JOURNAL OF SCHEDULING
J. Sched. **2**, 215–227 (1999)

ON THE APPROXIMABILITY OF AN INTERVAL SCHEDULING PROBLEM[†]

FRITS C. R. SPIEKSMA*

Department of Mathematics, Maastricht University, P.O. Box 616, NL-6200 MD Maastricht, The Netherlands

SUMMARY

In this paper we consider a general interval scheduling problem. The problem is a natural generalization of finding a maximum independent set in an interval graph. We show that, unless $\mathscr{P} = \mathscr{NP}$, this maximization problem cannot be approximated in polynomial time within arbitrarily good precision. On the other hand, we present a simple greedy algorithm that delivers a solution with a value of at least $\frac{1}{2}$ times the value of an optimal solution. Finally, we investigate the quality of an LP-relaxation of a formulation for the problem, by establishing an upper bound on the ratio between the value of the LP-relaxation and the value of an optimal solution. Copyright © 1999 John Wiley & Sons, Ltd.

KEY WORDS: interval scheduling; approximability; independent set

1. INTRODUCTION

Consider the following optimization problem. Given are n k-tuples of intervals on the real line, that is for each interval l a starting time s_l and a finishing time f_l $(> s_l)$ is known, $l = 1, \ldots, kn$. We assume that all starting and finishing times are integers. An interval is said to be *active* at time t iff $t \in [s_l, f_l)$. Two intervals *intersect* iff there is a time t during which both intervals are active. The problem is to select as many intervals as possible such that (i) no two selected intervals intersect, and (ii) at most one interval is selected from each k-tuple. A k-tuple of intervals is sometimes referred to as a *job*. We refer to this problem as the Job Interval Selection Problem with k intervals per job or JISPk for short. (Observe that an instance where the number of intervals per job is not the same for all jobs is easily transformed to an instance of JISPk for some k by duplicating intervals).

An alternative way of looking at JISPk is by adopting a graph-theoretical point of view. Indeed, let us construct a graph that has a node for each interval and in which two nodes are connected if the corresponding intervals belong to the same job (the *job* edges) or if the corresponding intervals intersect (the *intersection* edges). (Notice that an edge in this graph can be a job edge as well as an interval edge; this reflects the case when two intervals of a same job intersect). JISPk is now equivalent to finding a maximum independent set in this graph.

* Correspondence to: Frits C. R. Spieksma, Department of Mathematics, Maastricht University, P.O. Box 616, NL-6200 MD Maastricht, The Netherlands. E-mail: spieksma@math.unimaas.nl
[†] A preliminary version of this paper appeared in the Proceedings of the APPROX'98 Conference, Lecture Notes in Computer Science, vol. 1444, edited by K. Jansen and J. Rolim, 1998, 169–180.

CCC 1094-6136/99/050215–13$17.50

Obviously, the graph induced by the job edges consists of n disjoint cliques of size k, and the graph induced by the intersection edges is an interval graph. Thus, the graph constructed is the edge union of an interval graph and a graph consisting of n disjoint cliques of size k. Notice that in case $k = 1$ the problem reduces to finding a maximum independent set in an interval graph (which is solvable in polynomial time, see for instance [1]).

JISPk belongs to the field of interval scheduling problems. These problems arise in a variety of settings. Here, we simply refer to Carter and Tovey [2], Fischetti *et al.* [3], Jansen [4], Kroon *et al.* [5, 6] and the references contained therein for examples of applications related to interval scheduling. JISPk is considered in Nakajima and Hakimi [7], who mention applications in real-time environments, and in [8]. Our original interest in JISPk stems from an application in printed circuit board (PCB) manufacturing. Let us proceed to describe this application in more detail.

Numerically controlled machines are indispensable in the automated assembly of PCBs. Typically, a line of placement machines is used to assemble the boards. Each machine in the line is equipped with a so-called feeder rack which consists of slots. These slots are used to hold feeders that deliver the components to be placed on prespecified locations on the board. Notice that a feeder delivers components of a single type. A slot can be used for at most one feeder, and a feeder uses a small number of consecutive slots, usually 1, 2 or 3 slots. To maximize throughput, it is considered advantageous to place feeders in the vicinity of the locations on the board (or set of board types) which it has to serve. In this way, travel time for the arm that carries components from the feeder to the board is minimized. Consider now the feeder rack of a machine in the line. For each pair consisting of a feeder and a position in the rack (i.e. a set of consecutive slots needed to store the feeder), it is specified whether or not it is possible to place that feeder there. In fact, specifying possible positions is not based on distance considerations only: for instance, feeders corresponding to expensive components are required to be placed at the end of a rack so as not to lose these components when something goes wrong in assembling the board. Also, feeders that are popular are sometimes required to be placed at the centre of the rack (see e.g. [9] and the references therein for a more elaborate discussion of the feeder rack assignment problem). Summarizing, let us assume that for each feeder a number of possible positions in each feeder rack is given. Consider now the problem to maximize the number of different feeders to be placed in the line (which is a reasonable objective function when trying to accommodate a large family of different board types). When one concatenates the feeder racks of the machines in the line, and view it as the time axis, and when one views a feeder as a job (and the number of possible positions in all feeder racks as the number of intervals of that job), with the number of slots needed for that feeder equalling the length of each corresponding interval, an instance of JISPk arises.

