www.palgrave-journals.com/jors/

A graph-theoretic approach to interval scheduling on dedicated unrelated parallel machines

Chi To Ng[1]*, Tai Chiu Edwin Cheng[1], Andrei M Bandalouski[2], Mikhail Y Kovalyov[2] and Sze Sing Lam[3]

[1]The Hong Kong Polytechnic University, Kowloon, Hong Kong; [2]National Academy of Sciences of Belarus, Minsk, Belarus; and [3]The Open University of Hong Kong, Kowloon, Hong Kong

We study the problem of scheduling n non-preemptive jobs on m unrelated parallel machines. Each machine can process a specified subset of the jobs. If a job is assigned to a machine, then it occupies a specified time interval on the machine. Each assignment of a job to a machine yields a value. The objective is to find a subset of the jobs and their feasible assignments to the machines such that the total value is maximized. The problem is NP-hard in the strong sense. We reduce the problem to finding a maximum weight clique in a graph and survey available solution methods. Furthermore, based on the peculiar properties of graphs, we propose an exact solution algorithm and five heuristics. We conduct computer experiments to assess the performance of our and other existing heuristics. The computational results show that our heuristics outperform the existing heuristics.

Journal of the Operational Research Society (2014) **65**(10), 1571–1579. doi:10.1057/jors.2013.109
Published online 18 September 2013

Keywords: fixed interval scheduling; cargo loading/unloading; maximum weight clique; interval graph

1. Introduction and literature review

The problem studied in this paper arises in many decision-making situations such as assignment of transport devices to loading/unloading terminals in ports, work planning of personnel in companies, bandwidth allocation of communications channels, printed circuit board manufacturing, gene identification, and examining computer memory structures. We will give the related references after presenting the problem formulation below.

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 $U_l = U_{1l} \cup \cdots \cup U_{u_l l}$, $l=1, \ldots, m$. A job j assigned to machine l yields a weight (value) w_{jl} if it is processed on the machine for a *fixed processing interval* $I_{jl}=(s_{jl}, d_{jl}]$, $s_{jl}<d_{jl}$, $j=1, \ldots, n$, $l=1, \ldots, m$. Each job is to be either processed once or not at all (ie, rejected). All the data are assumed to be non-negative integers.

A schedule is characterized by the set of processed jobs and their assignments to the machines. A schedule and the corresponding assignments are feasible if the following constraints are satisfied: (a) if job j is assigned to machine l for processing within the interval I_{jl}, then $j \in N_l$ and $I_{jl} \cap U_l = \varnothing$, $j=1, \ldots, n$, $l=1, \ldots, m$; and (b) the processing intervals of the jobs assigned to the same machine do not overlap. The problem is to find a feasible schedule maximizing the total weight (value) of the processed jobs. We denote this problem as ISDU, which stands for Interval Scheduling on Dedicated Unrelated parallel machines.

Observe that if $I_{jl} \cap U_l \neq \varnothing$, then job j cannot be processed on machine l. Therefore, all such jobs can be removed from the set N_l. Let us remove all such jobs from each set N_l. After this modification, the relation $I_{jl} \cap U_l = \varnothing$ is satisfied for $j \in N_l$, $l=1, \ldots, m$, which means that the unavailability intervals do not affect the job assignment. Therefore, from now on, we assume without loss of generality that there is no unavailability interval on each machine. In this case, there are at most $4mn$ numbers in the input of the problem ISDU. They are the job indices from the sets N_l, and the values w_{jl}, s_{jl}, and d_{jl}, $j \in N_l$, $l=1, \ldots, m$.

Kolen *et al* (2007) and Kovalyov *et al* (2007) provide recent surveys of studies on interval scheduling. Below we briefly review the most relevant results.

Problem ISDU is a generalization of the problem studied by Arkin and Silverberg (1987). The difference is that in the latter problem $w_{jl}=w_j$, $s_{jl}=s_j$, and $d_{jl}=d_j$ for each job j, and $U_l=\varnothing$, $l=1, \ldots, m$, that is, the machines are identical and continuously available. We denote this problem as ISDI (Interval Scheduling on Dedicated Identical parallel machines) and its special case where each machine can process any job, that is, $N_l=\{1, \ldots, n\}$, $U_l=\varnothing$, $l=1, \ldots, m$, as ISI (Interval Scheduling on Identical parallel machines). Arkin and Silverberg prove that

**Correspondence: Chi To Ng, Department of Logistics and Maritime Studies, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong.*
E-mail: lgtctng@polyu.edu.hk

problem ISDI 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 network with O(mn^{m+1}) arcs. Arkin and Silverberg suggest several solution approaches for problem ISI, the best of which can be implemented in O(n^2 log n) time. Bouzina and Emmons (1996) suggest improved algorithms for problem ISI and its special case where all the job weights are unit. These algorithms run in O(mn log n) and O(n max{log n, m}) time, respectively. For the unit-weight case, Faigle and Nawijn (1995) use the same algorithm as that of Bouzina and Emmons. They highlight that the algorithm is an optimal online algorithm because it assigns a newly arrived job by using information only about the jobs that have arrived so far. In the considered online model, they assume that a non-completed job can be rejected. The best existing (off-line) algorithm for problem ISI with unit weights is due to Carlisle and Lloyd (1995). It runs in O(n log n) time. Carlisle and Lloyd also present an algorithm for the general problem ISI with the same time complexity as that of Bouzina and Emmons.

Problem ISDU is polynomially reducible to the *Weighted Job Interval Selection Problem on One Machine with Arbitrary Weights (WJISP*$_1$) studied by Erlebach and Spieksma (2003). In problem WJISP$_1$, there is a single machine and several jobs. A collection of time intervals is associated with each job. Let N be the total number of intervals. The objective is to select a maximum weight subset of the intervals such that (i) no two selected intervals intersect and (ii) at most one interval is selected for each job. Given an instance of problem ISDU, the corresponding instance of problem WJISP$_1$ can be obtained by shifting each interval I_{jl} to start $(l-1)T$ time units later, $l=1, \ldots, m$, where T is the length of the planning horizon in the corresponding instance of problem ISDU. Spieksma (1999) proves that problem WJISP$_1$ with unit weights is strongly NP-hard even if the length of each interval is equal to 2 and at most two intervals intersect at each time instant. Furthermore, this problem cannot have a *Polynomial Time Approximation Scheme*, unless $P=NP$. It follows from his proof that these results also apply to problem ISDU under the same conditions. Berman and DasGupta (2000) develop an O(n log n) time ρ-*approximation algorithm* with $\rho=1/2$ for WJISP$_1$, which delivers a solution with a value at least ρ times the value of an optimal solution for any instance of this problem. This approximation result also applies to problem ISDU.

