ELSEVIER

Discrete Applied Mathematics 58 (1995) 13–17

DISCRETE APPLIED MATHEMATICS

Note on scheduling intervals on-line

Ulrich Faigle*, Willem M. Nawijn

Department of Applied Mathematics, University of Twente, P.O. Box 217, NL-7500 AE Enschede, Netherlands

Received 19 November 1991; revised 21 October 1994

Abstract

An optimal on-line algorithm is presented for the following optimization problem, which constitutes the special case of the k-track assignment problem with identical time windows. Intervals arrive at times t_i and demand service time equal to their length. An interval is considered lost if it is not assigned to one of k identical service stations immediately or if its service is interrupted. Minimizing the losses amounts to coloring a maximal set of intervals in the associated interval graph properly with at most k colors. Optimality of the on-line algorithm is proved by showing that it performs as well as the optimal greedy k-coloring algorithm due to Faigle and Nawijn and, independently, to Carlisle and Lloyd for the same problem under full *a priori* information.

Keywords: Interval order; Scheduling; k-track assignment; Coloring; Greedy algorithm; On-line algorithm

1. Introduction

In this note, we study the following on-line scheduling problem. Intervals arrive at times corresponding to their left endpoints and demand service time by one of k identical servers equal to their length. An interval is *lost* if it is not serviced immediately and uninterruptedly. We are to minimize the number of losses. This problem amounts to determining a maximal k-colorable subgraph of the interval graph associated with the intervals arriving. Equivalently, we are to find a maximal subset of intervals that can be covered by k chains relative to the associated interval order. Relaxing the model requirements to possibly non-identical servers, we arrive at *loss systems* (see, e.g., [8]). It is an open problem to find optimal on-line algorithms for this general model.

Our problem can also be viewed as a special case of the *k-track assignment problem* (see, e.g., [1]). There, the server m, is available only during the *time window* $[w_m, w'_m]$.

* Corresponding author. E-mail: faigle@math. utwente.nl.

SSDI 0166-218X(95)00112-X

Our problem is thus equivalent to the k-track assignment problem with identical time windows. Indeed, we may assume without loss of generality that only intervals arrive that fit into the given time window. No optimal on-line algorithms are known for the general k-track assignment problem.

Our model entails that we have to assign the intervals *on-line*, based only on the partial information implied by the intervals seen so far. Yet, it turns out that there is an on-line algorithm that matches the performance of an optimal algorithm based on full a priori information. The latter is the greedy k-covering algorithm of Faigle and Nawijn [4], which was independently discovered also by Carlisle and Lloyd [2]. We review the k-greedy algorithm briefly in Section 2 and present the on-line algorithm in Section 3.

The existence of such an optimal on-line algorithm may be somewhat surprising in view of the following variant of the scheduling problem. While our model allows us to replace an interval that is currently being serviced by another interval (at the expense of losing the first interval, of course), one may require that an interval once assigned has to be serviced without interruption. Now the performance of any on-line algorithm can be arbitrarily bad compared with the performance of the k-greedy algorithm under full information. As it turns out, one can improve the on-line performance in the latter model by randomization. For details, we refer to [3, 7].

2. k-greedy coverings of interval orders

Let $\mathcal{I}$ be a (finite) set of compact intervals on the real line. Then $\mathcal{I}$ is naturally ordered via the relation

$$I \prec J \quad \text{whenever } r(I) < l(J),$$

where $r(I)$ and $l(J)$ denotes the right and left endpoint of $I, J \in \mathcal{I}$, respectively. The (partial) order induced this way on the collection of intervals is an *interval order*.

Given the parameter $k > 0$, we consider the problem of finding subsets $C_1, C_2, \ldots, C_k$ of intervals such that each C_i is a chain in the interval order and

$$|C_1 \cup C_2 \cup \cdots \cup C_k|$$

is as large as possible. Equivalently, of course, we may consider the associated *interval graph*, i.e., the incomparability graph of the interval order, and ask for a maximum cardinality k-colorable subset of nodes in the interval graph.

The k-covering problem for chains is well-known to be polynomial for any order relation (see, e.g., [5]). For interval orders, Faigle and Nawijn [4] exhibited a direct greedy algorithm to be successful (the same algorithm was proposed independently by Carlisle and Lloyd [2]).

The *k-greedy algorithm* (GLF for short) follows a last-fit strategy. It assumes that in a preprocessing phase the intervals have been ordered $I_1, I_2, \ldots, I_n$ according to increasing right endpoints, i.e., such that $r(I_1) \leqslant r(I_2) \leqslant \cdots \leqslant r(I_n)$.

