Journal of Algorithms **31,** 196–210 (1999)
Article ID jagm.1998.1001, available online at http://www.idealibrary.com on IDEAL®

A Greedy On-Line Algorithm for the k-Track Assignment Problem

U. Faigle,* W. Kern, and W. M. Nawijn

Department of Applied Mathematics, University of Twente, P.O. Box 217, 7500 AE Enschede, The Netherlands
E-mail: {faigle,kern,nawijn}@math.utwente.nl

Received August 5, 1997; revised August 14, 1998

Given a collection $\mathcal{I}$ of n jobs that are represented by intervals, we seek a maximal feasible assignment of the jobs to k machines such that not more than $c(M)$ intervals overlap pairwise on any machine M and that a job is only assigned to a machine if it fits into one of several prescribed time windows for that machine. We analyze an on-line version of this *NP*-complete problem and exhibit an algorithm that achieves at least half of the (theoretical) optimum. In a more detailed analysis, we bound the performance of our algorithm by an additive term that only depends on the time window structure of the machines (but not on the number of jobs). In the case where each machine M has capacity $c(M) = 1$, our bound implies that our algorithm loses at most $k - 1$ jobs relative to the optimum. We show by an explicit construction that the bound is tight for greedy on-line algorithms.

Key Words: greedy algorithm; k-track assignment; interval order; on-line; scheduling; time window.

1. INTRODUCTION

The *k-track assignment* problem is an interval scheduling problem that has received considerable attention in the literature (see, e.g., Arkin and Silverberg [1], Brucker and Nordmann [2], Carlisle and Lloyd [3], Faigle and Nawijn [4]). It arises when n jobs requiring specific processing times

*Corresponding author.

0196-6774/99 $30.00

are to be assigned *on-line* to k machines that are only available during certain *time windows*, for instance to satellites that are orbiting the earth (other real world applications are described, e.g., in Kolen and Lenstra [6]).

The overall assignment has to be done in such a way that no two jobs on the same machine overlap. The objective is to maximize the number of feasible individual assignments. Brucker and Nordmann [2] reduce the k-coloring problem for circular arc graphs, which is known to be *NP*-complete (Garey et al. [5]), to a special interval scheduling problem with time windows, thus showing that the problem in this generality is *NP*-hard.

The k-track assignment problem is usually studied in an *off-line* version, where all problem data are known in advance. Extending the work of Arkin and Silverberg, Brucker and Nordmann give an $O(n^{k-1})$ dynamic programming algorithm and note that the algorithm might be of practical use when there are not more than $k = 5$ machines.

Faigle and Nawijn [4] observe that a simple greedy-type algorithm not only solves the problem optimally but it can also be used as an (optimal) on-line algorithm when the time windows are identical. (This situation corresponds to the problem of maximizing the union of k chains in an interval order).

In this paper, we analyze the performance of the greedy on-line heuristic in the presence of possibly different time windows. The result turns out to be surprisingly good; our algorithm constructs a feasible schedule that is never more than $k - 1$ jobs off the (theoretical) optimum, independent of the number n of jobs. We actually study the problem in a slightly more general context, where the feasibility of a machine for a job is determined by several time windows per machine and jobs on a single machine may overlap as long as a prescribed machine capacity is not exceeded. The machine capacity then enters the performance bound (still being independent of the number of jobs).

Arkin and Silverberg [1] have generalized the scheduling problem to the model where job-machine feasibility is modeled by an arbitrary bipartite graph. While the dynamic programming approaches for the off-line version go through in this generality, we must leave the complexity analysis of the general on-line model as an open problem.

The paper is organized as follows. After introducing the full general model, we concentrate on the class of assignment problems relative to time windows. We exhibit an explicit on-line scheduling algorithm and bound its performance by an additive term that only depends on the time window structure of the machines. We show by example that our algorithm is optimal; i.e., for any set of problem parameters no on-line algorithm can improve our bound. Taking a different perspective, we finally observe that our algorithm achieves at least one half of the optimum. Also this bound can be shown to be sharp.

2. THE GENERAL INTERVAL SCHEDULING MODEL

We consider an (ordered) list $\mathcal{I} = (I_1, I_2, \ldots, I_n)$ of n "jobs" that are represented by intervals with both endpoints in the set $\mathbb{N}$ of natural numbers. Strictly speaking, the restriction to $\mathbb{N}$ in our model is not necessary. As will be seen below, the algorithmic performance analysis only uses the relative position of the intervals within the set of $\mathcal{I}$ intervals. For ease of exposition, however, we make this technical integrality assumption right from the beginning.

For every $I \in \mathcal{I}$, we denote by $l(I)$ ($r(I)$) the *left* (resp. *right*) endpoint of the interval representing I. The difference $r(I) - l(I) \geq 0$ is the *length* of I and can be understood as the "processing time" job I requires. Without loss of generality, we will always assume that every job I is "nontrivial," i.e., has strictly positive length $r(I) - l(I) \geq 1$.