Note that the specificity of problem ISDU can be used to derive better solution algorithms than those for the more general problem WJISP$_1$.

Recent studies of interval scheduling problems concentrate on online versions of the interval scheduling problem, and heuristic and meta-heuristic solution approaches. Epstein and Levin (2010) present online randomized algorithms for an online interval selection problem and evaluate the competitive ratios of such algorithms. Rossi *et al* (2010) develop a greedy heuristic and a grouping genetic algorithm for the problem ISI with the additional constraint that limits the difference between the machine completion and start times. Eliiyi and Azizoglu (2011) study a more constrained problem, in which the total processing load of each machine is also limited. They suggest a filtered beam search algorithm and a heuristic that generates and evaluates “promising” sets of selected jobs. Krumke *et al* (2011) present greedy rules for the online version of the problem that differs from problem ISI in that $d_j=s_j+p_j/v_l$, where p_j is the processing time of job j and v_l is the speed of machine l. They show that any online algorithm using these rules is 2-competitive. Zheng *et al* (2013) investigate the impact of the look ahead ability of online heuristics for problem WJISP$_1$ with unit length intervals to improve their competitive ratios.

Interval scheduling problems appear in many applications. For example, consider a cargo port with several loading/unloading ship terminals. An incoming ship is assigned to a terminal and occupies it until its loading/unloading is completed. A ship has a fixed arrival time or a fixed departure time. Its loading/unloading time is determined by its type and the type of the terminal. Each assignment of a ship to a terminal yields a value. A ship can be assigned to a specified subset of terminals. Terminals are available within specified time intervals. The problem is to make a feasible assignment of a subset of a given set of incoming ships that yields the maximum total value. Note that some ships can be rejected by an optimal solution. These ships can be considered for future planning, or alternatively, their values can be increased to force them to be in the optimal subset of the assigned ships.

Other applications provided in the literature include scheduling of bus drivers, class scheduling, VLSI chips design, assignment of incoming aircraft to gates, work planning of aircraft maintenance personnel, bandwidth allocation of communications channels, planning of satellite photography, printed circuit board manufacturing, genom comparison in molecular biology, spectroscopical methods in the identification of protein-encoding genes, solving problems of disparity between processor and memory speeds in computers, real-time computer task scheduling, fleet assignment and scheduling, hotel room assignment for reservations, (see, eg, Kolen *et al*, 2007; Kovalyov *et al*, 2007; Lee *et al*, 2012; Demaine *et al*, 2013; Dishan *et al*, 2013, and Li *et al*, 2013).

Recently, we encountered problem ISDU in an accommodation rental business. There, an information system is under development which is intended to solve two major problems: (1) dynamic determination of accommodation rental prices and control of accommodation quantities by converting one type of accommodation into another based on the analysis of price-sensitive demand for renting various accommodation categories, and (2) given prices and accommodation quantities, assignment of collected booking requests to specific accommodations over time. The second problem is a special case of the problem ISDU, in which jobs are booking requests for accommodations of the same type, machines are accommodations of this type, $s_{jl}=s_j$, $d_{jl}=d_j$ and $w_{jl}=\sum_{t=s_j}^{d_j-1} c_{t,d_j-s_j}$, $l=1, \ldots, m$, where $c_{t,L}$ is the accommodation price for the night between days t and $t+1$ depending on the length of stay L. Furthermore, $U_l \neq \varnothing$, $l=1, \ldots, m$, because accommodations can be occupied in some periods by earlier bookings.

Practical problems, where problem ISDU appears as a subproblem, are usually handled by MIP solvers, heuristics and meta-heuristics. Multi-agent methodology appears to be a useful tool for handling these problems in online and dynamic environments; see, for example, Wauters *et al* (2011), Renna (2011) and Nejad *et al* (2011). In this paper, we propose a graph-theoretic model and exact and heuristic algorithms based on this model. We experimentally prove their efficiency. Thus, they can be used as an alternative to the commercial software for solving practical problems.

The following section discusses some simple variants of problem ISDU that can be applied in practice. In Section 3 we reduce problem ISDU to the problem of finding a maximum weight clique in a specially constructed graph. We denote this problem as MWC(ISDU) and the problem of finding a maximum weight clique in an arbitrary graph as MWC. The required definitions from the graph theory are given in Sections 2 and 3. All the existing techniques for solving problem MWC can be used for solving problem ISDU. Among these techniques there exist polynomial time algorithms for specific graph classes. Furthermore, for some of these classes, there exist polynomial time algorithms for recognizing the membership of an arbitrary graph in such a class. These algorithms can be used to efficiently solve some instances of problem ISDU. In Section 4 we describe a specific exact algorithm for problem MWC(ISDU) based on an enumeration of the maximal cliques in graphs that describe interval intersections on individual machines. Although the algorithm is not polynomial, it is efficient for some special cases or particular instances of problem MWC(ISDU). In Section 5 we provide five polynomial time heuristic algorithms for problem ISDU. We report the results of computer experiments to test the performance of our and other existing heuristics for problem ISDU in Section 6. We conclude the paper with a summary of the results and suggestions for future research in the last section.

2. Simple variants of problem ISDU

The following problems are of interest and relevance to some practical situations.

Problem 1 Solving problem ISDU with the restriction that at most one job is assigned to each machine.

Problem 2 Finding a feasible solution to problem ISDU such that the number of machines occupied by at least one job is maximized. An equivalent formulation here is to maximize the number of processed jobs under the restriction that at most one job is assigned to each machine.

Problem 1 is a special case of problem ISDU when all the intervals on the same machine are pairwise intersecting. Its solution can be used to identify the most valuable assignment for a subset of jobs that compete for the available machine resources. A solution to Problem 2 provides additional information to a decision maker in situations where there are machines not processing any job in an optimal solution to problem ISDU.

Let us introduce some graph terminology. A graph $G=G(V, E)$ is characterized by the set of *vertices* V and the set of *edges* $E \subseteq V \times V$. An edge is denoted by an undirected pair (i, j), where i and j are *adjacent* vertices *connected* by this edge.