Improving results of Nakajima and Hakimi [7], Keil [8] proves that the problem of determining whether it is possible to select n intervals is $\mathscr{NP}$-complete for JISP3. This result remains true when each interval has length 2 [10]. Keil [8] also shows that this question is solvable in polynomial time for JISP2. On the other hand, Kolen [11] proves that given an integer K, the question whether one can select at least K intervals is already $\mathscr{NP}$-complete for JISP2.

Our focus in this paper is on the following question: when restricting oneself to polynomial-time algorithms, how good (in terms of quality of the solution) can one solve instances of JISPk, $k \geqslant 2$, in the worst case? Obviously, it follows from Keil [8] that, unless $\mathscr{P} = \mathscr{NP}$, no polynomial-time algorithm is able to solve JISPk exactly. Even more, we establish in Section 3 that, unless $\mathscr{P} = \mathscr{NP}$, no PTAS (see Section 2) exists for JISPk, for all $k \geqslant 2$. On the other hand, we present in Section 4 a polynomial time approximation algorithm that delivers a solution with a value of at least $\frac{1}{2}$ times the value of an optimal solution. In Section 5 we formulate JISPk as an integer programming model and establish bounds on the value of the LP-relaxation in terms of the value of an optimal solution. Finally, we discuss some open problems in Section 6. For an overview of non-approximability results for 'classical' scheduling problems, we refer to Hoogeveen *et al.* [12].

2. PRELIMINARIES

An introduction to the issue of approximation and complexity can be found in [13–15]. Here, we shortly list and describe some of the concepts we need.

1. *A polynomial-time ρ-approximation algorithm* for a maximization problem P is a polynomial time algorithm that, for all instances, outputs a solution with a value that is at least equal to ρ times the value of an optimal solution of P.
2. *A polynomial-time approximation scheme* (PTAS) is a family of polynomial time $(1-\varepsilon)$-approximation algorithms for all $\varepsilon > 0$.
3. An L-reduction. Given two maximization problems A and B, an L-reduction from A to B is a pair of functions R and S such that:

   (a) R and S are computable in polynomial time,
   (b) for any instance I of A with optimum value OPT(I), $R(I)$ is an instance of B with optimum value OPT($R(I)$), such that
   $$\mathrm{OPT}(R(I)) \leqslant \alpha \cdot \mathrm{OPT}(I) \tag{1}$$
   for some positive constant α;
   (c) for any feasible solution s of $R(I)$, $S(s)$ is a feasible solution of I such that
   $$\mathrm{OPT}(I) - c(S(s)) \leqslant \beta\,(\mathrm{OPT}(R(I)) - c(s)) \tag{2}$$
   for some positive constant β, where $c(S(s))$ and $c(s)$ denote the values of solutions $S(s)$ and s, respectively.

   An L-reduction is an *approximation-preserving* reduction, that is, if problem B can be approximated within $1-\varepsilon$ then problem A can be approximated within $1-\alpha\beta\varepsilon$ (assuming that there is an L-reduction from A and B).
4. The class MAX SNP is a class that contains optimization problems that are approximable in polynomial time within a constant factor.
5. The problem *Maximum Bounded 3-Satisfiability* (MAX 3-SAT-3):
   Input: A set of Boolean variables $X = \{x_1, x_2, \ldots, x_n\}$ and a set $C = \{C_1, C_2, \ldots, C_r\}$ of clauses over X. Each clause C_j $(j = 1, \ldots, r)$ consists of at most three literals and each variable x_i $(i = 1, \ldots, n)$ occurs at most three times in C (either as literal x_i or as literal $\bar{x}_i$).
   Goal: Find a truth assignment for the variables such that the number of satisified clauses in C is maximum.
   Measure: The number of satisfied clauses in C.

The following result is proved in [13] (see also [15]):

Lemma 2.1. MAX 3-SAT-3 is MAX SNP-hard.

Arora *et al.* [16] proved the following result:

Lemma 2.2. If there exists a PTAS for some MAX SNP-hard problem, then $\mathscr{P} = \mathscr{NP}$.

We now have sketched the tools that enable us to prove that JISPk has no PTAS (unless $\mathscr{P} = \mathscr{NP}$): this can be done by exhibiting an L-reduction from MAX 3-SAT-3 and using Lemmas 2.1 and 2.2.

3. A NON-APPROXIMABILITY RESULT

Theorem 3.1. JISPk does not have a PTAS unless $\mathscr{P} = \mathscr{NP}$ for each fixed $k \geqslant 2$.

Proof: We prove the theorem by presenting an L-reduction from MAX 3-SAT-3 to JISP2. The result then follows from Lemmas 2.1 and 2.2. Recall that $C = \{C_1, C_2, \ldots, C_r\}$ is a set consisting of r disjunctive clauses, each containing at most 3 literals. Let $x_1, x_2, \ldots, x_n$ denote the variables in the r clauses and, for each $i = 1, \ldots, n$, let $m(i)$ denote the number of occurrences of variable x_i (either as literal x_i or as literal $\bar{x}_i$). Arbitrarily index the occurrences of variable x_i as occurrence $1, 2, \ldots, m(i)$. Notice that without loss of generality we can assume that each variable occurs at least twice in C, thus we have $2 \leqslant m(i) \leqslant 3$ for all i and that $\sum_i m(i) \leqslant 3r$.