Thinking of the jobs as arriving at points $t = 1, 2, \ldots, n$ in "time," we assume

$$l(I_1) \leq l(I_2) \leq \cdots \leq l(I_n).$$

Note that we use the term "time" synonymously with the index set of $\mathcal{I}$. In a real-world situation, the "actual arrival time" of a job I could, for example, be the left endpoint $l(I)$ of its processing interval. We prefer our terminology in the performance analysis since it allows us to specify a unique sequential order among jobs with the same "actual arrival time."

The jobs are to be placed on the k "machines" in the set

$$\mathcal{M} = \{M_1, M_2, \ldots, M_k\},$$

where the placement is subject to two conditions:

First, we assume that there exists a binary relation

$$f \subseteq \mathcal{I} \times \mathcal{M},$$

indicating if a machine M is *feasible* for a job I. We use the notation $(I, M) \in f$ or, equivalently, $f(I, M) = 1$ for this feasibility ($(I, M) \notin f$ is denoted by $f(I, M) = 0$).

Second, we assume that a machine $M \in \mathcal{M}$ has a *capacity* $c(M) \in \mathbb{N}$ restricting the number of jobs on which M can work simultaneously.

A (*feasible*) *schedule* is thus a subset $\sigma \subseteq \{I_1, I_2, \ldots, I_n\}$ of jobs I that can be assigned to machines M such that the following feasibility conditions hold:

(S_1) If I is assigned to M, then $f(I, M) = 1$.

(S_2) If the subset $\mathcal{I}' \subseteq \mathcal{I}$ is assigned to the same machine M, then $\mathcal{I}'$ contains no $c(M) + 1$ jobs with pairwise overlapping intervals.

A (feasible) schedule σ^* is *optimal* if

$$|\sigma^*| = \max\{|\sigma| : \sigma \text{ schedule}\}.$$

We also use the notation $\sigma^* = \sigma^*(\mathcal{I}, \mathcal{M})$ to indicate that $|\sigma^*|$ depends on the problem $(\mathcal{I}, \mathcal{M})$. (Strictly speaking, a problem is defined by the triple $(\mathcal{I}, \mathcal{M}, f)$. To simplify the notation, however, we usually suppress the symbol f.)

We propose a greedy algorithm for this scheduling problem that is also *on-line* in the sense that it deals with the jobs in the given order $I_1, I_2, \ldots, I_n$. We do not assume that full information about the problem $(\mathcal{I}, \mathcal{M}, f)$ is available in advance. Only at time t, when job I_t "arrives" to be dealt with, knowledge of the interval $[l(I_t), r(I_t)]$ and the set

$$\mathcal{M}(I_t) := \{M \in \mathcal{M} : f(I_t, M) = 1\}$$

is needed.

At each time t, the "incoming" job I_t is either discarded or (tentatively) placed on some machine M. A tentatively placed job may be replaced by a job that comes in later and thereby can be discarded. A job may not be placed anymore once it has been discarded. The algorithm stops after I_n has been dealt with. The associated schedule consists of the jobs that have not been discarded in the course of the algorithm.

To be more precise, we need more terminology. We say that machine M is *busy* at time t if $c(M)$ jobs I_v are on machine M with $v < t$ and

$$l(I_t) + 1 \le r(I_v),$$

i.e., I_t overlaps with each I_v. If M is not busy, M is *free*.

An algorithm G for the scheduling problem $(\mathcal{I}, \mathcal{M})$ is a *greedy algorithm* if for $t = 1, \ldots, n$ G acts on I_t according to the following rules:

(G_0) If $\mathcal{M}(I_t)$ contains some free machine M at time t, place I_t onto M.

(G_1) If all machines in $\mathcal{M}(I_t)$ are busy at time t, find some I currently placed on $M \in \mathcal{M}(I_t)$ with $r(I)$ as large as possible. If $r(I) \ge r(I_t) + 1$, replace I by I_t (thereby removing I).

(G_2) If neither (G_0) nor (G_1) applies, discard I_t right away.

So during the course of G, at each time t, every feasible machine M is either free or busy with exactly $c(M)$ jobs. G places I_t on a free feasible machine if such a machine is available. Otherwise G tries to replace a job I_v by I_t on a feasible machine with the goal to reduce the maximal remaining processing time. If neither action is possible, G discards I_t.

Clearly, a greedy algorithm constructs a schedule for $(\mathcal{I}, \mathcal{M})$ that respects the feasibility relation between jobs and machines. Note, however, that (G_0), (G_1), and (G_2) do not specify G uniquely. It is easy to see that different greedy algorithms may produce schedules of different cardinality. We want to bound the worst-case performance of the family of greedy algorithms.