Problem 1 reduces to the following *weighted bipartite matching problem.* Introduce a *bipartite graph* $G(X,Y)$, where $X=X_1 \cup X_2$, $X_1 \cap X_2=\varnothing$. The set of vertices $X_1=\{1, \ldots, n\}$ corresponds to the jobs, the set of vertices $X_2=\{1, \ldots, m\}$ corresponds to the machines, and there is an edge $(j, l) \in Y$, $j \in X_1$, $l \in X_2$, with weight w_{jl} if and only if job j can be processed on machine l, that is, $j \in N_l$. A *matching* in the graph $G(X, Y)$ is a collection of its pairwise disjoint edges. Problem 1 is equivalent to finding a matching in the graph $G(X, Y)$ with the maximum total weight. The problem can be solved in $O(n^3)$ time (see, eg, Lawler (1976)).

Problem 2 reduces to the *maximum cardinality bipartite matching problem*, which is, in fact, the weighted bipartite matching problem described above with unit weights. It is solvable in $O(n^{2.5})$ time (see Hopcroft and Karp, 1973).

3. A reduction to the maximum weight clique problem

Consider a finite graph $G=G(V, E)$ without self-loops and parallel edges. Assume that a weight w_i is associated with each vertex of graph G. For $Z \subseteq V$, an *induced subgraph* $G(Z)$ of graph $G(V, E)$ is defined by $G(Z)=G(Z, E')$, where $E'=\{(i, j) \mid i, j \in Z, (i, j) \in E\}$. Graph G is *complete* if all its vertices are pairwise adjacent. Given graph $G(V, E)$, a set of vertices $C \subseteq V$ is a *clique* if $G(C)$ is complete. The *weight of a clique* is the total weight of its vertices. The *Maximum Weight Clique (MWC) problem* is to find a maximum weight clique in a given graph.

Our reduction of problem ISDU to an MWC problem can be described as follows: Introduce an *m-layer graph* $G(V, E)$ (see Figure 1 for an example).

The set of vertices is $V=V_1 \cup \cdots \cup V_m$, where set V_l, $|V_l|=|N_l|$, determines *layer l* containing intervals I_{jl} for jobs $j \in N_l$ on machine l, $l=1, \ldots, m$. Weight w_{jl} is associated with each vertex I_{jl}. In Figure 1, the weight of a vertex is put in the round brackets next to it. In the sequel, we do not distinguish a vertex and the corresponding interval.

The set of edges is $E=E_1 \cup \cdots \cup E_{m+1}$, where $E_l=\{(I_{il}, I_{jl}) \mid I_{il} \cap I_{jl}=\varnothing\}$, $l=1, \ldots, m$, and $E_{m+1}=\{(I_{il}, I_{jq}) \mid l \neq q,\ i \neq j,\ l, q=1, \ldots, m,\ i, j=1, \ldots, n\}$. Verbally, 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 ISDU in the way described above as an *ISDU-graph.* Its *adjacency matrix* $\|a_{IJ}\|$ of dimension $(\sum_{l=1}^{m}|V_l|) \times (\sum_{l=1}^{m}|V_l|)$ can be easily constructed in $O(m^2n^2)$ time based on its definition. Here $a_{IJ}=1$ if edge

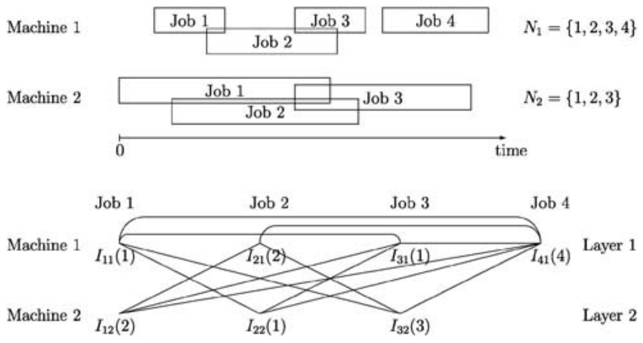

Figure 1 Job intervals on two machines and the corresponding ISDU-graph.

$(I, J) \in E$, and $a_{IJ} = 0$ otherwise. In some cases, it is not necessary to explicitly enumerate all the edges of the set E_{m+1}. The ISDU-graph can be represented by a collection of graphs $G(V_l, E_l)$, $l = 1, \ldots, m$, and a simple rule that determines if there is an edge between any given two vertices of different layers.

Each graph $G(V_l, E_l)$ can be constructed in $O(|V_l| \log |V_l|)$ time as follows: To facilitate discussion, let $(s_1, d_1], \ldots, (s_k, d_k]$ be all of the intervals from V_l. Renumber them such that $s_1 \leqslant s_2 \leqslant \cdots \leqslant s_k$. For each i, $i = 1, \ldots, k$, find index r_i such that $d_i \leqslant s_{r_i}$ and $d_i > s_{r_i - 1}$. Index r_i can be found in O(log k) time by a bisection search over the range $i+1, i+2, \ldots, k$. It is easy to see that interval i does not intersect with intervals $r_i, r_i + 1, \ldots, k$. Therefore, vertices $r_i, r_i + 1, \ldots, k$ are all adjacent to vertex i in $G(V_l, E_l)$ such that there is an edge $(i, j) \in E_l$, $j = r_i, r_i + 1, \ldots, k$.

Given ISDU-graph $G(V, E)$, let us associate an assignment of jobs to the 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 maximum weight clique in the ISDU-graph $G(V, E)$ determines a schedule for the corresponding problem ISDU.*

There exists a vast body of literature on solving the MWC problem; see Kovalyov *et al* (2007). The MWC problem is polynomially solvable on the class of *perfect graphs*, which includes *interval* and *co-interval graphs*.

Note that each graph $G(V_l, E_l)$ is a co-interval graph. However, an ISDU-graph made of these graphs can be neither interval nor co-interval. In general, the ISDU-graph is not perfect. For example, the graph in Figure 1 is not perfect. The Berge's conjecture proved by Chudnovsky *et al* (2006) can be used to identify whether a particular ISDU-graph is perfect or not.

We denote the MWC problem with an ISDU-graph as *problem MWC(ISDU)*.

4. Solving problem MWC(ISDU) through enumeration of maximal cliques in graphs $G(V_l, E_l)$

Let an ISDU-graph $G(V, E)$ be given. A *maximal clique* is a clique $C \subseteq V$ such that $C \cup \{i\}$ is not a clique for any $i \in V \backslash C$. Our approach to solving problem MWC(ISDU) is based on the following obvious statement.