We now construct an instance of JISP2, that is a graph $G = (V, E)$ which is the edge union of an interval graph and matching. Let I denote an instance of MAX 3-SAT-3 and $R(I)$ the corresponding instance of JISP2 with corresponding optimal values OPT(I) and OPT($R(I)$). For each variable x_i in I, $i = 1, \ldots, n$, we have a subgraph $H1_i = (V1_i, E1_i)$ in $R(I)$, where $V1_i = \{T_{i1}, F_{i1}, T_{i2}, F_{i2}, \ldots, T_{i,m(i)}, F_{i,m(i)}\}$ and $E1_i = \{\{T_{i1}, F_{i1}\}, \{F_{i1}, T_{i2}\}, \{T_{i2}, F_{i2}\}, \ldots, \{T_{i,m(i)}, F_{i,m(i)}\}, \{F_{i,m(i)}, T_{i1}\}\}$. So for each variable x_i in I we have a cycle consisting of $2m(i)$ nodes in $R(I)$ (see Figure 1).

When no ambiguity is likely to arise, we refer to the nodes T_{ij} (F_{ij}), $j = 1, \ldots, m(i)$, in subgraph $H1_i$ as T-nodes (F-nodes).

For each clause C_j in I, $j = 1, \ldots, r$, we have a subgraph $H2_j = (V2_j, E2_j)$ in $R(I)$ which depends on the cardinality of C_j as depicted in Figure 2.

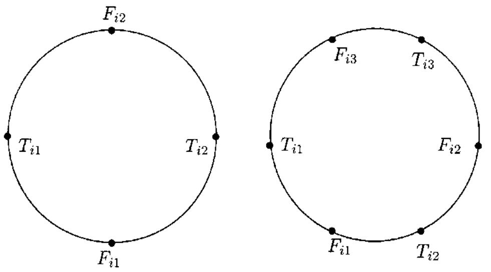

Figure 1. The subgraph $H1_i$ when $m(i) = 2$ (left) and when $m(i) = 3$ (right)

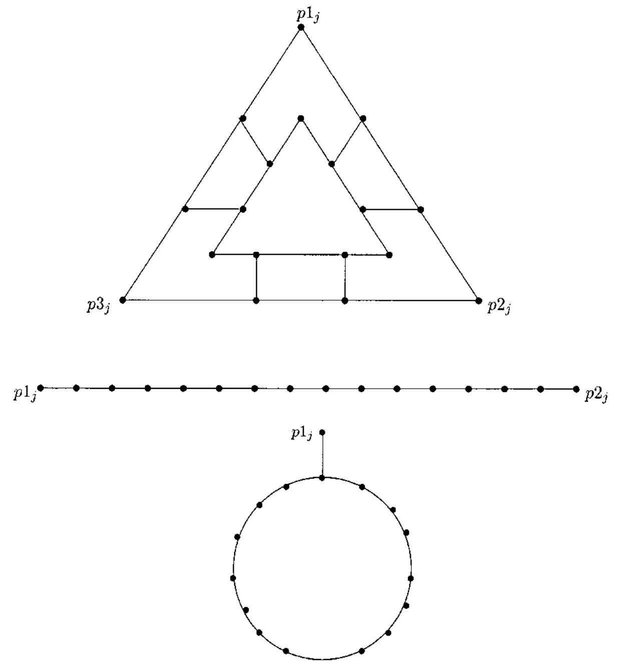

Figure 2. The subgraph $H2_j$ when $|C_j| = 3$ (upper figure), when $|C_j| = 2$ (middle figure) and when $|C_j| = 1$ (lower figure)

Again, when no ambiguity is likely to arise, we refer to the nodes $p1_j$, $p2_j$ and $p3_j$ in a subgraph $H2_j$ as p-nodes. A crucial property of each of the graphs in Figure 2 is the following: the size of a maximum independent set in any graph $H2_j$ is bounded by 8; moreover, if one is not allowed to use p-nodes in an independent set, no more than 7 nodes from a graph $H2_j$ can be in an independent set, $j = 1, \ldots, r$.

To connect the subgraphs introduced so far in $R(I)$, consider some clause C_j, and consider the first variable occurring in this clause C_j, say x_i. Let this be the qth occurrence of this variable x_i in C, $q \in \{1, 2, \ldots, m(i)\}$. If the variable x_i occurs as literal x_i add the edge $\{p1_j, F_{iq}\}$ to E. If the variable x_i occurs as literal $\bar{x}_i$ add the edge $\{p1_j, T_{iq}\}$ to E. In case $|C_j| \geqslant 2$, consider now the second (third) variable occurring in C_j, say x_l, and let this be the qth occurrence of this variable x_l in C, $q \in \{1, 2, \ldots, m(i)\}$. If the variable x_l occurs as literal $\bar{x}_l$ add the edge $\{p2_j, F_{lq}\}$ ($\{p3_j, F_{lq}\}$) to E. If the variable x_l occurs as literal $\bar{x}_l$ add the edge $\{p2_j, T_{lq}\}$ ($\{p3_j, T_{lq}\}$) to E. This is done for all clauses C_j, $j = 1, \ldots, r$.

Now the graph $G = (V, E)$ is completely specified.

Let us argue that the resulting graph G is the edge union of an interval graph and a perfect matching, which implies that we have constructed an instance of JISP2.

