_1, \ldots, z_m)$ and $(x_1, \ldots, x_m)$ of layers v and r for $v < r$. The length of this arc is equal to w_j if $u_r = s_j$, $x_l = j$, $z_l = \phi$ for $l \in \{1, \ldots, m\}$ and $z_k = x_k$, $k \neq l$, $k = 1, \ldots, m$. The length of this arc is equal to zero if $u_r = t_j$, $x_l = \phi$, $z_l = j$ for $l \in \{1, \ldots, m\}$ and $z_k = x_k$, $k \neq l$, $k = 1, \ldots, m$.

A longest path between vertices S and T in the digraph $G(F, D)$ corresponds to an optimal solution to problem P4-I.

An integer programming formulation and an approximation algorithm for problem P4-I based on the Lagrangian relaxation and decomposition was presented by Dijkstra et al. [30]. Heuristics for this problem based on a reduction to a maximum independent set problem were presented by Gabrel [42]. Approximation algorithms with worst-case performance guarantees for problem P4-I were given by Bar-Noy et al. [9], Bar-Noy et al. [8] and Bhatia et al. [11]. The best algorithm that guarantees a solution with a value at most $(1 - 1/e)$ times the optimum, where $e = 2.71828\ldots$, was presented by Bhatia et al. [11].

Problem P4 has been studied by Ng et al. [78]. They reduced it to the maximum weight clique (MWC) problem as follows. Introduce an *m-layer graph* $G(V, E)$, see Fig. 1 for an example.

The set of vertices is $V = N_1 \cup \cdots \cup N_m$, where set N_l determines *layer l* containing intervals I_{jl} for jobs $j \in N_l$ on machine l, $l = 1, \ldots, m$. The weight w_{jl} is associated with each vertex I_{jl}. We do not distinguish a vertex and its corresponding interval.

The set of edges is $E = E_1 \cup \cdots \cup E_{m+1}$, where $E_l = \{(I_{il}, I_{jl}) | I_{il} \cap I_{jl} = \phi\}$, $l = 1, \ldots, m$, and $E_{m+1} = \{(I_{il}, I_{jq}) | l \neq q, i \neq j, l, q = 1, \ldots, m, i, j = 1, \ldots, n\}$. Ver- bally, there is an edge between two intervals if they belong to the same layer and do not intersect or if they belong to different layers and do not correspond to the same job.

We call a graph constructed for problem P4 in the way described above as an *P4-graph*. This graph can be represented by a collection of graphs $G(N_l, E_l)$, $l = 1, \ldots, m$, and a simple rule that determines if there is an edge between any given two vertices of different layers.

Given P4-graph $G(V, E)$, let us associate an assignment of jobs to machines with a set of vertices $Z \subseteq V$ as follows: if $I_{jl} \in Z$, then job j is assigned to machine l. It is easy to see that the following statement holds.

Statement 1. A set of vertices Z, $Z \subseteq V$, of the P4-graph $G(V, E)$ is a clique if and only if the corresponding assignment of jobs to machines is feasible.

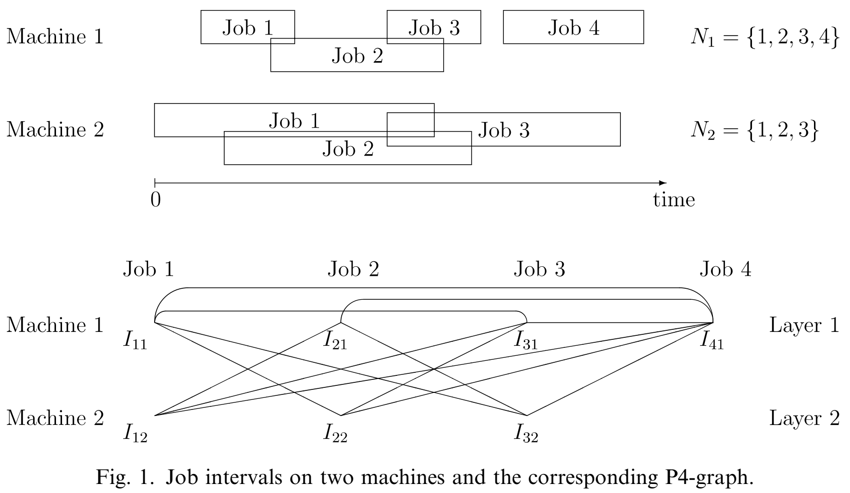

Fig. 1. Job intervals on two machines and the corresponding P4-graph.