Statement 2 *Let $C^* \subseteq V$ be an arbitrary optimal solution to problem MWC(ISDU) and $X_l^* = V_l \cap C^*$ be the set of vertices of layer V_l in C^*, $l = 1, \ldots, m$. Then there exists a maximal clique A_l^* in the graph $G(V_l, E_l)$ such that $X_l^* \subseteq A_l^*$, $l = 1, \ldots, m$.*

Proof Assume the contrary: There exists index $l \in \{1, \ldots, m\}$ such that set X_l^* is not a subset of a maximal clique in the graph $G(V_l, E_l)$. Then, since every clique is a subset of a maximal clique in the same graph, X_l^* is not a clique in the graph $G(V_l, E_l)$. It follows that X_l^*, and hence C^*, contains at least two vertices that are not connected by an edge. Therefore, C^* is not a clique, which is a contradiction.

Let $w(Z)$ denote the total weight of vertices in a set $Z \subseteq V$. Consider an arbitrary optimal solution to problem MWC (ISDU). Let $A^* = A^* \cup \cdots \cup A_m^*$, where A_l^* is the maximal clique in the graph $G(V_l, E_l)$ that contains all vertices of this optimal solution from the graph $G(V_l, E_l)$, $l = 1, \ldots, m$. If set A^* is a clique, then it is an optimal solution to problem MWC(ISDU). Assume that set A^* is not a clique. Then problem MWC(ISDU) reduces to finding a subset $Q^* \subset A^*$ such that $A^* \backslash Q^*$ is a clique and

$$w(Q^*) = \min\{w(Q) \mid Q \subset A^*, A^* \setminus Q \text{ is a clique}\}.$$

We have $C^* = A^* \backslash Q^*$, where C^* is an optimal solution to problem MWC(ISDU).

Let us analyse why set A^* is not a clique. The only reason is that in A^* there are vertices with the same job index and different machine indices, which are not connected in the original graph $G(V, E)$. Let $J_j = \{I_{jl} | I_{jl} \in A^*, \exists r : r \neq l, I_{jr} \in A^*\}$ be the set of all the vertices in A^* with the same job index j. Vertices in the set J_j, $J_j \neq \varnothing$, are pairwise disconnected and each vertex in this set is connected to each vertex in $A^* \backslash J_j$. Therefore, it is optimal to remove from A^* all the vertices of J_j but one, with weight $w_{jl_j} = \max\{w_{jl} \mid I_{jl} \in J_j\}$. Thus, we have shown that $Q^* = \cup_{j=1}^{n} (J_j \setminus \{I_{jl_j}\})$.

We can use the following algorithm to solve problem MWC (ISDU). In this algorithm, all combinations $(A_1, \ldots, A_m)$ of maximal cliques A_l are enumerated, where A_l is a maximal clique in the graph $G(V_l, E_l)$, $l = 1, \ldots, m$. For each m-tuple $(A_1, \ldots, A_m)$, a clique $C_A \subseteq A_1 \cup \cdots \cup A_m$ with the maximum weight is constructed. The best of all these cliques is an optimal solution of problem MWC(ISDU).

Let X_l denote the set of all maximal cliques in the graph $G(V_l, E_l)$.

4.1. Algorithm EMC (Enumeration of Maximal Cliques)

Input: Subgraphs $G(V_l, E_l)$, $l = 1, \ldots, m$, of an ISDU-graph $G(V, E)$.

Output: Maximum weight clique $C^* \subseteq V$.

Step 1: (Initialization) Set $C^* = \varnothing$ and $w(C^*) = -1$.

Step 2: Construct sets X_l of all the maximal cliques in graphs $G(V_l, E_l)$, $l = 1, \ldots, m$. For each maximal clique $B \in \cup_{l=1}^{m} X_l$, calculate its weight $w(B)$.

Step 3: For each m-tuple $(A_1, \ldots, A_m)$, where $A_l \in X_l$ is a maximal clique in the graph $G(V_l, E_l)$, $l = 1, \ldots, m$, perform the following computation.

(a) Calculate the set of vertices $A = \cup_{l=1}^{m} A_l$ and their total weight $w(A) = \sum_{l=1}^{m} w(A_l)$.

(b) For $j = 1, \ldots, n$, find in the set A the set of vertices with the same job index j:

$$J_j(A) = \{I_{jl} \mid I_{jl} \in A_l,\ \exists r : r \neq l,\ I_{jr} \in A_r,\ l,\ r = 1,\ \ldots\ , m\}$$

and the corresponding set of machine indices:

$$M_j(A) = \{l \mid I_{jl} \in J_j(A)\}.$$

Set $J_A = \cup_{j=1}^{n} J_j(A)$ and calculate $w(J_A) = \sum_{j=1}^{n} \sum_{l \in M_j(A)} w_{jl}$. Here we assume that any summation over an empty set produces a zero value.

(c) For each $j = 1, \ldots, n$, determine machine index l_j such that

$$w_{jl_j} = \max\{w_{jl} \mid l \in M_j(A)\}.$$

If the above maximum is taken over an empty set, then $w_{jl_j} = 0$ and $I_{jl_j} = \varnothing$.

(d) Compute a clique

$$C_A = \{A \setminus J_A\} \cup \{I_{1l_1},\ \ldots\ , I_{nl_n}\},$$

which is a candidate for an optimal clique. Calculate

$$w(C_A) = w(A) - w(J_A) + \sum_{j=1}^{n} w_{jl_j}$$

If $w(C_A) > w(C^*)$, then re-set $C^* = C_A$ and $w(C^*) = w(C_A)$.

Step 4: Output C^*.

Let us establish the time complexity of Algorithm EMC. Steps 1 and 4 require O(n) time. Step 2 requires $O(\Sigma_{l=1}^{m} T_l)$ time, where T_l is the time of finding all the maximal cliques and calculating their weights in graph $G(V_l, E_l)$, $l = 1, \ldots, m$. Since there are at most $\prod_{l=1}^{m} |X_l|$ different tuples $(A_1, \ldots, A_m)$, $A_l \in X_l$, $l = 1, \ldots, m$, the number of iterations of Step 3 is $O(\prod_{l=1}^{m} |X_l|)$. Each iteration of this step requires $O(mn)$ time if each clique A_l is represented by a 0–1 vector $x^{(l)} = (x_1^{(l)}, \ldots, x_n^{(l)})$ such that $x_j^{(l)} = 1$ if and only if $I_{jl} \in A_l$. Therefore, the overall time complexity of Algorithm EMC is equal to $O(\Sigma_{l=1}^{m} T_l + mn\Pi_{l=1}^{m} |X_l|)$.

For completeness, we describe below Algorithm SMC(l) that can be used to calculate the set X_l of all the maximal cliques in the graph $G(V_l, E_l)$ and the weights of these cliques.