Observe that no node in G has degree exceeding 3. We now exhibit a perfect matching M in G; these edges are the job edges (see Section 1). M consists of two parts: edges in $\bigcup_i H1_i$ and edges in $\bigcup_j H2_j$. For the first part we take $\bigcup_i \{\{T_{ij}, F_{ij}\} \mid j = 1, \ldots, m(i)\}$. For the second part we take the following edges: if $|C_j| \leqslant 2$, it is trivial to exhibit a perfect matching in $H2_j$ (see Figure 2), else consider the dotted edges depicted in Figure 3.

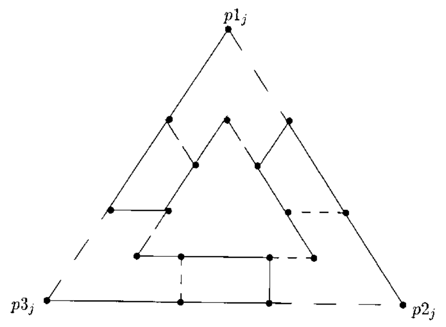

Figure 3. The subgraph H2j

Obviously, M is indeed a matching. Also, one easily verifies that the remaining edges in G (the intersection edges) from a set of disjoint paths, which corresponds to an interval graph.

In order to show that this reduction fulfills inequalities (1) and (2), consider the following. Observe that $v \equiv \mathrm{OPT}(I) \geqslant \frac{1}{2} r$. (Indeed, by considering the assignment: all variables true, and all variables false, it follows that each clause is true in at least one of both assignments.) We have

$$\mathrm{OPT}(R(I)) \leqslant 3r + 8r = 11r \leqslant 22v = 22\ \mathrm{OPT}(I)$$

which proves (1). The first inequality follows from the fact that

(i) at most $m(i)$ nodes can be selected from each $H1_i$, $i = 1, \ldots, n$ (see Figure 1), and $\sum_i m(i) \leqslant 3r$, and
(ii) at most 8 nodes can be selected from each subgraph $H2_j$, $j = 1, \ldots, r$ (see Figure 2).

To establish (2), we do the following. Consider an arbitrary solution to $R(I)$, that is any independent set s in G with size $c(s)$. We will map this solution s using intermediate solutions s' and s'' to a solution of MAX 3-SAT-3, called $S(s)$. To do this we need the following definition. An independent set s in G is called *consistent* iff for each $i = 1, \ldots, n$, $m(i)$ nodes from $V1_i$ are in s.

Now we state a procedure which takes as input an independent set s. The output of the procedure is a consistent independent set called s' with the property that $c(s') \geqslant c(s)$.

Procedure
Consider s. For $i = 1, \ldots, n$, consider $V1_i$. There are two possibilities.

1. $m(i)$ nodes from $V1_i$ are in s. Then either all T-nodes or all F-nodes from $V1_i$ are in s and we leave s unaltered.
2. Less than $m(i)$ nodes from $V1_i$ are in s. Let cT (cF) be the number of T-nodes (F-nodes) in $V1_i$ that are connected to p-nodes that are in s. (Notice that $cT + cF \leqslant 3$).

   Distinguish two subcases:

   (a) If $cT > cF$, it follows that $cF \leqslant 1$. Modify s by selecting all F-nodes from $V1_i$ (and undo the selection of any T-nodes in s), and, if $cF = 1$, undo in s the selection of the p-node connected to an F-node. Notice that this modification does not decrease the number of nodes in the independent set. The case $cT < cF$ is treated similarly.

(b) $cT = cF$. In that case, select from $V1_i$ all T-nodes, and undo the selection of a p-node connected to a T-node. (Notice that there can be at most 1 such node). Again, modifying s in this way does not decrease the number of selected nodes.

End of Procedure

After applying this procedure to any independent set s in G, a consistent solution s' is delivered. Now we describe how to modify s' to get solution s''. Consider in s' those subgraphs $H2_j$ whose corresponding p-nodes cannot all be chosen, due to nodes from $\bigcup_i V1_i$ in s'. Suppose there are $r - l$ of those subgraphs in s'. Then we modify s' such that in l subgraphs $H2_j$ 8 nodes are selected and in $r - l$ subgraphs $H2_j$ 7 nodes (this is always possible, see Figures 1 and 2). This gives us a consistent solution s'' with $c(s'') \geqslant c(s')$. Since s'' is consistent, it is now straightforward to identify the corresponding solution $S(s)$ in MAX 3-SAT-3: simply set variable $x_i, \ldots, n$, true if all T-nodes in subgraph $H1_i$ are selected in s'', else set x_i false. How many clauses in I are satisfied by this truth assignment? Observe that the construction of G implies that if for some consistent independent set s each p-node from some $H2_j$ is connected to a node in $\bigcup_i V1_i$ that is in s, then the corresponding truth assignment renders clause C_j not satisfied, and vice versa. Thus, by the construction of s'', it follows that a subgraph $H2_j$ for which 7 nodes are in s'' corresponds to a not satisfied clause, and otherwise the clause is satisfied, $j = 1, \ldots, r$. This implies that l clauses in I are satisfied by this truth assignment.