This statement immediately implies

Statement 2. A maximum weight clique in the P4-graph $G(V,E)$ determines an optimal schedule for the corresponding problem P4.

There exists a vast body of the literature on solving the MWC problem. A comprehensive survey of equivalent formulations and solution methods was given by Bomze et al. [13]. The MWC problem admits several appropriate mathematical programming formulations. Among them are integer linear programs [76,77,49,50], quadratic 0-1 problems [88,80,41], and problems of maximizing or minimizing a quadratic function of continuous variables over a polyhedron [74,14,46,12]. A variety of heuristic techniques is available for the MWC problem. Earlier results can be found in Johnson and Trick [58] and Babel [3], and later results can be found in Battiti and Protasi [7], Marchiori [71], Fenet and Solnon [38] and Locatelli et al. [69].

The most famous class of graphs where the MWC problem is polynomially solvable is the class of perfect graphs. Recently, Chudnovsky et al. [27,28] proved the following Strong Perfect Graph Conjecture: a graph is perfect if and only if it is a *Berge graph*. The P4-graph is not perfect in general. An example is given in Fig. 1. There is a cycle $(I_{21},I_{32},I_{11},I_{31},I_{12},I_{21})$ with five vertices, which is the vertex subgraph of the original graph. Chudnovsky et al. [26] recently provided an $O(|V|^9)$ time algorithm for recognizing Berge graphs, which together with the fact that the sets of perfect graphs and Berge graphs coincide, imply that it is polynomial to recognize whether an arbitrary graph is perfect or not.

The MWC problem on perfect graphs was shown to be polynomially solvable by Grötschel et al. [49,50]. The main steps of their approach include formulating the corresponding integer linear program, demonstrating its optimal solution to be an optimal solution to the relaxed non-integer problem, and adapting the ellipsoid method [87,60] for solving the latter problem.

For some subclasses of perfect graphs, the recognition problem and the MWC problem can be solved more efficiently, see [47]. An example is the class of interval and co-interval graphs. However, a P4-graph can be neither interval nor co-interval. There are other classes of graphs differing from perfect for which the MWC problem is polynomially solvable. These classes, among others, include *h-perfect graphs* [48], TR^k, $k=1,\dots,6$, *graphs* [6], CSG^k *graphs* ((Chmeiss and Jégou [25], in their model, all weights are unit and k is a constant), and *interval-filament graphs* [44].

Problem P4 can be handled as follows. First, reformulate it as the MWC problem. Second, apply one of the existing methods for the latter problem, assuming that the structure of the corresponding P4-graph is arbitrary. Alternatively, classify the P4-graph in the given instance manually or by using the existing recognition algorithms. If the graph falls into a graph class admitting an efficient solution procedure for the MWC problem, use it.

Denote the MWC problem with an P4-graph as *problem MWC(P4)*. Ng et al. [78] suggested an algorithm, denoted as AP4, for solving problem MWC(P4). This algorithm enumerates different combinations of maximal cliques in the graphs $G(N_l,E_l)$, $l=1,\dots,m$. The time complexity of this algorithm is equal to $O(n^2+mn\prod_{l=1}^{m}|X_l|)$, where X_l is the set of all maximal cliques in the graph $G(N_l,E_l)$. Algorithm AP4 is exponential in m. For fixed m, it is polynomial in n if the number of maximal cliques in each graph $G(N_l,E_l)$ is bounded by a polynomial of n. A trivial case where algorithm AP4 is polynomial in n and m emerges when all the intervals on the same machine are mutually non-intersecting. In this case, there is a single maximal clique in each graph $G(N_l,E_l)$ being the set N_l of its vertices. Algorithm AP4 can be modified to run in $O(mn)$ time in this case.

Ng et al. [78] suggested two *greedy* heuristics to construct an approximate solution to problem MWC(P4).

Observe that problem P4 can be also formulated as the problem of finding a maximum weight independent set in the complement of the P4-graph. Therefore, all the related results of graph theory can be applied to solve this problem.