In this order, the intervals are now processed by assigning labels from the set $\{1,\ldots,k\}\cup\{\$\}$. In Step (i) of GLF, the interval I_i is treated as follows:

(i): Try to find an interval I_j of maximal index $j < i$ such that

(a) $lab(I_j) \neq \$$,

(b) I_j is maximal in the chain of intervals with label $lab(I_j)$,

(c) $I_j \prec I_i$.

If such an interval I_j exists set $lab(I_i) = lab(I_j)$. If no such I_j exists and there is a label l that has not been used so far, set $lab(I_i) = l$. Otherwise, set $lab(I_j) = \$$.

The intervals I receiving label $lab(I) = \$$ may be considered to be discarded. The remaining intervals clearly decompose into at most k chains according to the labels received during the execution of GLF.

Theorem 2.1 (Faigle and Nawijn [4] and Carlisle and Lloyd [2]). *Algorithm GLF solves the k-covering problem optimally for interval orders.*

We remark that Theorem 2.1 also applies in a seemingly more general context. Let $G = (V, A)$ be a directed acyclic graph and define an order relation P on the set V of vertices of G via

$$x\,P\,y \quad \text{if there is a directed path from } x \text{ to } y.$$

Say that G is *interval* if P is isomorphic to some interval order. Using the recognition algorithm of Gabow [6], one can find a permutation $x_1\,x_2\,\ldots\,x_n$ of the elements of V such that $N(x_i) \supseteq N(x_{i+1})$ for $i = 1,\ldots,n-1$, where

$$N(x) = \{y \in V \mid x\,P\,y\}.$$

From this it is not difficult to obtain an explicit representation of P by compact intervals that obeys the preprocessing requirements of algorithm GLF. This preprocessing step can be carried out in time polynomial in $|V| + |A|$.

3. On-line scheduling of interval ordered tasks

In this section, we deal with the following problem. Clients i, $i = 1, 2, \ldots$, arrive at (a priori unknown) times t_i. Once client i arrives, he demands $p_i \geqslant 0$ time units of service time, where also p_i only becomes known at the moment of arrival time t_i. If a client is not served immediately or if his service is interrupted, he is considered "lost".

There are k identical service stations at our disposal. Each station may serve at most one customer at a time but may interrupt the service in order to start serving a new customer without setup time.

The problem consists in assigning the clients, as they arrive, to service stations so that the total number of losses is minimized.

Assuming $t_1 \leqslant t_2 \leqslant \cdots$, we define the intervals $I_i = [t_i, t_i + p_i]$ and consider the interval order P of the first n intervals $I_1, I_2, \ldots, I_n$ arriving. Note that an optimal assignment of P to the k service stations corresponds to the constructions of k chains in P that comprise as many intervals as possible. Our model, however, does not allow us to apply algorithm GLF of the previous section because the *right* endpoints of the intervals presented in time are not necessarily increasing. In fact, the preprocessing required for an application of GLF can only be done at time t_n when the *full* information on P is available.

We, therefore, investigate the following strategy which processes the interval $I_1, I_2, \ldots$ in the order of their arrival, i.e., in the order of their *left* endpoints.

Greedy On-line Algorithm (GOL)

(1): Assign I_1 to an arbitrary service station.

(i): Assign I_i to any free service station.
If all k stations are busy at time t_i, find a currently assigned I whose right endpoint $r(I)$ is as large as possible.
If $r(I_i) \geqslant r(I)$, then discard I_i.
If $r(I_i) < r(I)$, then replace I by I_i.

Theorem 3.1. *Algorithm GOL minimizes the number of losses.*

We prove Theorem 3.1 by comparing the performance of GOL with that of GLF applied to the suitably preprocessed interval order P. To this end, let $L = I^1 I^2 \ldots I^n$ be an arrangement of the members of P such that for $i = 1, \ldots, n-1$,

(a) $r(I^i) \leqslant r(I^{i+1})$,

(b) $r(I^i) = r(I^{i+1})$ implies $l(I^i) \leqslant l(I^{i+1})$.

In view of Theorem 2.1, the proof of Theorem 3.1 now follows from the next lemma.