Again, let $v = \mathrm{OPT}(I)$, and let $c(S(s)) = l$. The following (in)equalities are true:

1. $c(s) \leqslant c(s'')$ (by construction),
2. $c(s'') = \sum_i m(i) + 8l + 7(r - l) = \sum_i m(i) + 7r + l$ (by construction), and
3. $\mathrm{OPT}(R(I)) \geqslant \sum_i m(i) + 8v + 7(r - v) = \sum_i m(i) + 7r + v$ (consider the truth assignment that is optimum for I; evidently, we can exhibit in $R(I)$) a corresponding independent set of size $\sum_i m(i) + 7r + v$.

Thus

$$\begin{aligned}\mathrm{OPT}(R(I)) - c(s) &\geqslant \mathrm{OPT}(R(I)) - c(s'')\\ &\geqslant \sum_i m(i) + 7r + v - \left(\sum_i m(i) + 7r + l\right)\\ &= v - l = \mathrm{OPT}(I) - c(S(s))\end{aligned}$$

which proves (2). □

There are a number of implications that can be observed from this reduction.

Remark. First of all, the reduction remains valid if there are restrictions on the number of intervals that is active at time t for some t. More specifically, let $\imath_t(I)$ be the number of intervals in I that is active at time t, and define the *maximum intersection* as $\imath(I) = \max_t \imath_t(I)$. Notice that Theorem 3.1 remains true even when $\imath(I) \leqslant 2$ (whereas the problem becomes trivial when $\imath(I) \leqslant 1$).

Remark. Also, the reduction remains valid for short, equal processing times. Indeed, even if $f_l - s_l = 2$ for all intervals l, Theorem 3.1 remains true (whereas the problem again becomes trivial in the case that $f_l - s_l = 1$ for all l).

Remark. Consider the decision version of JISPk. As mentioned in Section 1, Keil [8] proves that the question whether one can select n intervals in a JISP2 instance is solvable in polynomial time. In fact, this result can also be seen in a graph-theoretical context. Graphs for which the size of a maximum matching equals the size of a minimum vertex cover are said to have the *König property*. Since the complement of a minimum vertex cover is a maximum independent set, it follows that for graphs with the *König property* the cardinality of a maximum independent set can be found in polynomial time. Now, the size of a maximum matching for a graph corresponding to a JISP2 instance equals n. So the question Keil [8] answered is equivalent to the question whether the graph corresponding to a JISP2 instance has the König property. This problem can be solved in polynomial time (see [17] and the references contained therein).

Remark. The proof of Theorem 3.1 can be seen as an alternative proof of the fact that maximum independent set in graphs of degree at most 3 is MAX SNP complete as was recently shown in [18, 19]. See also the recent work of Berman and Karpinski [20] for an inapproximability result concerning this problem.

Finally, when formulating Theorem 3.1 in graph theoretic terms we get

Corollary 3.2. Finding a maximum independent set in a graph that is the edge-union of an interval graph with degree at most 2 and a matching is MAX SNP complete.

4. AN APPROXIMATION ALGORITHM

Instances of JISPk have the property that an interval from the job corresponding to the earliest ending interval (say job j) is selected in an optimal solution. Indeed, suppose not, and consider an optimal solution not containing an interval from job j. Then one can interchange the earliest ending interval in this solution with the earliest ending interval of job j. This property motivates the following greedy algorithm. Start 'at the left', and select, repeatedly, the earliest ending interval such that (i) it does not intersect with the last selected interval, and (ii) it does not belong to an earlier selected job. Applied to any instance I of JISPk, this gives at least $\frac{1}{2}$ OPT(I). A more formal description of the algorithm, referred to as GREEDY, is as follows. Let $G(I)$ be the set of intervals selected from I by GREEDY, and let $J(l)$ be the set of intervals belonging to the job corresponding to interval l, $l = 1, \ldots, kn$.

```
GREEDY:
$T := -\infty$;
$G(I) := \emptyset$;
$S :=$ set of all intervals in $I$;
while $\max_{l \in S} s_l \geqslant T$ do
begin
        $l^* := \arg(\min_{l \in S} \{f_l \mid s_l \geqslant T\})$ (break ties arbitrarily);
        $G(I) := G(I) \bigcup \{l^*\}$;
        $S := S \setminus J(l^*)$;
        $T := f_{l^*}$;
end;
```

Obviously, GREEDY is a polynomial time algorithm.

Theorem 4.1. GREEDY is a $\frac{1}{2}$-approximation algorithm for JISPk, $k \geqslant 1$. Moreover, there exist instances of JISPk for which this bound is tight, for all $k \geqslant 2$.

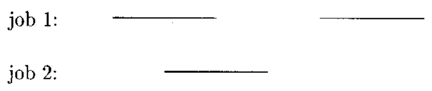

Figure 4. A worst-case instance for GREEDY

Proof. Consider some instance I of JISPk. Applying GREEDY gives us a solution with $|G(I)|$ intervals selected. The idea of the proof is to partition I into two instances I_1 and I_2, and show that for each of those instances it is impossible to select more than $|G(I)|$ intervals. Clearly, then no more than $2|G(I)|$ intervals can be selected, proving the first part of the theorem.