3. SCHEDULING WITH TIME WINDOWS

In this section, we concentrate on the class of interval scheduling problems, where the machine-job feasibility arises in the following way:

With each machine $M \in \mathcal{M}$, there is associated a (finite) set $\mathcal{W}(M)$ of "time windows." Each $W \in \mathcal{W}(M)$ has an "opening time" $l(W)$ and a "closing time" $r(W)$:

(W) $I \in \mathcal{I}$ is feasible for $M \in \mathcal{M}$ if and only if there is a $W \in \mathcal{W}(M)$ with

$$l(W) \le l(I) < r(I) \le r(W).$$

We remark that the class of scheduling problems with time windows amounts to the class of scheduling problems in the general model satisfying the property

$(\tilde{W})$ $l(I) \le l(J) < r(J) \le r(I)$ implies $\mathcal{M}(J) \supseteq \mathcal{M}(I)$ for all $I, J \in \mathcal{I}$.

In our analysis, however, we shall follow the terminology of time windows.

Let $R = \{r(W) : W \in \mathcal{W}(M),\ M \in \mathcal{M}\}$ be the set of the distinct right endpoints of the time windows. If $|R| = s$, then $R = \{r_1, \ldots, r_s\}$, where we assume

$$r_1 < r_2 < \cdots < r_{s-1} < r_s.$$

We furthermore define

$$r_0 := \min\{l(W) : W \in \mathcal{W}(M),\ M \in \mathcal{M}\}.$$

Without loss of generality, we will assume $r_0 = 0$.

The time window W is said to be of *type* m, $1 \le m \le s$, if $r(W) = r_m$. Time windows of the same type are defined to be *equivalent*. If $m \ge 1$, we set $k_0 := 0$ and

$$k_m := \sum c(M),$$

where the summation extends over all machines M having a time window of type m. With the additional notation

$$d_s(k_1, \ldots, k_s) := \sum_{m=0}^{s-1} \min(k_m, k_{m+1} + \cdots + k_s),$$

we can now state our main result.

THEOREM 3.1. *Let $(\mathcal{I}, \mathcal{M})$ be a scheduling problem with time windows. Let furthermore σ be a schedule obtained by some greedy algorithm and let σ^* be an optimal schedule for $(\mathcal{I}, \mathcal{M})$; then*

$$|\sigma^*| - |\sigma| \leq d_s(k_1, \ldots, k_s).$$

Note that the bound in Theorem 3.1 is often quite small; if each machine has capacity 1 and comprises exactly one time window, $d_s(k_1, \ldots, k_s)$ is less than the number k of machines.

Theorem 3.1 generalizes the result of Faigle and Nawijn [4], which can be formulated in the present context as

COROLLARY 3.1. *If $s = 1$, then every greedy schedule is an optimal schedule for $(\mathcal{I}, \mathcal{M})$.*

We prove Theorem 3.1 by showing that a counterexample could be assumed to have a special structure. Further analysis of this structure then shows that no counterexample exists. We first state a technical lemma.

LEMMA 3.1. *Let $(\mathcal{I}, \mathcal{M})$ be a scheduling problem with time windows, and let σ be some greedy schedule for $(\mathcal{I}, \mathcal{M})$. Then there exists a scheduling problem $(\hat{\mathcal{I}}, \mathcal{M})$ and a greedy schedule $\hat{\sigma} = \hat{\sigma}(\hat{\mathcal{I}}, \mathcal{M})$ such that*

(a) $|\hat{\mathcal{I}}| = |\mathcal{I}|$

(b) $|\sigma^*(\hat{\mathcal{I}}, \mathcal{M})| \geq |\sigma^*(\mathcal{I}, \mathcal{M})|$ *and* $|\hat{\sigma}| = |\sigma|$.

(c) *$\hat{\sigma}$ is obtained by a greedy algorithm that never replaces a job but only either places a job on a free machine or discards a job right away.*

Proof. With each sequence $\mathcal{I}$ of jobs, we associate its *weight*,

$$w(\mathcal{I}) := \sum_{I \in \mathcal{I}} l(I).$$

Suppose that, for some $n, l \in \mathbb{N}$, counterexamples to the lemma with n jobs and largest left interval endpoint $l(I_n) = l$ exist. Among all these counterexamples, consider the ones that maximize the weight $w(\mathcal{I})$ of the sequence $\mathcal{I}$ of jobs. In the latter class, finally, choose a counterexample such that the associated greedy algorithm G constructs the greedy sched-

ule σ by a minimal number of replacements. We will derive a contradiction to our choice of G and thus conclude that no counterexample to the lemma exists.

In fact, we claim that G has the properties stipulated in property (c) of the lemma (and hence yields no counterexample). To prove the claim, we have to show that G performs no replacements of jobs.

Suppose, to the contrary, that there exists a minimal time t, $1 < t \leq n$, so that G replaces a (tentatively scheduled) job I_v by the job I_t on a machine M, relative to the scheduling problem $(\mathcal{I}, \mathcal{M})$.

Let us modify job I_v to a job I'_v with $l(I'_v) = l(I_t)$ and $r(I'_v) = r(I_v)$. Putting I'_v immediately before I_t, let $(\mathcal{I}', \mathcal{M})$ denote the modified problem. Let G' be the greedy algorithm on $(\mathcal{I}', \mathcal{M})$ that tries to follow the actions of G whenever possible.