7. Problem P5 and problem P

Problem P5 is a special case of the general interval scheduling problem P, in which $m=1$. On the other hand, problem P is polynomially reducible to problem P5, see Section 1.

Consider problem P5. Since there is a single machine, we can omit the index l in the notation. Similar to problem P4, let us modify set N_1 such that there is no unavailability interval on the machine in problem P5, i.e., $U_1=\phi$.

Let us denote problem P5 with unit weights as P5-1. Referring to private communication, Spieksma [91] mentioned that Kolen proved problem P5-1 to be NP-hard even if the number of intervals $n_j \leqslant 2$ for each j. Furthermore, Spieksma provided an alternative proof that problem P5-1 is strongly NP-hard even if $n_j \leqslant 2$ for each j, the length of each interval I_{jk} is equal to 2, and at most two intervals intersect at each time instant. It follows from his proof that problem P4 with unit weights and $m = 2$ in strongly NP-hard.

Spieksma proved that, unless $\mathscr{P} = \mathscr{NP}$, problem P5-1 cannot be approximated in polynomial time within a factor of ε for an arbitrarily small $\varepsilon > 0$ (in other words, a polynomial time approximation scheme does not exist for this problem) and provided an algorithm that delivers a solution with a value at least 1/2 times the value of an optimal solution. He further established that the linear programming relaxation of an integer programming formulation of problem P5-1 gives a solution with a value at most 2 times the optimal value for problem P5-1 with arbitrary n_j, and with a value at most 5/3 times the optimal value for problem P5-1 with $n_j \leqslant 2, j = 1,\ldots,n$. All these results can be extended to the unit weights cases of the general problem P and problem P4.

Keil [59] showed that the existence of a feasible solution, where exactly one interval is selected for each job (all the jobs are scheduled) can be verified in $O(g + n\log n)$ time for problem P5 with $n_j \leqslant 2$, $j = 1,\ldots,n$, where g is the number of edges in the associated graph that we call *P5-graph*. A vertex in the P5-graph is associated with each interval and two vertices are connected by an edge if and only if the corresponding intervals belong to the same job or intersect. Note that g can be $O(n^2)$. Keil provided an $O(g + n\log n)$ reduction of the above problem of recognizing whether all n jobs can be scheduled to the 2-satisfiability problem, which is solvable in $O(n)$ time, see [32]. This result also applies for problem P4 with $m = 2$.

Approximation algorithms and their worst-case performance analysis for problems P5 and P were provided by Erlebach and Spieksma [33,34]. In particular, they presented a greedy algorithm that delivers a solution with a value at least 1/8 times the value of an optimal solution for problem P5 and a solution with a value at least $3 - 2\sqrt{2}$ times the value of an optimal solution for problem P5 for the case where the weights of all the intervals corresponding to the same job are equal.

A relation between problem P5 and the problem of finding a maximum weight independent set in the P5-graph was studied by Waterer et al. [94]. They established the properties of a P5-graph that can be used for describing facets in an integer programming formulations of problem P5.

Similar to problems P4, P5 can be formulated as the MWC problem of finding a maximum weight clique in the complement of the P5-graph. All the results for the general MWC problem reviewed in Section 6 can be applied for problem P5.

8. Conclusions

We have provided a survey of the results for the interval scheduling problem and its variants. These problems have numerous applications. Further research can be conducted on identifying well-solvable special cases of problem P that are interesting from a practical point of view, and on the development of efficient enumerative and approximate methods.

There is a limited number of publications on problem P2 with arbitrary machine unavailability intervals. This problem deserves further investigation because it can be used for modelling such important applications as satellite data transmitting [35], retail trade [62] and planning cargo loading/unloading processes [78].

Investigation of a feasible domain of integer programming (IP) formulations of the interval scheduling problem is another topic for future research because it can help to solve practical instances of these problems by using existing IP techniques and commercial IP solvers. The on-line or semi on-line versions of the problem are of interest, too, because they are relevant to situations where job parameters become known only upon job arrival. In semi on-line versions, partial information about the jobs to arrive is available such as their number or order of arriving, etc.