Now, let I_1 consist of the jobs whose intervals are selected by GREEDY, and let I_2 consist of all other jobs. Obviously, $\mathrm{OPT}(I_1) \leqslant |G(I)|$, since I_1 contains no more as $|G(I)|$ jobs. Let the finishing times of all intervals selected by GREEDY be indexed $e_1 < e_2 < \ldots = e_{|G(I)|}$ and let $e_0 = -\infty$. For each interval in I_2 we know that it is active at $e_j - 1$ for some $j = 1, \ldots, |G(I)|$. (Otherwise it would have been selected by GREEDY). In other words, all intervals in I_2 that have a starting time in $[e_{j-1}, e_j)$ have a finishing time after time e_j, $j = 1, \ldots, |G(I)|$. Thus at most one of those can be in a solution of I_2. Since there are only $|G(I)|$ such time intervals $[e_{j-1}, e_j)$, at most $|G(I)|$ intervals can be selected. Summarizing, we have $\mathrm{OPT}(I) \leqslant \mathrm{OPT}(I_1) + \mathrm{OPT}(I_2) \leqslant |G(I)| + |G(I)|$.

To show that this is best possible for GREEDY, consider the instance of JISP2 depicted in Figure 4 (where the interval corresponding to job 2 has multiplicity 2).

It is easy to see that for this instance I, $\mathrm{OPT}(I) = 2$, whereas $|G(I)| = 1$. □

Remark. Notice that, for $k = 1$, GREEDY reduces to a special case of an algorithm described by Carlisle and Lloyd [21] and Faigle and Nawijn [22], and hence always finds an optimal solution.

5. AN IP-FORMULATION FOR JISPk

A popular way of devising approximation algorithms for combinatorial optimization problems is by formulating the combinatorial problem as an integer program, solve the LP-relaxation and next try to round (in some problem-dependent way) the LP-relaxation to a feasible solution while not sacrificing too much. Of course, the gap that can arise between the value of the LP-relaxation and the optimum integral solution can serve as an indication of the potential quality of any approximation algorithm based on this approach. In this section we will show that the value of the LP-relaxation of a (straightforward) integer programming formulation of JISPk can be almost two times the value of an optimal solution. This implies that straightforward algorithms based on this LP-relaxation cannot (substantially) improve on the bound of GREEDY.

Consider now the following integer programming formulation that expresses JISPk as a set packing (SP) problem. Assume w.l.o.g. that job i consists of intervals $k(i-1)+1$, $k(i-1)+2, \ldots, ki$. Let $x_l = 1$ if interval l is selected and 0 otherwise, and let $A(l) = \{j$: interval j is active at $f_l - 1\}$, $l = 1, \ldots, kn$. (Notice that $l \in A(l)$).

$$(SP) \quad \text{Maximize} \quad \sum_{l=1}^{kn} x_l$$

$$\text{subject to} \quad x_{k(i-1)+1} + \ldots + x_{ki} \leqslant 1 \quad \text{for all } i = 1, \ldots, n \tag{3}$$

$$\sum_{j \in A(l)} x_j \leqslant 1 \qquad \text{for all } l = 1, \ldots, kn \qquad (4)$$

$$x_l \in \{0, 1\} \qquad \text{for all } l = 1, \ldots, kn \qquad (5)$$

Constraints (3) express that at most 1 interval per job can be selected, while constraints (4) ensure that no intersection occurs in the set of selected intervals. Constraints (5) are the integrality constraints. Let $v_{\mathrm{LP}}(I)$ denote the value of the LP-relaxation of (SP) with respect to instance I of JISPk.

Theorem 5.1. $v_{\mathrm{LP}}(I) \leqslant 2\ \mathrm{OPT}(I)$ *for all I. Moreover, this bound is asymptotically tight.*

Proof. The idea is as follows. Let us construct a solution which is feasible to the dual of the LP-relaxation of (*SP*). This solution will have a value, say $v_{\mathrm{D}}(I)$, bounded by 2 OPT(I) for all I. Then, by LP-duality we are done: $v_{\mathrm{LP}}(I) \leqslant v_{\mathrm{D}}(I) \leqslant 2\mathrm{OPT}(I)$ for all I.

Associating z-variables to the first set of constraints of (SP) and y-variables to the second set of constraints, we get the following dual of the LP-relaxation of (SP) (let $A^{-1}(l) = \{j\colon$ interval l is active at $f_j - 1\}$, $l = 1, \ldots, kn$):

$$\text{(D)} \quad \text{Minimize} \quad \sum_{l=1}^{kn} y_l + \sum_{i=1}^{n} z_i$$

$$\text{subject to} \quad z_{\lceil l/k \rceil} + \sum_{j \in A^{-1}(l)} y_j \geqslant 1 \quad \text{for all } l = 1, \ldots, kn$$

$$\text{all variables} \geqslant 0.$$

One can think of the z-variables as horizontal lines, such that z_i covers intervals $k(i-1)+1, \ldots, ki$ ($i = 1, \ldots, n$) and of the y-variables as vertical lines such that y_l is at time $f_l - 1$ and covers all intervals in $A(l)$ ($l = 1, \ldots, kn$). The dual problem (D) is now to give the dual variables non-negative weights such that total weight is minimized and every interval receives at least weight 1 from those dual variables by which it is covered.

Consider now the set of all optimal solutions to (SP) with respect to some instance I (so the optimal integral solutions). From this set we are going to identify a special optimal solution which we call EARLYOPT(I). Consider for each optimal solution the finishing times of the intervals in that solution in increasing sequence. EARLYOPT(I) is now that optimal solution in which the sequence of finishing times is the lexicographic smallest sequence. Let SISTERS(I) be the set of intervals whose corresponding jobs have an interval in EARLYOPT(I) (formally SISTERS(I) $= \bigcup_{l \in \mathrm{EARLYOPT}(I)} (J(l) \backslash l)$), and let REST($I$) be the set of all remaining intervals. Thus, we have partitioned the set of intervals in I into three subsets.