We need the following terminology. 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 $j_i \to j_{i+1} \in D$, $i = 1, \ldots, k-1$. It is denoted as $(j_1, \ldots, j_k)$. A *maximal path* of a digraph $G(F, D)$ is a path that is not a part of another path in $G(F, D)$. 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 is an arc $i \to j$. A digraph is *acyclic* if every of its vertices is not a successor of itself.

4.2. Algorithm SMC(l) (Set of Maximal Cliques in graph G(V_l, E_l))

Input: Graph $G(V_l, E_l)$.
Output: Set X_l of all the maximal cliques in $G(V_l, E_l)$ and their weights $w(C)$, $C \in X_l$.

Step 1: Renumber intervals I_{jl} in V_l in non-decreasing order of their start times: $s_{j_1 l} \leqslant s_{j_2 l} \leqslant \cdots \leqslant s_{j_{|V_l|} l}$. Introduce new notation for the intervals, $r = I_{j_r l}$, $r = 1, \ldots, |V_l|$.

Step 2: Construct digraph $G(V_l, D_l)$ with the set of vertices $V_l := \{1, \ldots, |V_l|\}$ and the set of arcs D_l such that $i \to j \in D_l$ if and only if $(i, j) \in E_l$ and $i < j$. It is easy to see that

- digraph $G(V_l, D_l)$ is acyclic (because all its edges are oriented from a vertex with a smaller index to a vertex with a larger index),
- there is a one-to-one correspondence between maximal paths in $G(V_l, D_l)$ and maximal cliques in $G(V_l, E_l)$ such that a path $(i_1, i_2, \ldots, i_k)$ is a maximal path in $G(V_l, D_l)$ if and only if the set of vertices $\{i_1, \ldots, i_k\}$ is a maximal clique in $G(V_l, E_l)$ (because the fact that intervals i_{r-1} and i_r do not intersect implies that i_{r-1} does not intersect with i_v, $v = r+1, \ldots, k$, and i_r does not intersect with i_v, $v = 1, \ldots, r-2$).

Step 3: Calculate all the maximal paths in $G(V_l, D_l)$ and their weights as follows: With each vertex, associate parts of all the maximal paths (and their weights) going to this vertex from vertices having no predecessors. At the beginning, associate a single path with each vertex that has no predecessor. This path consists of the vertex itself. Calculate its weight. Label the considered vertices. Furthermore, find an unlabelled vertex of which all the immediate predecessors are labelled. This can be done in O($|V_l|$) if we store with each vertex a variable indicating the number of its labelled immediate predecessors. Let it be vertex i. Calculate the paths (and their weights) going to vertex i. They are all the paths of the immediate predecessors of i

extended by vertex i. This can be done in $O(\sum_{j\in IP_i} x_j)$ time, where IP_i is the set of the immediate predecessors of i and x_j is the number of paths calculated at vertex j. Since all of these intermediate paths are different, their number does not exceed the number of maximal cliques in the graph $G(V_l, E_l)$, that is, $\sum_{j\in IP_i} x_j \leqslant |X_l|$. Label vertex i. Increase by 1 the number of labelled immediate predecessors of each vertex $j \in IS_i$, where IS_i is the set of the immediate successors of i. This can be done in $O(|IS_i|)$ time. Continue until there is an unlabelled vertex. The set of maximal paths is the set of paths associated with vertices having no successors in $G(V_l, D_l)$.

Step 4: For each maximal path in $G(V_l, D_l)$, calculate the corresponding maximal clique in $G(V_l, E_l)$. Its weight is equal to the weight of the corresponding path. Output all the maximal cliques and their weights.

The time complexity of Algorithm SMC(l) is determined by Step 3, which can be implemented in $O(\sum_{i\in V_l}(|V_l| + |X_l| + |IS_i|)) = O(|V_l|^2 + |V_l||X_l| + |D_l|)$ time. Then the time complexity of Algorithm EMC for solving problem MWC (ISDU) can be estimated as $O(n^2 + n\sum_{l=1}^{m}|X_l| + mn\prod_{l=1}^{m}|X_l|) = O(n^2 + mn\prod_{l=1}^{m}|X_l|)$. The space requirement of Algorithm EMC is determined by the representation of the sets of maximal cliques X_l, $l = 1, \ldots, m$, in its Step 2, and the maximum dimension of a tuple $(A_1, \ldots, A_m)$ in its Step 3. Since each clique in $G(V_l, E_l)$ consists of at most n vertices, it is easy to see that the space requirement of Algorithm EMC is $O(n\sum_{l=1}^{m}|X_l|)$.

Algorithm EMC is polynomial in n if each graph $G(V_l, E_l)$ has the property that the total number of its maximal cliques is bounded by a polynomial of n. Classes of graphs having this property are discussed by Rosgen and Stewart (2007). The class of co-interval graphs does not possess this property. For example, there are $8 + 2^{(n-9)/3}$ maximal cliques in the co-interval graph defined by the set of intervals $\{[i-1, i+1], [i, i+1], [i+1, i+2] \mid i = 1 + 3(k-1),\ k = 1, \ldots, n/3\}$. However, specific graphs $G(V_l, E_l)$ can have a polynomial number of maximal cliques.

A trivial case appears when all the intervals on the same machine are mutually non-intersecting. In this case, there is a single maximal clique in each graph $G(V_l, E_l)$ (being the set V_l of its vertices), Algorithm SMC(l) is not needed and Algorithm EMC will run in $O(mn)$ time.

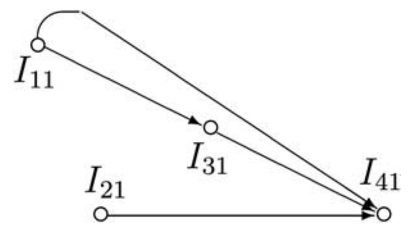

Figure 2 Digraph $G(V_1, D_1)$.

Example Consider problem ISDU given in Figure 1. The digraph $G(V_1, D_1)$ constructed by Algorithm SMC(1) is given in Figure 2.

The digraph $G(V_2, D_2)$ consists of isolated vertices I_{12}, I_{22}, and I_{32}. The sets of maximal cliques are $X_1 = \{\{I_{11}, I_{31}, I_{41}\}, \{I_{21}, I_{41}\}\}$ and $X_2 = \{\{I_{12}\}, \{I_{22}\}, \{I_{32}\}\}$.