The crucial point to observe is that G' will construct a schedule σ' that is *identical* with σ. To see this, consider a job I_k with $v < k < t$. Because I_t is the first job (under G) that replaces a tentatively scheduled job, G either places I_k on a free machine or discards I_k right away.

If G places I_k on a free machine, G' can apparently do the same. If G discards I_k because M is infeasible for I_k, also G' will have to discard I_k. It, therefore, remains to deal with the case where I_k is feasible for M but was discarded under G. Assume that I_k is placed onto M under G'.

Then we know that $r(I_k) \geq r(I_v)$ holds (otherwise G should have replaced I_v with I_k). Hence machine M will be busy under G' with some job I, say, instead of I_v with $r(I) \geq r(I_v) = r(I'_v)$, when I'_v arrives. G' will not place I'_v onto a machine $M' \neq M$; if M' were free for I'_v under G', it would have been free for I_t under G and if I'_v were to replace some job on M' under G', I_t should have done the same under G (because $r(I'_v) = r(I_v) \geq r(I_t) + 1$). Hence G' will either replace the job I on M with I'_v or it will discard I'_v (namely if $r(I) = r(I'_v)$). Thus, in any case, G' will place I_t onto M. So G and G' will end up having selected exactly the same jobs.

Because, apparently, $|\sigma^*(\mathcal{I}', M)| \geq |\sigma^*(\mathcal{I}, M)|$ holds, $l(I_v) < l(I_t)$ would exhibit $(\mathcal{I}', M)$ to also yield a counterexample, contradicting the assumed maximality of $w(\mathcal{I}) = w(\mathcal{I}') - l(I_t) + l(I_v)$. Therefore, we know that $l(I_v) = l(I_t)$ must be true. If this is the case, however, there is no loss of generality when we assume that I_v is the immediate predecessor of I_t, i.e., $v = t - 1$. Indeed, if there is a job I_k, $v < k < t$, that is discarded by G because of the presence of I_v on M, we have $l(I_k) = l(I_t)$ and

$$r(I_k) \geq r(I_v) > r(I_t).$$

Thus, if we move I_k immediately after I_t in the job sequence, the corresponding greedy schedule will be unaffected. If the presence of I_v on M does not influence the action of G on I_k, moving I_k behind I_t clearly

leaves the schedule and the number of replacements performed by the greedy algorithm unaltered.

Recalling $v = t - 1$ and $l(I_v) = l(I_t)$, let us now "switch" the jobs I_v and I_t in $\mathcal{I}$ to obtain the problem $(\mathcal{I}'', \mathcal{M})$. Let G'' be the greedy algorithm on $(\mathcal{I}'', \mathcal{M})$ that follows the actions of G whenever possible. Then G'' generates the same greedy schedule σ, placing I_t on the free machine M and discarding I_v (if G'' were forced to place I_v onto some other machine M'', then M'' would also be feasible for I_t and G would have had to place I_t onto M'' as well). So G'' will yield a counterexample with less replacements than G, which contradicts the choice of G.

Summarizing the argument, we conclude that G performs no replacements and thus satisfies the lemma. ∎

Before we proceed to the proof of Theorem 3.1, a remark is in order.

Let G be some version of the greedy algorithm that produces the schedule σ for the scheduling problem $(\mathcal{I}, \mathcal{M})$ without performing any replacements of tentatively scheduled jobs. Let $L \in \mathcal{I}$ be a job such that $r(L)$ is maximal in $\mathcal{I}$ and consider the scheduling problem $(\mathcal{I}', \mathcal{M})$, where $\mathcal{I}' = \mathcal{I} \setminus L$.

Take G' to be a greedy algorithm that follows the actions of G as closely as possible. So G' is a greedy algorithm running virtually on the modified problem such that G' takes the same decisions as G whenever possible. Let σ' be the associated greedy schedule of $(\mathcal{I}', \mathcal{M})$. Then the monotonicity property holds:

$$|\sigma'| \leq |\sigma|.$$

Indeed, if G discards L, then apparently $\sigma' = \sigma$ holds. If G places L on some (free) machine M, consider an arbitrary job J arriving after L. Either J is placed on a free machine M' by G (and thus also by G') or J is discarded. If J is discarded because of the presence of L on M, then $r(J) = r(L)$ holds (otherwise G would replace L by J). So G' can place J on M and all future decisions of G' will be the same as those of G. If G discards J for other reasons, G' does the same. So G' can possibly place one such job J onto M instead of L and selects otherwise exactly the same jobs as G.

Note, furthermore, that also G' is a greedy algorithm performing no replacements. (We mention without going into details that this monotonicity property may fail to hold if the greedy algorithm G does not have the special property above.)

As a consequence, we derive a further special structural property that a counterexample to Theorem 3.1 can be assumed to have. To this end, we

introduce the notation

$$\mathcal{L} = \mathcal{L}(\mathcal{I}) := \{I \in \mathcal{I} : r(I) = r_s\}$$

for the subset of "long" jobs in $\mathcal{I}$.