We now construct the following dual solution:

(1) $y_l = 1$ for all $l \in$ EARLYOPT(I). Notice that the construction leading to EARLYOPT(I) implies that each interval from REST(I) is covered by some y_l, $l \in$ EARLYOPT(I). (Indeed, suppose not, then there exists an 'earlier' optimal solution than EARLYOPT(I) which is impossible.) Thus, by choosing these weight, each interval from EARLYOPT(I) as well as each interval from REST(I) receives weight 1. Total weight spent: OPT(I).
(2) $z_{\lceil l/k \rceil} = 1$ for all $l \in$ EARLYOPT(I). This implies that each interval from EARLYOPT(I) as well as from SISTERS(I) receives weight 1. Total weight spent: OPT(I).
(3) All other dual variables are 0.

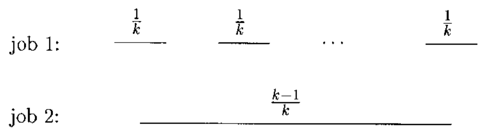

Figure 5. An instance of JISPk

It is easy to verify that this constitutes a feasible dual solution with weight 2 OPT (I). The first part of the theorem then follows. To establish the second part, consider the following instance, depicted in Figure 5 (where the interval corresponding to job 2 has multiplicity k).

It is not hard to verify that the numbers above the intervals in Figure 5 are the optimal LP-values of the corresponding x-variables. Thus, for this instance we have that $v_{\mathrm{LP}}(I) = 2 - 1/k$, whereas OPT($I$) clearly equals 1. □

Remark. Notice that we actually proved a stronger statement than announced in Theorem 5.1. Indeed, let $v_{\mathrm{SC}}(I)$ be the value corresponding to the formulation which arises when the constraints $y, z \in \{0, 1\}$ are added to problem D (then a set covering problem arises). Arguments in the proof of Theorem 5.1 imply that $v_{\mathrm{SC}}(I) \leqslant 2\mathrm{OPT}(I)$ and the instance in Figure 5 shows that this inequality is right for each $k \geqslant 2$.

Although the bound in Theorem 5.1 is asympotically tight, there remains a sizable gap for JISPk instance with small values of k (for $k = 2$, the gap is $\frac{3}{2}$ versus 2). The following theorem closes part of this gap.

Theorem 5.2. $v_{\mathrm{LP}}(I) \leqslant \frac{5}{3}$ OPT (I) for all JISP2 instances I.

Proof: We refine the proof of Theorem 5.1. Construct the following dual solution.

(1) $y_l = \frac{2}{3}$ for all $l \in$ EARLYOPT (I). It follows (see the proof of Theorem 5.1) that each interval from EARLYOPT (I) as well as each interval from REST (I) receives weight $\frac{2}{3}$. Total weight spent: $\frac{2}{3}$ OPT (I).

(2) $z_{\lceil l/2 \rceil} = \frac{1}{3}$ for all $l \in$ EARLYOPT (I). This implies that each interval from EARLYOPT (I) as well as from SISTERS (I) receives weight $\frac{1}{3}$. Total weight spent: $\frac{1}{3}$ OPT (I).

To proceed, we construct from instance I an instance I' by deleting from I all intervals in EARLYOPT (I). Obviously, OPT(I') $\leqslant$ OPT(I).

(3) $y_l = \frac{1}{3}$ for all $l \in$ EARLYOPT (I'). It follows that each interval from EARLYOPT (I') as well as each interval from REST (I') receives weight $\frac{1}{3}$. Total weight spent: at most $\frac{1}{3}$ OPT (I).

Construct now the instance I'' by taking all intervals from SISTERS (I) and SISTERS (I'). Observe that there are no two intervals present in I'' belonging to a same job (this follows from the construction of I' and the fact that $k = 2$). Thus, we are now dealing with finding a maximum independent set in an interval graph. Such an instance is solvable by GREEDY as explained earlier. Set

(4) $y_l = \frac{1}{3}$ for all $l \in G(I'')$. Notice that each interval from I'' is covered by some y_l, $l \in G(I'')$. Notice also that $|G(I'')| \leqslant$ OPT (I), thus total weight spent: at most $\frac{1}{3}$ OPT (I).

All other dual variables get weight 0. If we sum total weight spent in (1)–(4) it follows we have spent not more as $\frac{5}{3}$ OPT. It remains to argue that each interval from the instance has received weight at least 1. Take any interval from I and distinguish 5 cases:

(i) It belongs to EARLYOPT(I). Then it gets $\frac{2}{3}$ from (1) and $\frac{1}{3}$ from (2).
(ii) It belongs to SISTERS(I). Then it gets $\frac{1}{3}$ from (2), $\frac{1}{3}$ from (3) (since each interval from SISTERS(I) is either an EARLYOPT(I') or a REST(I') interval) and it gets $\frac{1}{3}$ from (4).
(iii) It belongs to REST(I) and EARLYOPT(I'). Then it gets $\frac{2}{3}$ from (1) and $\frac{1}{3}$ from (3).
(iv) It belongs to REST(I) and REST(I'). Then it gets $\frac{2}{3}$ from (1) and $\frac{1}{3}$ from (3).
(v) It belongs to REST(I) and SISTERS(I'). Then it gets $\frac{2}{3}$ from (1) and $\frac{1}{3}$ from (4).