In Step 3 of Algorithm EMC, the following cliques of the ISDU-graph will be constructed:

$$C_1 = \{I_{31}, I_{41}, I_{12}\}, \quad w(C_1) = 1 + 4 + 2 = 7,$$

$$C_2 = \{I_{11}, I_{31}, I_{41}, I_{22}\}, \quad w(C_2) = 1 + 1 + 4 + 1 = 7,$$

$$C_3 = \{I_{11}, I_{41}, I_{32}\}, \quad w(C_3) = 1 + 4 + 3 = 8,$$

$$C_4 = \{I_{21}, I_{41}, I_{12}\}, \quad w(C_4) = 2 + 4 + 2 = 8,$$

$$C_5 = \{I_{41}, I_{21}\}, \quad w(C_5) = 4 + 2 = 6,$$

$$C_6 = \{I_{21}, I_{41}, I_{32}\}, \quad w(C_6) = 2 + 4 + 3 = 9.$$

The maximum weight clique is C_6. In the corresponding optimal schedule, jobs 2 and 4 are processed on machine 1 and job 3 on machine 2. Job 1 is rejected.

Remark If a job can be processed only on one machine and the corresponding interval does not intersect with other intervals on this machine, then this job has to be assigned to this interval in any optimal solution. The size of the original problem can be reduced by removing this job from the input. In our example, job 4 is such a job.

5. Heuristic algorithms

The following *greedy* heuristics, denoted as G1-MW, G2-MW, G1-MTW, G2-MTW and G3, can be used to construct an approximate solution to problem MWC(ISDU) with ISDU-graph $G(V, E)$.

Heuristic G1-MW labels an interval with maximum weight and iteratively expands a clique of labelled intervals by adding an appropriate unlabelled interval with maximum weight.

5.1. Heuristic G1-MW

Input: ISDU-graph $G(V, E)$.
Output: A clique $C^{(1)} \in V$.

Step 1: Set $C^{(1)} = \emptyset$. Consider all the intervals as unlabelled.

Step 2: (General iteration) In the graph $G(V, E)$, choose an unlabelled interval that is connected to all the labelled intervals and has the maximum weight. If there is no such interval, then go to Step 3. Otherwise, label this interval, include it in $C^{(1)}$, and remove all the other intervals of the same job from graph $G(V, E)$ (together with the associated edges). Retain the same notation $G(V, E)$ for the new graph. Repeat Step 2.

Step 3: Output $C^{(1)}$.

Heuristic G1-MW can be implemented to run in $O(n^3)$ time by operating with the adjacency matrix.

Heuristic G2-MW makes use of the weighted bipartite matching formulation in Section 2 and Heuristic G1-MW.

5.2. Heuristic G2-MW

Input: ISDU-graph $G(V, E)$.
Output: A clique $C^{(2)} \in V$.

Step 1: Consider all the intervals as unlabelled. Solve problem ISDU under the restriction that at most one job is assigned to each machine (see Section 2). Let $C^{(2)}$ be the corresponding solution. Label intervals in $C^{(2)}$ and remove all other intervals of the jobs in $C^{(2)}$ from graph $G(V, E)$ (together with the associated edges). Retain the same notation $G(V, E)$ for the new graph.

Step 2: In the graph $G(V, E)$, choose an unlabelled interval that is connected to all the labelled intervals and has the maximum weight. If there is no such interval, then go to Step 3. Otherwise, label this interval, include it in $C^{(2)}$, and remove all the other intervals of the same job from graph $G(V, E)$ (together with the associated edges). Retain the same notation $G(V, E)$ for the new graph. Repeat Step 2.

Step 3: Output $C^{(2)}$.

Heuristic G2-MW requires as much operations as heuristic G1-MW plus operations required to solve the weighted bipartite matching problem, which gives $O(n^3)$ in total.

Heuristics G1-MTW and G2-MTW differ from G1-MW and G2-MW, respectively, in that, instead of an interval with the maximum weight, an interval with the maximum total weight of this interval and all the unlabelled intervals adjacent to it is selected. Their run time estimations remain $O(n^3)$.

Heuristic G3 is a simplification of our enumeration Algorithm EMC in that only one maximum weight clique in each layer of the ISDU-graph is considered.

5.3. Heuristic G3

Input: Subgraphs $G(V_l, E_l)$, $l = 1, \ldots, m$, of an ISDU-graph $G(V, E)$.
Output: A clique $C^{(3)} \subseteq V$.

Step 1: Construct a maximum weight clique C_l in each graph $G(V_l, E_l)$, $l = 1, \ldots, m$. Set $A = C_1 \cup \cdots \cup C_m$.

Step 2: For $j = 1, \ldots, n$, find in the set A the set of vertices with the same job index j: $J_j(A) = \{I_{jl} \mid I_{jl} \in C_l,\ \exists\, r : r \neq l,\ I_{jr} \in C_r,\ l,\ r = 1, \ldots, m\}$ and the corresponding set of machine indices: $M_j(A) = \{l \mid I_{jl} \in J_j(A)\}$. Set $J_A = \cup_{j=1}^{n} J_j(A) J_A$.
For each $j = 1, \ldots, n$, determine machine index l_j such that $w_{jl_j} = \max\{w_{jl} \mid l \in M_j(A)\}$. If the above maximum is taken over an empty set, then $I_{jl_j} = \varnothing$.

Step 3: Consider clique $C^0 = \{A \setminus J_A\} \cup \{I_{1l_1}, \ldots, I_{nl_n}\}$. Determine a set of jobs J^0 such that $j \in J^0$ if $I_{jl} \in C^0$, $l = 1, \ldots, m$. For every $j \in J^0$, determine a set of machines M_j^0 such that $l \in M_j^0$ if interval I_{jl} is connected to all the intervals in C^0. For each $j \in J^0$, determine machine index l_j^0 such that $w_{jl_j^0} = \max\{w_{jl} \mid l \in M_j^0\}$. If the above maximum is taken over an empty set, then $I_{jl_j^0} = \varnothing$.

Step 4: Output clique $C^{(3)} = C^0 \cup \{I_{jl_j^0} \mid j \in J^0\}$.

Since a maximum weight clique in a co-interval graph $G(V_l, E_l)$ can be found in $O(|V_l| + |E_l|)$ time, see Frank (1976) and Farber (1982), heuristic G3 can be implemented to run in $O(mn)$ time.

It is easy to see that the weight of the clique $C^{(3)}$ is larger than or equal to the weight of any of the maximum weight cliques $C_1, \ldots, C_m$, whose total weight is larger than or equal to the optimal solution value $w(C^*)$. Therefore, $w(C^{(3)})/w(C^*) \geqslant 1/m$.