Acknowledgements

This research was supported in part by The Hong Kong Polytechnic University under grant number A628 from the *Area of Strategic Development in China Business Services*. M.Y. Kovalyov was in addition supported by INTAS under grant number 03-51-5501. We are grateful to three anonymous referees for their constructive comments on an earlier version of this paper.

References

[1] R.K. Ahuja, T.L. Magnanti, J.B. Orlin, Network Flows—Theory, Algorithms, and Applications, Prentice-hall, Englewood Cliffs, NJ, 1993.
[2] E.M. Arkin, E.L. Silverberg, Scheduling jobs with fixed start and finish times, Discrete Applied Mathematics 18 (1987) 1–8.
[3] L. Babel, A fast algorithm for the maximum weight clique problem, Computing 52 (1994) 31–38.
[4] E. Bach, J. Boyar, L. Epstein, L.M. Favrholdt, T. Jiang, K.S. Larsen, G.-H. Lin, R. van Stee, Tight bounds on the competitive ratio on accomodating sequences for the seat reservation problem, Journal of Scheduling 6 (2003) 131–147.
[5] N. Balakrishnan, J.J. Kanet, S.V. Sridharan, Early/tardy scheduling with sequence dependent setups on uniform parallel machines, Computers and Operations Research 26 (1999) 127–141.
[6] E. Balas, V. Chvátal, J. Nešetril, On the maximum weight clique problem, Mathematics of Operations Research 12 (1987) 522–535.
[7] R. Battiti, M. Protasi, Reactive local search for the maximum clique problem, Algorithmica 29 (2001) 610–637.
[8] A. Bar-Noy, R. Bar-Yehuda, A. Freund, J.S. Naor, B. Schieber, Approximating the throughput of multiple machines in real-time scheduling, SIAM Journal on Computing 31 (2001) 331–352.
[9] A. Bar-Noy, S. Guha, J.S. Naor, B. Schieber, A unified approach to approximating resource allocation and scheduling, Journal of the ACM 48 (2001) 1069–1090.
[10] S. Ben-David, A. Borodin, R. Karp, G. Tardos, A. Wigderson, On the power of randomization in on-line algorithms, Algorithmica 11 (1994) 2–14.
[11] R. Bhatia, J. Chuzhoy, A. Freund, J. Naor, Algorithmic aspects of bandwidth trading, Lecture Notes in Computer Science 2719 (2003) 751–766.
[12] I.M. Bomze, On standard quadratic optimization problem, Journal of Global Optimization 13 (1998) 369–387.
[13] I.M. Bomze, M. Budinich, P.M. Pardalos, M. Pelillo, The maximum clique problem, in: D.-Z. Du, P.M. Pardalos (Eds.), Handbook of Combinatorial Optimization, Kluwer Academic Publishers, Dordrecht, 1999, pp. 1–74.
[14] I.M. Bomze, Global escape strategies for maximizing quadratic forms over a simplex, Journal of Global Optimization 11 (1997) 325–338.
[15] K.I. Bouzina, H. Emmons, Interval scheduling on identical machines, Journal of Global Optimization 9 (1996) 379–393.
[16] J. Boyar, K.S. Larsen, The seat reservation problem, Algorithmica 25 (1999) 403–417.
[17] M. Brehob, S. Wagner, E. Torng, R. Enbody, Optimal replacement is NP-hard for nonstandard caches, IEEE Transactions on Computers 53 (2004) 73–76.
[18] P. Brucker, L. Nordmann, The *k*-track assignment problem, Computing 52 (1994) 97–122.
[19] R. Canetti, S. Irani, Bounding the power of preemption in randomized scheduling, SIAM Journal on Computing 27 (1998) 993–1015.
[20] M.C. Carlisle, E.L. Lloyd, On the *k*-coloring of intervals, Discrete Applied Mathematics 59 (1995) 225–235.