LEMMA 3.2. *Let $(\mathcal{I}, \mathcal{M})$ be a scheduling problem with time windows, and let σ be some greedy schedule for $(\mathcal{I}, \mathcal{M})$. Assume that σ is obtained by a greedy algorithm that never replaces a tentatively scheduled job. Then there exists a scheduling problem $(\hat{\mathcal{I}}, \mathcal{M})$ and a greedy schedule $\hat{\sigma} = \hat{\sigma}(\hat{\mathcal{I}}, \mathcal{M})$ such that property* (c) *of Lemma* 3.1 *holds and*

(i) $|\hat{\mathcal{I}}| \le |\mathcal{I}|$.

(ii) $|\sigma^*(\hat{\mathcal{I}}, \mathcal{M})| \ge |\sigma^*(\mathcal{I}, \mathcal{M})|$ *and* $|\hat{\sigma}| \le |\sigma|$.

(iii) $\mathcal{L} \subseteq \hat{\sigma}^*$ *for every optimal schedule* $\hat{\sigma}^*$ *of* $(\hat{\mathcal{I}}, \mathcal{M})$.

(iv) *The jobs in $\mathcal{L}$ form the tail end of $\hat{\mathcal{I}}$.*

Proof. Suppose that Lemma 3.2 is false. Consider the scheduling problem $(\mathcal{I}, \mathcal{M})$ that satisfies properties (i) and (ii) of the lemma but allows us to exhibit a counterexample with the number $n = |\mathcal{I}|$ of jobs as small as possible. Let G be the realization of the greedy algorithm upon input $(\mathcal{I}, \mathcal{M})$ yielding such a counterexample and let σ be the associated greedy schedule.

We first argue that every optimal schedule σ^* of $(\mathcal{I}, \mathcal{M})$ contains all of $\mathcal{L}$.

Indeed, if there existed some job $L \in \mathcal{L}$ with $L \notin \sigma^*$, then the problem $(\mathcal{I} \setminus L, \mathcal{M})$ would still yield σ^* as an optimal schedule. The monotonicity property above would, therefore, imply that also $(\mathcal{I} \setminus L, \mathcal{M})$ gives rise to a counterexample, which would contradict our minimal choice of $(\mathcal{I}, \mathcal{M})$.

Second, we claim that we can assume the jobs in $\mathcal{L}$ form the tail end of the sequence $\mathcal{I}$. To support this claim, we will separately consider the case where L is discarded by G and the case where L is accepted by G.

Consider first the last job I_n. If the greedy algorithm G placed I_n on a free machine, then clearly $(\mathcal{I} \setminus I_n, \mathcal{M})$ would offer a counterexample as well, contradicting the minimality of our counterexample. So we know that I_n is discarded by G.

If the job $L \in \mathcal{L}$ is discarded by G, modify L to L' such that $l(L') = l(I_n)$ and $r(L') = r(L) = r_s$ and place L' immediately after I_n. Then the corresponding greedy algorithm G' will discard L' as well (otherwise, I_n could have been placed by G!). Hence, the nature of the counterexample remains unaltered.

If $L \in \mathcal{L}$ is placed by G on a machine M, let us again modify L to L' as before and move L' behind I_n. Suppose the corresponding run of G will now place a successor J of L which could not be placed by G before.

Then $J \in \mathscr{L}$ (otherwise G would have been forced to replace L with J in the former situation). Because $r(J) = r_s$, L' will be discarded (otherwise, the machine available for L' would have been free for I_n before). In other words, the modification does not improve the performance of the greedy algorithm and yields a counterexample of the same type as before.

In this way, we can transform the scheduling problem until property (iv) holds, i.e., until the lemma is satisfied, which contradicts the choice of $(\mathscr{I}, \mathscr{M})$ as a counterexample. ∎

The proof of Theorem 3.1 can now be finished by induction on s.

Proof of Theorem 3.1. Suppose the theorem is false. By Lemma 3.1 and Lemma 3.2, we can find a counterexample $(\mathscr{I}, \mathscr{M})$ with corresponding greedy algorithm G such that

(α) G never replaces a tentatively scheduled job.

(β) Every optimal schedule σ^* contains $\mathscr{L}$.

(γ) $\mathscr{L}$ forms the tail end of $\mathscr{I}$.

Among all the counterexamples satisfying the properties (α)–(γ), assume that $(\mathscr{I}, \mathscr{M})$ minimizes the number $n = |\mathscr{I}|$ of jobs.

The first observation to make is that $r(L) - l(L) \geq 2$ must hold for every $L \in \mathscr{L}$.

Suppose to the contrary that there exists some $L \in \mathscr{L}$ with $l(L) = r_s - 1$. Then also $l(I_n) = r_s - 1$ must hold; i.e., we can assume $L = I_n$. G will necessarily schedule I_n. This is so because $\mathscr{L} \subseteq \sigma^*$ and, therefore, not all machines can be busy when I_n arrives (otherwise, the machine capacity would not be enough to include *all* of $\mathscr{L}$ in any feasible schedule). Consequently, $(\mathscr{I} \setminus I_n, \mathscr{M})$ would offer a smaller counterexample to the theorem, which contradicts our choice of $(\mathscr{I}, \mathscr{M})$.