6. Computer experiments

We computationally evaluated the performance of heuristics G1-MW, G2-MW, G1-MTW, G2-MTW and G3 and compared them against two existing heuristics: the 2PA algorithm of Berman and DasGupta (2000) and the Greedy$_\alpha$ algorithm of Erlebach and Spieksma (2003) on randomly generated instances. The testing environment is a 32-bit Windows 7 with 2.67 GHz Core 2 Duo CPU and 3.25 GB usable physical memory.

We generated the instances of the problem ISDU in the following way. Denote by E_j the set of *eligible* machines, each of which can process job j.

N_l: For each job j, the cardinality of the set of eligible machines, $|E_j|$, was first uniformly sampled in the range $[1, m]$, and then $|E_j|$ distinct machine indices were uniformly sampled in the same range $[1, m]$. Sets N_l, $l = 1, \ldots, m$, were then formed on the basis of the sets E_j, $j = 1, \ldots, n$

w_{jl}: The normal value was uniformly sampled in the range $[1, 10]$

$|I_{jl}|$: The normal value was uniformly sampled in the range $[1, 15]$

s_{jl}: The normal value was uniformly sampled in the range $[1, n^*\max\{|I_{jl}|\}/\beta]$, where $\max\{|I_{jl}|\}$ is the maximum length of the intervals and β is the compression ratio to control the overlapping of the intervals. Increasing the ratio generates more overlapped intervals. In the experiments, we varied the compression ratio from 1.5 to 2.0 (in increments of 0.1)

$$d_{jl} : d_{jl} = s_{jl} + |I_{jl}|.$$

Optimal solutions were found by our enumeration Algorithm EMC. We coded the algorithms in C language and used the C implementation of the Hungarian method by Brian Gerkey

Table 1 Quality of heuristic solutions

Heuristic	*Percentage of optimal solutions found (%)*	*Relative percentage error*		*Running time (s)*	
		Average (%)	*Maximum (%)*	*Average*	*Maximum*
G1-MW	28.9	3.2	26.2	0.0001	0.030
G2-MW	26.3	3.3	26.2	0.0001	0.016
G1-MTW	12.8	5.6	22.5	0.0001	0.011
G2-MTW	14.4	4.8	26.0	0.0001	0.015
G3	26.7	3.2	16.8	0.0002	0.016
All together	46.1	1.5	14.5	0.0002	0.030

(http://robotics.stanford.edu/~gerkey/tools/hungarian.html) to find the maximum weighted bipartite matching in heuristics G2-MW and G2-MTW.

To evaluate the quality of the solutions obtained by heuristics G1-MW, G2-MW, G1-MTW, G2-MTW and G3, we randomly generated problems with 2-4 machines and 15-25 jobs. For each problem size, 10 instances were generated. Algorithm EMC was used to find the optimal solution value w^*, and heuristics G1-MW, G2-MW, G1-MTW, G2-MTW and G3 were used to find approximate solution values $w^{(1)}, w^{(2)}, \dots, w^{(5)}$, respectively. The relative percentage errors, $[(w^* - w^{(h)})/w^*] \times 100\%$, $h = 1, 2, \dots, 5$, were calculated. The average and maximum relative percentage error, the percentage of optimal solutions found, and the average and maximum running time for each heuristic are summarized in Table 1.

The performance of heuristics G1-MW, G2-MW, G1-MTW, G2-MTW and G3 is comparable. Heuristics G1-MW, G2-MW and G3 give better results than those by heuristics G1-MTW and G2-MTW. Excluding heuristics G1-MTW and G2-MTW yields an average relative percentage error between 3.2 and 3.3%, and produces more than 26% of the optimal solutions. Apply all heuristics to find the solution yield an average relative percentage error of 1.5% and produce more than 46% of the optimal solutions.

To compare heuristics G1-MW, G2-MW, G1-MTW, G2-MTW and G3 against the 2PA and Greedy$_\alpha$ algorithms, medium size instances were randomly generated, which include 2-5 machines and 25-50 (in increments of 5) jobs. Same as before, 10 instances were generated for each problem size. Algorithm Greedy$_\alpha$ was repeatedly run with the parameter α varied from 0 to 1 in increments of 0.1. The maximum and average relative percentage decrease over the best value found by all the heuristics, the percentage of the best solutions found, and the average and maximum running time for each heuristic are summarized in Table 2.

The results show that heuristics G1-MW, G2-MW, G1-MTW, G2-MTW and G3 significantly outperform the two existing heuristics. The average relative percentage decrease over the best solution produced by heuristics G1-MW, G2-MW and G3 is less than 1.4%, while those of Algorithms 2PA and Greedy$_\alpha$ are both 19.9%. More than 99% of the best solutions are found by heuristics G1-MW, G2-MW, G1-MTW, G2-MTW and G3.

Table 2 Comparison of heuristics G1-MW, G2-MW, G1-MTW, G2-MTW and G3 with 2PA and Greedy in medium size problems

Heuristic	*Percentage of best solutions found (%)*	*Relative percentage decrease over the best value*		*Running time (s)*	
		Average (%)	*Maximum (%)*	*Average*	*Maximum*
G1-MW	61.0	1.0	12.1	0.0005	0.002
G2-MW	57.2	1.1	12.1	0.0010	0.003
G1-MTW	13.7	4.5	20.0	0.0007	0.002
G2-MTW	13.6	3.7	15.9	0.0006	0.002
G3	39.0	1.4	12.6	0.0408	7.619
2PA	0.3	19.9	48.4	0.0001	0.004
Greedy$_\alpha$	0.1	19.9	41.0	0.0005	0.007

Table 3 Comparison of heuristics G1-MW, G2-MW, G1-MTW and G2-MTW with 2PA and Greedy in large size problems

Heuristic	*Percentage of best solutions found (%)*	*Relative percentage decrease over the best value*		*Running time (s)*	
		Average (%)	*Maximum (%)*	*Average*	*Maximum*
G1-MW	93.9	0.0	0.2	0.9	1.1
G2-MW	92.8	0.0	0.3	0.9	1.1
G1-MTW	0.0	6.6	9.3	1.4	1.6
G2-MTW	0.0	6.4	9.4	1.4	1.6
2PA	0.0	25.1	29.1	0.0	0.0
Greedy$_\alpha$	0.0	27.9	32.7	0.1	0.1

The maximum relative percentage errors of Algorithms 2PA and Greedy$_\alpha$ are quite stable within the range 41–48.4%. Although heuristics G1-MW, G2-MW, G1-MTW, G2-MTW and G3 have larger variations in the maximum relative percentage error, their maximum relative percentage errors do not exceed 20%.