Lemma 3.1. *If algorithm GOL looses some interval $I \in P$, then $lab(I) = \$$ when algorithm GLF is applied to L. Hence the optimality of GLF implies the optimality of GOL.*

Proof. Without loss of generality, we assume

$$t_1 = l(I_1) < l(I_2) < \cdots < l(I_n) = t_n.$$

Furthermore, it is convenient to introduce a technical modification in algorithm GOL:

(i'): If I_i can be assigned to a free service station, choose the station where the last assigned interval I had $r(I)$ as large as possible. Otherwise proceed as in Step (i) of GOL.

It is not hard to see that the modification does not affect the collection of intervals lost by algorithm GOL.

Suppose now that Lemma 3.1 is false and let i be the smallest index such that $lab(I^i) \neq \$$ but I^i is lost in algorithm GOL. We analyze the situation by distinguishing two cases.

Case 1: I^i *is discarded by GOL.* This means that, at time t_i, each station is already assigned an interval I with $r(I) \leqslant r(I^i)$. In particular, all intervals not lost by GOL so far are in the set $\{I^1, I^2, \ldots, I^{i-1}\}$.

Continue now with GOL until all of $\{I^1, I^2, \ldots, I^{i-1}\}$ is processed. Because the index i is chosen minimal, we know that the intervals not lost by GOL and having right endpoint not exceeding $r(I^i)$ are identical with those intervals $I \in \{I^1, I^2, \ldots, I^{i-1}\}$ satisfying $lab(I) \neq \$$.

Note that GOL will at most replace intervals assigned by intervals that will block an assignment of I^i. So when GLF is to process I^i, k intervals will already be assigned that are incomparable with I^i. Hence we conclude $lab(I^i) = \$$, a contradiction.

Case 2: I^i *is assigned at time* t_i *and replaced by* I_j *at time* $t_j > t_i$. After the replacement, as in Case 1, all intervals not lost by GOL are in $\{I^1, I^2, \ldots, I^{i-1}\}$ (otherwise I^j would have replaced some interval other than I^i).

Continue with GOL until all of $\{I^1, I^2, \ldots, I^{i-1}\}$ has been processed. By the modification step (i'), the result will be identical with GLF applied to $\{I^1, I^2, \ldots, I^{i-1}\}$ directly. As in Case 1, we see that $lab(I^i) = \$$ must be true, contradicting our assumption $lab(I^i) \neq \$$. □

We close by remarking that the optimality of the algorithm GOL (Theorem 3.1) may not be as "obvious" as one intuitively feels. Let us relax the algorithm GOL in Step (i) by allowing I_i to replace *any* currently assigned I with $r(I) > r(I_i)$. Then GOL is still "locally optimal" but may no longer be globally optimal!

Example. Let $I_1 = [0,4]$, $I_2 = [1,8]$, $I_3 = [2,3]$, $I_4 = [5,7]$, $I_5 = [6,9]$. With $k = 2$, I_1 and I_2 will be assigned at the moment I_3 arrives. If I_3 now replaces I_1 instead of I_2, the final number of losses will be 2, which is not optimal.

References

[1] P. Brucker and L. Nordmann, The k-track assignment problem, preprint No. 157, FB Mathematik/Informatik, Universität Osnabrück (1993).

[2] M.C. Carlisle and E.L. Lloyd, On the k-coloring of intervals, in: F Dehne, F. Fiala and W.W. Koczkodaj, eds., Advances in Computing and Information – ICCI'91, Lecture Notes in Computer Science, 497 (Springer, Berlin, 1991) 90–101.

[3] U. Faigle, R. Garbe and W. Kern, Randomized online algorithms for maximizing busy time interval scheduling, preprint, Department of Applied Mathematics, University of Twente (1994).

[4] U. Faigle and W.M. Nawijn, Greedy k-decomposition of interval orders, in: U. Faigle and C. Hoede, eds., Proceedings 2nd Twente Workshop on Graphs and Combinatorial Optimization, Memorandum No. 964, Department of Applied Mathematics, University of Twente (1991) 53–56.

[5] A. Frank, On chain and antichain families of a partially ordered set, J. Combin. Theory Ser. B 29 (1980) 176–184.

[6] H.N. Gabow, A linear-time recognition algorithm for interval dags, Inform. Process. Lett. 12 (1981) 20–22.

[7] R.J. Lipton and A. Tomkins, Online interval scheduling, in: Proceedings Symposium on Discrete Algorithms, 23–25 January 1994, Arlington, VA (to appear).

[8] W.M. Nawijn, Minimum loss scheduling problems, European J. Oper. Res. 56 (1992) 364–369.