[21] M.W. Carter, C.A. Tovey, When is the classroom assignment problem hard? Operations Research 40 (1992) 28–39.
[22] B. Chen, R. Hassin, M. Tzur, Allocation of bandwidth and storage, IIE Transactions 34 (2002) 501–507.
[23] Z.Z. Chen, T. Jiang, G.H. Lin, R. Rizzi, J.J. Wen, D. Xu, Y. Xu, More reliable protein NMR peak assignment via improved 2-interval scheduling, Lecture Notes in Computer Science 2832 (2003) 580–592.
[24] Z.-Z. Chen, T. Jiang, G. Lin, J. Wen, D. Xu, J. Xu, Y. Xu, Approximation algorithms for NMR spectral peak assignment, Theoretical Computer Science 299 (2003) 211–229.
[25] A. Chmeiss, P. Jégou, A generalization of chordal graphs and the maximum clique problem, Information Processing Letters 62 (1997) 61–66.
[26] M. Chudnovsky, G. Cornuéjols, X. Liu, P. Seymour, K. Vušković, Recognizing Berge graphs, Combinatorica 25 (2005) 143–187.
[27] M. Chudnovsky, N. Robertson, P.D. Seymour, R. Thomas, The strong perfect graph theorem, Annals of Mathematics 164 (2006) 51–229.
[28] M. Chudnovsky, N. Robertson, P.D. Seymour, R. Thomas, Progress on perfect graphs, Mathematical Programming Series B 97 (2003) 405–422.
[29] G.L. Dantzig, D.R. Fulkerson, Minimizing the number of tankers to meet a fixed schedule, Naval Research Logistics Quarterly 1 (1954) 217–222.
[30] M.C. Dijkstra, L.G. Kroon, J.A.E.E. van Nunen, M. Salomon, A DSS for capacity planning of aircraft maintenance personnel, International Journal of Production Research 23 (1991) 69–78.
[31] V.R. Dondeti, H. Emmons, Fixed job scheduling with two types of processors, Operations Research 40 (1992) S76–S85.
[32] S. Even, A. Itai, A. Shamir, On the complexity of timetable and multicommodity flow problems, SIAM Journal on Computing 5 (1976) 691–703.
[33] T. Erlebach, F.C.R. Spieksma, Simple algorithms for a weighted interval selection problem, Lecture Notes in Computer Science 1969 (2001) 228–240.
[34] T. Erlebach, F.C.R. Spieksma, Interval selection: Applications, algorithms, and lower bounds, Journal of Algorithms 46 (2003) 27–53.
[35] U. Faigle, W. Kern, W.M. Nawijn, A greedy on-line algorithm for the *k*-track assignment problem, Journal of Algorithms 31 (1999) 196–210.
[36] U. Faigle, W.M. Nawijn, Note on scheduling intervals on-line, Discrete Applied Mathematics 58 (1995) 13–17.
[37] D.B.C. Faneyte, F.C.R. Spieksma, G.J. Woeginger, A branch-and-price algorithm for a hierarchical crew scheduling problem, Naval Research Logistics 49 (2002) 743–759.
[38] S. Fenet, C. Solnon, Searching for maximum cliques with ant colony optimization, Lecture Notes in Computer Science 2611 (2003) 236–245.
[39] M. Fischetti, S. Martello, P. Toth, Approximation algorithms for fixed job schedule problems, Operations Research 40 (1992) S96–S108.
[40] A. Frank, On chain and antichain families of a partially ordered set, Journal of Combinatorial Theory Series B 29 (1980) 176–184.
[41] K. Fujisawa, M. Kojima, K. Nakata, Exploiting sparsity in primal-dual interior-point methods for semidefinite programming, Mathematical Programming 79 (1997) 235–253.
[42] V. Gabrel, Scheduling jobs within time windows on identical parallel machines: New model and algorithms, European Journal of Operational Research 83 (1995) 320–329.