To further evaluate the quality of heuristics G1-MW, G2-MW, G1-MTW and G2-MTW, 30 large size instances of five machines and 500 jobs were randomly generated for each compression ratio ranging from 1.5 to 2.0. The results are summarized in Table 3. Similar to the results obtained for the medium size problems, heuristics G1-MW and G2-MW outperform the others with all the best solutions found by either of them.

7. Conclusions

We have studied problem ISDU, which is a generalization of the interval scheduling problem ISDI to the case of unrelated parallel machines. We reduced problem ISDU to finding a maximum weight clique in a graph. The existing methods for solving the latter problem can be used to solve the former problem. We suggested a new exact solution method based on an enumeration of maximal cliques in the co-interval graphs, associated with the machines, and five polynomial time heuristic algorithms. We conducted computer experiments on randomly generated instances to assess the performance of our and two other existing heuristics. The computational results show that our heuristics outperform the existing heuristics. The developed software has been used to solve practical instances of the accommodation rental business problem described in the introduction. Our exact method and the heuristics can be an alternative to the commercial software that can also be used to solve problem ISDU.

Further research can concentrate on identifying well-solvable special cases of problem ISDU that are interesting from a practical point of view and on the development of approximation methods with guaranteed worst-case performance. The online or semi-online versions of the problem are of interest as well because they are relevant to real-life situations where job parameters become known only upon job arrival.

An interesting problem related to problem ISDU is the problem of minimizing the (weighted) number of the unrelated parallel machines such that all the jobs of a given set are processed within the specified time intervals on these machines. This problem can be used to design an optimal loading/unloading system with periodically arriving transport devices. Investigation of this problem is another worthy topic for future research in the area of interval scheduling.

Acknowledgements—This research was supported in part by The Hong Kong Polytechnic University under grant number G-YL36. Research of M. Y. Kovalyov was also conducted within the project number Ф10ФП − 001 of the Belarusian Republican Foundation for Fundamental Research.

References

Arkin EM and Silverberg EL (1987). Scheduling jobs with fixed start and end times. *Discrete Applied Mathematics* **18**(1): 1–8.

Berman P and DasGupta B (2000). Multi-phase algorithms for throughput maximization for real-time scheduling. *Journal of Combinatorial Optimization* **4**(3): 307–323.

Bouzina KI and Emmons H (1996). Interval scheduling on identical machines. *Journal of Global Optimization* **9**(3–4): 379–393.

Carlisle MC and Lloyd EL (1995). On the k-coloring of intervals. *Discrete Applied Mathematics* **59**(3): 225–235.

Chudnovsky M, Robertson N, Seymour P and Thomas R (2006). The strong perfect graph theorem. *Annals of Mathematics* **164**(1): 51–229.

Demaine ED, Ghodsi M, Hajiaghayi M, Sayedi-Roshkhar AS and Zadimoghaddam M (2013). Scheduling to minimize gaps and power consumption. *Journal of Scheduling* **16**(2): 151–160.

Dishan Q, Chuan H, Jin L and Manhao M (2013). A dynamic scheduling method of earth-observing satellites by employing rolling horizon strategy. *The Scientific World Journal*. http://dx.doi.org/10.1155/2013/304047.

Eliiyi DT and Azizoglu M (2011). Heuristics for operational fixed job scheduling problems with working and spread time constraints. *International Journal of Production Economics* **132**(1): 107–121.

Epstein L and Levin A (2010). Improved randomized results for the interval selection problem. *Theoretical Computer Science* **411**(34–36): 3129–3135.

Erlebach T and Spieksma FCR (2003). Interval selection: Applications, algorithms, and lower bounds. *Journal of Algorithms* **46**(1): 27–53.

Farber M (1982). *Applications of linear programming duality to problems involving independence and domination*. PhD Thesis, Rutgers University.

Faigle U and Nawijn WM (1995). Note on scheduling intervals on-line. *Discrete Applied Mathematics* **58**(1): 13–17.

Frank A (1976). Some polynomial algorithms for certain graphs and hypergraphs. In: *Proceedings of the 5th British Combinatorial Conference (Aberdeen 1975), Congr. Numer. XV*, Utilitas Mathematica Publishing House: Winnipeg, pp 211–226.

Hopcroft JE and Karp RM (1973). An $n^{(5/2)}$ algorithm for maximum matchings in bipartite graphs. *SIAM Journal on Computing* **2**(4): 225–231.

Kolen AWJ, Lenstra JK, Papadimitriou CH and Spieksma FCR (2007). Interval scheduling: A survey. *Naval Research Logistics* **54**(5): 530–543.

Kovalyov MY, Ng CT and Cheng TCE (2007). Fixed interval scheduling: Models, applications, computational complexity and algorithms. *European Journal of Operational Research* **178**(2): 331–342.

Krumke SO, Thielen C and Westphal S (2011). Interval scheduling on related machines. *Computers & Operations Research* **38**(12): 1836–1844.

Lawler E.L. (1976). *Combinatorial Optimization: Networks and Matroids*. Holt, Rinehart andWinston: New York.

Lee S, Turner J, Daskin MS, Homem-De-Mello T and Smilowitz K (2012). Improving fleet utilization for carriers by interval scheduling. *European Journal of Operational Research* **218**(1): 261–269.

Li Y, Wang B and Caudillo-Fuentes LA (2013). Modeling a hotel room assignment problem. *Journal of Revenue and Pricing Management* **12**: 120–127.

Nejad HTN, Sugimura N and Iwamura K (2011). Agent-based dynamic integrated process planning and scheduling in flexible manufacturing systems. *International Journal of Production Research* **49**(5): 1373–1389.

Renna P (2011). Multi-agent based scheduling in manufacturing cells in a dynamic environment. *International Journal of Production Research* **49**(5): 1285–1301.

Rosgen B and Stewart L (2007). Complexity results on graphs with few cliques. *Discrete Mathematics and Theoretical Computer Science* **9**(1): 127–136.

Rossi A, Singh A and Sevaux M (2010). A metaheuristic for the fixed job scheduling problem under spread time constraints. *Computers & Operations Research* **37**(6): 1045–1054.

Spieksma FCR (1999). On the approximability of an interval scheduling problem. *Journal of Scheduling* **2**(5): 215–227.

Wauters T, Verbeeck K, Berghe GV and De Causmaecker P (2011). Learning agents for the multi-mode project scheduling problem. *Journal of the Operational Research Society* **62**(2): 281–290.

Zheng F, Cheng Y, Liu M and Xu Y (2013). Online interval scheduling on a single machine with finite lookahead. *Computers & Operations Research* **40**(1): 180–191.

Received December 2012;
accepted August 2013 after one revision