Assume that the counterexample has been chosen so that (in addition to the previous assumptions) the parameter r_s is minimal and that $\mathscr{M}$ contains no time window of length 0 (the latter assumption clearly can be made without any loss of generality). Our second observation is that the following property must hold:

(δ) $r_{s-1} = r_s - 1$.

To arrive at (δ), suppose to the contrary that $r_s \geq r_{s-1} + 2$. Consider then the problem $(\mathscr{I}', \mathscr{M}')$ that arises from $(\mathscr{I}, \mathscr{M})$ by replacing each job $L \in \mathscr{L}$ with a job L' such that

$$l(L') = l(L), \quad r(L') = r_s - 1$$

and each time window W with $r(W) = r_s$ by a time window W' such that

$$l(W') = l(W), \quad r(W') = r_s - 1.$$

Because $r(L) - l(L) \geq 2$ for all $L \in \mathcal{L}$, the scheduling problem $(\mathcal{I}', \mathcal{M})$ is completely equivalent with $(\mathcal{I}, \mathcal{M})$, but it has a smaller parameter r_s which is impossible. So (δ) holds.

We now come to the inductive step of the proof of the theorem. Suppose there were a counterexample $(\mathcal{I}, \mathcal{M})$ with $s = 1$. By property (δ), we then can assume

$$0 = r_0 = r_1 - 1 = r_s - 1.$$

This, however, means that the scheduling problem is trivial and the greedy algorithm is clearly optimal. In other words, no counterexample with $s = 1$ exists.

Let us assume now that $(\mathcal{I}, \mathcal{M})$ is a counterexample to the theorem with properties (α)–(δ) above and satisfies $s > 1$. Moreover, we assume that the bound in the theorem holds for all problems $(\mathcal{I}, \bar{\mathcal{M}})$ whenever $\bar{\mathcal{M}}$ has fewer equivalence classes of time windows than $\mathcal{M}$. We will show that this assumption leads to a contradiction and thus we conclude that no counterexample exists.

In view of the recurrence relation

$$d_s(k_1, \ldots, k_s) = d_{s-1}(k_1, \ldots, k_{s-2}, k_{s-1} + k_s) + \min(k_{s-1}, k_s),$$

there are two cases to be investigated.

Case 1. $\min(k_{s-1}, k_s) = k_{s-1}$. Consider the problem $(\mathcal{I}, \mathcal{M}')$, where $\mathcal{M}'$ arises from $\mathcal{M}$ by replacing each time window W with $r(W) = r_{s-1}$ by a time window W' so that

$$l(W') = l(W), \quad r(W') = r_s.$$

Denote by k'_m the parameters of the modified machine structure. Thus $k'_{s-1} = k_{s-1} + k_s$ and the theorem holds for $(\mathcal{I}, \mathcal{M}')$ by the minimality of the number s of distinct equivalence classes of time windows.

LEMMA 3.3. *There exists a greedy schedule* $\sigma' = \sigma'(\mathcal{I}, \mathcal{M}')$ *so that*

$$|\sigma'| \leq |\sigma| + k_{s-1}.$$

Proof. This is an immediate consequence of our assumption that G either places every job on a free machine or discards it right away. ∎

Obviously, $|\sigma^*(\mathcal{I}, \mathcal{M}')| \geq |\sigma^*(\mathcal{I}, \mathcal{M})|$. So we conclude from Lemma 3.3 that

$$\begin{aligned} |\sigma^*(\mathcal{I}, \mathcal{M})| - |\sigma(\mathcal{I}, \mathcal{M})| &\leq |\sigma^*(\mathcal{I}, \mathcal{M}')| - |\sigma'(\mathcal{I}, \mathcal{M}')| + k_{s-1} \\ &\leq d_{s-1}(k_1, \ldots, k_{s-1}, k_{s-1} + k_s) \\ &\quad + \min(k_{s-1}, k_s) \\ &= d_s(k_1, \ldots, k_{s-1}, k_s), \end{aligned}$$

a contradiction to the choice of $(\mathcal{I}, \mathcal{M})$ as a counterexample.

Case 2. $\min(k_{s-1}, k_s) = k_s$. Consider the problem $(\mathcal{I}'', \mathcal{M}'')$, where $\mathcal{M}''$ arises from $\mathcal{M}$ by replacing each time window W with $r(W) = r_s$ of $\mathcal{M}$ by a time window W'' so that

$$l(W'') = l(W), \qquad r(W'') = r_{s-1},$$

and $(\mathcal{I}'')$ arises from $\mathcal{I}$ by replacing each $I \in \mathcal{I}$ with $r(I) = r_s$ by a job I'' so that

$$l(I'') = l(I), \quad r(I'') = r(I) - 1 = r_{s-1}.$$

(Note that $\mathcal{I}''$ is a feasible list of jobs since each $L \in \mathcal{L}$ has length at least 2.)