[43] M.R. Garey, D.S. Johnson, G.L. Miller, Papadimitriou, The complexity of coloring circular arcs and hords, SIAM Journal on Algebraic Discrete Methods 1 (1980) 216–227.

[44] F. Gavril, Maximum weight independent sets and cliques in intersection graphs of filaments, Information Processing Letters 73 (2000) 181–188.

[45] I. Gertsbakh, H.I. Stern, Minimal resources for fixed and variable job schedules, Operations Research 18 (1978) 68–95.

[46] L.E. Gibbons, D.W. Hearn, P.M. Pardalos, M.V. Ramana, Continuous characterizations of the maximum clique problem, Mathematics of Operations Research 22 (1997) 754–768.

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

[48] M. Grötschel, L. Lovász, A. Schrijver, Geometric Algorithms and Combinatorial Optimization, Springer, Berlin, 1993.

[49] M. Grötschel, L. Lovász, A. Schrijver, The ellipsoid method and its consequences in combinatorial optimization, Combinatorica 1 (1981) 169–197.

[50] M. Grötschel, L. Lovász, A. Schrijver, Corrigendum to our paper, The ellipsoid method and its consequences in combinatorial optimization, Combinatorica 4 (1984) 291–295.

[51] U.I. Gupta, D.T. Lee, J.Y.-T. Leung, An optimal solution for the channel-assignment problem, IEEE Transactions on Computers 28 (1979) 807–810.

[52] J.J. Harms, A simple optimal algorithm for scheduling variable-sized requests, Information Processing Letters 68 (1998) 291–293.

[53] A. Hashimoto, J. Stevens, Wire routing by optimizing channel assignment with large apertures, in: Proceedings of the 8th Design Automation Conference, 1971, pp. 155–169.

[54] R.B. Heady, Z. Zhu, Minimizing the sum of job earliness and tardiness in a multi-machine system, International Journal of Production Research 36 (1998) 1619–1632.

[55] K. Hiraishi, E. Levner, M. Vlach, Scheduling of parallel identical machines to maximize the weighted number of just-in-time jobs, Computers and Operations Research 29 (2002) 841–848.

[56] J.Y. Hsiao, C.Y. Tang, R.S. Chang, An efficient algorithm for finding a maximum weight 2-independent set on interval graphs, Information Processing Letters 43 (1992) 229–235.

[57] W.L. Hsu, K.H. Tsai, A linear time algorithm for the two-track assignment problem, in: Proceedings of 27th Allerton Conference on Communication, Control and Computing, 1989, pp. 291–300.

[58] D.S. Johnson, M.A. Trick (Eds.), Cliques Coloring and Satisfiability, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 26, American Mathematical Society, DIMACS, 1996.

[59] J.M. Keil, On the complexity of scheduling tasks with discrete starting times, Operations Research Letters 12 (1992) 293–295.

[60] L.G. Khachiyan, A polynomial algorithm in linear programming, Dokladi Akademii Nauk SSSR 244 (1979) 1093–1096.

[61] A.J.W. Kolen, L.G. Kroon, On the computational complexity of (maximum) class scheduling, European Journal of Operational Research 54 (1991) 23–28.

[62] A.J.W. Kolen, J.K. Lenstra, Combinatorics in operations research, in: R. Graham et al. (Eds.), Handbooks of Combinatorics, vol. II. 1995, pp. 1901–1904.

[63] J. Korst, E. Aarts, J.K. Lenstra, J. Wessels, Periodic assignment and graph colouring, Discrete Applied Mathematics 51 (1994) 291–305.

[64] L.G. Kroon, M. Salomon, L.N. van Wassenhove, Exact and approximation algorithms for the tactical fixed interval scheduling problem, Operations Research 45 (1997) 624–638.

[65] A. Lann, G. Mosheiov, A note on the maximum number of on-time jobs on parallel identical machines, Computers and Operations Research 30 (2003) 1745–1749.

[66] E.L. Lawler, Combinatorial Optimization: Networks and Matroids, Holt, Rinehart, and Winston, New York, 1976.

[67] D.T. Lee, M. Sarrafzadeh, Maximum independent set of a permutation graph in K-tracks, Lecture Notes in Computer Science 557 (1991) 2–11.

[68] R.J. Lipton, A. Tomkins, Online interval scheduling, in: Proceedings of 5th Annual ACM-SIAM Symposium on Discrete Algorithms, ACM press, 1994, pp. 302–311.

[69] M. Locatelli, I.M. Bomze, M. Pelillo, The combinatorics of pivoting for the maximum weight clique, Operations Research Letters 32 (2004) 523–529.

[70] D. Long, M. Thakur, Scheduling real-time disk transfers for continuous media applications, in: 12th IEEE Symposium on Mass Storage Systems, 1993, pp. 227–232.

[71] E. Marchiori, Genetic, iterated and multistart local search for the maximum clique problem, Lecture Notes in Computer Science 2279 (2002) 112–121.