This completes the proof. □

6. CONCLUSION AND OPEN PROBLEMS

In this paper we introduced an interval scheduling problem. Formulated in graph-theoretic terms the problem is a natural generalization of finding a maximum independent set in an interval graph. We prove that a PTAS is unlikely to exist for this problem, and we show that a simple greedy algorithm is guaranteed to find a solution with a value at least $\frac{1}{2}$ times the value of an optimal solution. Moreover, we provide motivation that algorithms based on rounding the LP-relaxation of a straightforward integer programming formulation will not give substantially better performance guarantees.

There are a number of questions that remain open. Of course, it would be interesting to have better approximation algorithms, or more generally to shrink the gap between what is and what is not achievable in polynomial time. Also establishing a tight ratio between the value of the LP-relaxation and the optimum value for each k seems worthy of further study. Finally, there may be potential of generalizing these results to other graph classes and to the weighted case.

ACKNOWLEDGEMENTS

I would like to express my gratitude to the referees for their comments.

REFERENCES

1. M. C. Golumbic, *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, San Diego, CA, 1980.
2. M. W. Carter and C. A. Tovey, 'When is the classroom assignment problem hard?', *Oper. Res*., **40,** S28–S39 (1992).
3. M. Fischetti, S. Martello and P. Toth, 'Approximation algorithms for fixed job schedule problems', *Oper. Res*., **40,** S96–S108 (1992).
4. K. Jansen, 'Approximation results for the optimum cost chromatic partition problem', *Proc. DIMACS Workshop on Network Design*, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Vol. 40, 1998, pp. 143–168,
5. L. G. Kroon, A. Sen, H. Deng and A. Roy, 'The optimal cost chromatic partition problem for trees and interval graphs', *Proc. Workshop on Graph Theoretical Concepts in Computer Science*, Lecture Notes in Computer Science, Vol. 1197, 1997, pp. 279–292.
6. L. G. Kroon, M. Salomon and L. N. van Wassenhove, 'Exact and approximation algorithms for the tactical fixed interval scheduling problem', *Oper. Res*., **45,** 624–638 (1997).
7. K. Nakajima and S. L. Hakimi, 'Complexity results for scheduling tasks with discrete starting times', *J. Algorithms*, **3,** 344–361 (1982).
8. J. M. Keil, 'On the complexity of scheduling tasks with discrete starting times', *Oper. Res. Lett*., **12,** 293–295 (1992).
9. Y. Crama, O. E. Flippo, J. J. van de Klundert and F. C. R. Spieksma, 'The assembly of printed circuit boards: a case with multiple machines and multiple board types', *European J. Oper. Res*., **98,** 457–472 (1997).

10. F. C. R. Spieksma and Y. Crama, 'The complexity of scheduling short tasks with few starting times', *Reports in operations research and systems theory M92-06*, Maastricht University.
11. A. W. J. Kolen, Personal communication.
12. J. A. Hoogeveen, P. Schuurman and G. J. Woeginger, 'Non-approximability results for scheduling problems with minsum criteria', *Proc. 6th Conf. on Integer Programming and Combinatorial Optimization*, Lecture Notes in Computer Science, Vol. 1412, Springer, Berlin, 1998, pp. 353–366.
13. C. H. Papadimitriou, *Computational Complexity*, Addision-Wesley, Reading, MA 1994.
14. P. Crescenzi and V. Kann, 'A compendium of $\mathcal{NP}$ optimization problems', http://www.nada.kth.se/~viggo/wwwcompendium/
15. G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti Spaccamela and M. Protasi, *Complexity and Approximation. Combinatorial Optimization Problems and their Approximability Properties*, Springer, Berlin, 1999, to appear.
16. S. Arora, C. Lund, R. Motwani, M. Sudan and M. Szegedy, 'Proof verification and hardness of approximation problems', *Proc. 33rd IEEE Symp. on the Foundations of Computer Science*, 1992, pp. 14–23.
17. M. D. Plummer, 'Matching and vertex packing: how "hard" are they?', *Ann. Discrete Math.*, **55,** 275–312 (1993).
18. P. Alimonti and V. Kann, 'Hardness of approximating problems on cubic graphs', *Proc. 3rd Conf. on Algorithms and Complexity.* Lecture Notes in Computer Science, Vol. 1203, Springer, Berlin, 1997, pp. 288–298.
19. P. Berman and T. Fujito, 'On approximation properties of the independent set problem for degree 3 graphs', *Proc. 4th Workshop on Algorithms and Data Structures*, Lecture Notes in Computer Science, Vol. 955, Springer, Berlin, 1995, pp. 449–460.
20. P. Berman and M. Karpinski, 'On some tighter inapproximability results', *Electronic Colloquium on Computational Complexity*, *Report No. 29*, 1998.
21. M. C. Carlisle and E. L. Lloyd, 'On the *k*-coloring of intervals', *Discrete Appl. Math.*, **59,** 225–235 (1995)
22. U. Faigle and W. M. Nawijn, 'Note on scheduling intervals on-line', *Discrete Appl. Math.*, **58,** 13–17 (1995).