LEMMA 3.4. *There exists a greedy schedule* $\sigma'' = \sigma''(\mathcal{I}'', \mathcal{M}'')$ *so that*

$$|\sigma''(\mathcal{I}'', \mathcal{M}'')| \le |\sigma(\mathcal{I}, \mathcal{M})|.$$

Proof. Again, this follows directly from the fact that G performs no replacements of scheduled jobs. ▮

Because $r_{s-1} = r_s - 1$, $|\sigma^*(\mathcal{I}, \mathcal{M})| \le |\sigma^*(\mathcal{I}'', \mathcal{M}'')| + k_s$ holds. Thus we conclude as before,

$$\begin{aligned} |\sigma^*(\mathcal{I}, \mathcal{M})| - |\sigma(\mathcal{I}, \mathcal{M})| &\le |\sigma^*(\mathcal{I}'', \mathcal{M}'')| - |\sigma''(\mathcal{I}'', \mathcal{M}'')| + k_s \\ &\le d_{s-1}(k_1, \ldots, k_{s-1}, k_{s-1} + k_s) \\ &\quad + \min(k_{s-1}, k_s) \\ &= d_s(k_1, \ldots, k_{s-1}, k_s), \end{aligned}$$

a contradiction to the choice of $(\mathcal{I}, \mathcal{M})$ as a counterexample.

Hence no counterexample to Theorem 3.1 can exist. Q.E.D.

We finish this section with an inductive construction of scheduling problems showing that the bound $d_s(k_1, \ldots, k_s)$ in Theorem 3.1 is best possible (see, however, an alternative bound in the next section!) for greedy on-line algorithms.

Every machine M in our set of problems has capacity $c(M) = 1$ and has exactly one time window. We will, therefore, identify the machines with the time windows for notational convenience. We write for $m = 1, \ldots, s$,

$$\mathcal{M}_m := \{M \in \mathcal{M}: r(M) = r_m\}.$$

Of course, the bound $d_1(k_1) = 0$ is tight for $s = 1$.

Let $(\mathcal{I}, \mathcal{M})$ be a problem with $s-1$ classes $\mathcal{M}_m$ of machines such that

$$|\mathcal{M}_m| = \begin{cases} k_m, & \text{if } m \le s-2, \\ k_{s-1} + k_s, & \text{if } m = s-1, \end{cases}$$

and assume that there is a greedy schedule σ for $(\mathcal{I}, \mathcal{M})$ for which the bound

$$d_{s-1}(k_1, \ldots, k_{s-2}, k_{s-1} + k_s)$$

is tight. There is no loss of generality when we assume that $r(I) \le r_{s-1}$ holds for all $I \in \mathcal{I}$.

Choose k_{s-1} machines in $\mathcal{M}_{s-1}$ and enlarge the right endpoints of their time windows to

$$r'_{s-1} := r_{s-1} + 2.$$

Enlarge the right endpoints of the remaining k_s machines in $\mathcal{M}_{s-1}$ to

$$r'_s := r_{s-1} + 3$$

and denote by $\mathcal{M}'$ the new set of machines.

Because $r(I) \le r_{s-1}$ for all $I \in \mathcal{I}$, the greedy algorithm for $(\mathcal{I}, \mathcal{M})$ can be carried out identically as a greedy algorithm for $(\mathcal{I}, \mathcal{M}')$.

Now append the following new jobs to $\mathcal{I}$:

k_{s-1}	jobs with associated intervals	$[r_{s-1}, r_{s-1}+1]$
$\min(k_{s-1}, k_s)$	jobs with associated intervals	$[r_{s-1}, r_{s-1}+2]$
k_s	jobs with associated intervals	$[r_{s-1}+1, r_{s-1}+3]$.

Continue the greedy algorithm for $(\mathcal{I}, \mathcal{M})$ by placing the first k_{s-1} new jobs on $\mathcal{M}'_{s-1}$, the next $\min(k_{s-1}, k_s)$ new jobs on $\mathcal{M}'_s$ and, finally, $k_s - \min(k_{s-1}, k_s)$ of the remaining new jobs on $\mathcal{M}'_s$.

The resulting greedy schedule σ' then satisfies

$$|\sigma'| = |\sigma| + k_{s-1} + k_s.$$

On the other hand, an optimal schedule σ^{**} for the augmented problem consists of an optimal schedule σ^* for $(\mathcal{I}, \mathcal{M})$ plus an optimal assignment of the new jobs to $\mathcal{M}'_{s-1} \cup M'_s$. Thus,

$$|\sigma^{**}| = |\sigma^*| + k_{s-1} + k_s + \min(k_{s-1}, k_s),$$

which yields

$$|\sigma^{**}| - |\sigma'| = d_s(k_1, k_2, \ldots, k_{s-1}, k_s).$$

4. REMARKS AND OPEN PROBLEMS

There is an alternative approach to bounding the performance of the greedy algorithm for interval scheduling problems with time windows.