[72] S. Martello, P. Toth, A heuristic approach to the bus driver scheduling problem, European Journal of Operational Research 24 (1986) 106–117.

[73] H. Miyazawa, T. Erlebach, An improved randomized on-line algorithm for a weighted interval selection problem, Journal of Scheduling 7 (2004) 293–311.

[74] T.S. Motzkin, E.G. Straus, Maxima for graphs and a new proof of a theorem of Turán, Canadian Journal of Mathematics 17 (1965) 533–540.

[75] J. Naor, H. Shachnai, T. Tamir, Real-time scheduling with a budget, Lecture Notes in Computer Science 2719 (2003) 1123–1137.

[76] G.L. Nemhauser, L.E. Trotter, Properties of vertex packings and independence system polyhedra, Mathematical Programming 6 (1974) 48–61.

[77] G.L. Nemhauser, L.E. Trotter, Vertex packings: Structural properties and algorithms, Mathematical Programming 8 (1975) 232–248.

[78] C.T. Ng, M.Y. Kovalyov, T.C.E. Cheng, A graph-theoretic approach to interval scheduling on dedicated unrelated parallel machines, submitted for publication.

[79] M. Pal, G.P. Bhattacharjee, A sequential algorithm for finding a maximum weight K-independent set on interval graphs, International Journal of Computer Mathematics 60 (1996) 205–214.

[80] P.M. Pardalos, G.P. Rodgers, A branch and bound algorithm for the maximum clique problem, Computers and Operations Research 19 (1992) 363–375.

[81] S.A. Vavasis, Y. Ye, A primal–dual interior point method whose running time depends only on the constraint matrix, Mathematical Programming 74 (1996) 79–120.

[82] A. Saha, M. Pal, Maximum weight k-independent set problem on permutation graphs, International Journal of Computer Mathematics 80 (2003) 1477–1487.

[83] M. Sarrafzadeh, R.D. Lou, Maximum *k*-covering of weighted transitive graphs with applications, Algorithmica 9 (1993) 84–100.

[84] A. Schrijver, Polyhedral proof methods in combinatorial optimization, Discrete Applied Mathematics 14 (1986) 111–133.

[85] S.S. Seiden, Randomized online interval scheduling, Operations Research Letters 220 (1998) 171–177.

[86] L. Sha, T. Abdelzaher, K.-E. Arzen, A. Cervin, T. Baker, A. Burns, G. Buttazzo, M. Caccamo, J. Lehoczky, A.K. Mok, Real time scheduling theory: A historical perspective, Real-Time Systems 28 (2004) 101–155.

[87] N.Z. Shor, Convergence rate of the gradient descent method with dilatation of the space, Kibernetika 2 (1970) 80–85.

[88] N.Z. Shor, Dual quadratic estimates in polynomial and Boolean programming, in: P.M. Pardalos, J.B. Rosen (Eds.), Computational Methods of Global Optimization, Annals of Operations Research, vol. 25, 1990, pp. 163–168.

[89] D. Sleator, R. Tarjan, Amortized efficiency of list update and paging rules, Communications ACM 28 (1970) 202–208.

[90] F. Sivrikaya-Serifoglu, G. Ulusoy, Parallel machine scheduling with earliness and tardiness penalties, Computers and Operations Research 26 (1999) 773–787.

[91] F.C.R. Spieksma, On the approximability of an interval scheduling problem, Journal of Scheduling 2 (1999) 215–227.

[92] E. Torng, A unified analysis of paging and caching, Algorithmica 20 (1998) 175–200.

[93] V. Veeramachaneni, P. Berman, W. Miller, Aligning two fragmented sequences, Discrete Applied Mathematics 127 (2003) 119–143.

[94] H. Waterer, E.L. Johnson, P. Nobili, M.W.P. Savelsbergh, The relation of time indexed formulations of single machine scheduling problems to the node packing problem, Mathematical Programming Series A 93 (2002) 477–494.

[95] G.J. Woeginger, On-line scheduling of jobs with fixed start and end times, Theoretical Computer Science 130 (1994) 5–16.

[96] M. Yannakakis, F. Gavril, The maximum *k*-colorable subgraph problem for chordal graphs, Information Processing Letters 24 (1987) 133–137.