THEOREM 4.1. *Let σ be a greedy schedule for the problem $(\mathcal{I}, \mathcal{M})$ with time windows. Then*

$$|\sigma^*| \leq 2|\sigma|.$$

Proof. Suppose the theorem is false. By Lemma 3.1, we can then find a counterexample $(\mathcal{I}, \mathcal{M})$ with associated greedy algorithm G such that G never replaces a tentatively scheduled job.

Note, however, that we can assume the capacity to satisfy $c(M) = 1$ for each machine $M \in \mathcal{M}$. Indeed, we can replace each M by $c(M)$ identical copies M' of M with capacity $c(M') = 1$ and obtain a completely equivalent scheduling problem. We, therefore, make this assumption for the remainder of the proof.

With each $I \in \sigma^* \setminus \sigma$ we want to associate a job $f(I) \in \sigma$ such that

$$f(I) \neq f(J) \quad \text{whenever } I \neq J.$$

Clearly, the bound of the theorem will follow if we can construct such an injection f.

Consider a fixed assignment of the optimal schedule σ^* to the machines. If $I \in \sigma^*$ is discarded by G, the machine M^* to which I is assigned in σ^* is busy with some job I' when I arrives. I' is a member of σ because G never replaces a tentatively scheduled job. Set $f(I) := I'$.

Let $f(I) = f(J)$ for some $I, J \in \sigma^* \setminus \sigma$. W.l.o.g. assume

$$l(f(I)) \leq l(I) \leq l(J).$$

Then I and J overlap (otherwise, I should have replaced $f(I)$ on M^*, instead of being discarded by G). But $c(M^*) = 1$ yields that I and J cannot both be feasibly be assigned to M^* unless $I = J$. ∎

In view of Theorem 4.1, the greedy on-line algorithm is also optimal for the class of scheduling problems with property $(\tilde{W})$ in the following sense: For every on-line algorithm A, there exists a problem with property $(\tilde{W})$ on which A achieves at most half of the theoretical optimum.

EXAMPLE. Let $(\mathcal{I}, \mathcal{M})$ be a scheduling problem with $\mathcal{M} = \{M_1, M_2\}$ and $l(I_t) = 2t$ for all $I_t \in \mathcal{I}$ and assume

$$r(I_t) = \begin{cases} 2t + 2 & \text{if } t \text{ is even,} \\ 2t + 3 & \text{if } t \text{ is odd.} \end{cases}$$

Assume that both machines are feasible for the odd jobs. Let now A be some on-line scheduling algorithm and let for every even job exactly the machine be feasible on which A has placed the preceding odd job. Clearly, A will not achieve more than half of the theoretical optimum.

If the time windows are explicitly known to the scheduler, however, it might be possible that on-line algorithms with an even better performance guarantee than the one in Theorem 3.1 can be obtained. In particular, it would be interesting to have an analysis of probabilistic on-line scheduling under time windows.

Theorem 4.1 is false for the general class of interval scheduling problems.

EXAMPLE. Consider four machines M_1, M_2, M_3, M_4 and an interval sequence with five jobs $I_1 = I_2 = [0, 4]$, $I_3 = [0, 3]$, $I_4 = [0, 2]$, $I_5 = [2, 4]$. I_1 and I_2 are feasible for all machines, I_3 and I_4 for machines $\{M_1, M_2\}$ and I_5 only for M_1.

Assume the greedy algorithm G places I_1 and I_2 on M_1 and M_2 and then replaces I_1 on M_1 by I_3 and I_2 on M_2 by I_4. So I_5 will be discarded and G achieves a schedule for only 2 jobs, while the theoretical optimum is 5. ∎

Using a larger number of machines, the idea of this example can be extended to construct instances for which the greedy algorithm in the worst case schedules only approximately one third of the theoretically optimal number of jobs.

We do not know whether there is an on-line algorithm in our model with bounded performance ratio on the class of general interval scheduling problems.

REFERENCES

1. E. M. Arkin and E. B. Silverberg, Scheduling jobs with fixed start and end times, *Discrete Appl. Math*. **18** (1987), 1–8.
2. P. Brucker and L. Nordmann, The *k*-track assignment problem, *J. Computing* **52** (1994), 97–122.
3. M. C. Carlisle and E. L. Lloyd, On the *k*-coloring of intervals, *in* "Advances in Computing and Information—ICCI'91," (F. Dehne et al., Eds.), Lecture Notes in Computer Science, Vol. 497, pp. 90–101, Springer-Verlag, New York/Berlin, 1991.
4. U. Faigle and W. M. Nawijn, Note on scheduling intervals on-line, *Discrete Appl. Math*. **58** (1995), 13–17.
5. M. R. Garey, D. S. Johnson, G. L. Miller, and C. H. Papadimitriou, The complexity of coloring circular arcs and cords, *SIAM J. Algebraic Discrete Methods* **1** (1980), 216–227.
6. A. W. J. Kolen and J. K. Lenstra, Interval scheduling, *in* "Handbook of Combinatorics, Vol. II," (R. Graham et al., Eds.), pp. 1901–1904, Elsevier, Amsterdam, 1995.