ACADEMIC PRESS

Available at WWW.MATHEMATICSWEB.ORG POWERED BY SCIENCE DIRECT®

Journal of Algorithms 46 (2003) 27–53

Journal of Algorithms

www.elsevier.com/locate/jalgor

Interval selection: Applications, algorithms, and lower bounds ☆

Thomas Erlebach [a] and Frits C.R. Spieksma [b,*]

[a] *Computer Engineering and Networks Laboratory (TIK), ETH Zürich, Gloriastr. 35, CH-8092 Zürich, Switzerland*

[b] *Quantitative Methods, Faculty of Economic and Applied Economic Sciences, Katholieke Universiteit Leuven, Naamsestraat 69, B-3000 Leuven, Belgium*

Received 4 April 2001

Abstract

Given a set of jobs, each consisting of a number of weighted intervals on the real line, and a positive integer m, we study the problem of selecting a maximum weight subset of the intervals such that at most one interval is selected from each job and, for any point p on the real line, at most m intervals containing p are selected. We give a parameterized algorithm GREEDY$_\alpha$ that belongs to the class of "myopic" algorithms, which are deterministic algorithms that process the given intervals in order of non-decreasing right endpoint and can either reject or select each interval (rejections are irrevocable). We show that there are values of the parameter α so that GREEDY$_\alpha$ produces a 2-approximation in the case of unit weights, an 8-approximation in the case of arbitrary weights, and a $(3+2\sqrt{2})$-approximation in the case where the weights of all intervals corresponding to the same job are equal. We also show that no deterministic myopic algorithm can achieve ratio better than 2 in the case of unit weights, better than ≈ 7.103 in the case of arbitrary weights, and better than $3+2\sqrt{2}$ in the case where the weights of all intervals corresponding to the same job are equal. Furthermore, we give additional results for the case where all intervals have the same length as well as a lower bound of $\frac{e}{e-1} \approx 1.582$ on the approximation ratio of randomized myopic algorithms in the case of unit weights.

Keywords: Intervals; Algorithms; Approximation; Lower bounds; Independent set

☆ A preliminary version of some of the results in this paper has appeared in [Proc. 11th Ann. Internat. Symp. on Algorithms and Computation, ISAAC'00, in: Lecture Notes in Comput. Sci., Vol. 1969, 2000, pp. 228–240].

* Corresponding author.

E-mail addresses: erlebach@tik.ee.ethz.ch (T. Erlebach), frits.spieksma@econ.kuleuven.ac.be (F.C.R. Spieksma).

0196-6774/03/$ – see front matter
PII: S0196-6774(02)00291-2

1. Introduction

We study a *weighted job interval selection problem*, called WJISP. The input consists of jobs, each of which is given by a set of intervals on the real line, and a number m of available machines. (We use "WJISP$_m$" instead of "WJISP" if we want to make an explicit reference to the number m of machines.) Each interval i has a positive weight $w(i)$. A feasible solution is a subset of the given intervals such that

(1) at most one interval is selected from each job, and
(2) for any point p on the real line, at most m intervals overlapping p are selected.

The goal is to find a feasible solution that maximizes the sum of the weights of the selected intervals. We let n denote the total number of intervals in the input. We assume that a sorted list of all interval endpoints is available; such a list can be constructed in time $O(n \log n)$.

Notice that the requirement that any point on the real line is overlapped by at most m selected intervals is equivalent to the requirement that the selected intervals can be partitioned into m subsets such that the intervals in each subset are pairwise disjoint. In some applications (see Section 2) this partition (in addition to specifying the selected intervals) is required as output. However, the subproblem of computing such a partition given the selected intervals can be solved efficiently by coloring the corresponding interval graph. In fact, n intervals can be colored in time $O(n)$ if the sorted list of interval endpoints is given. Therefore, we concentrate here on the problem of selecting the intervals and assume that, if required, an appropriate coloring procedure is employed to compute the partitioning.

WJISP$_1$ can be formulated in graph-theoretic terms. By constructing a graph such that there is a vertex for each interval, and two vertices are connected via an edge if the corresponding intervals overlap or if they belong to the same job, WJISP$_1$ can be viewed as a maximum-weight independent set problem in a graph. This graph is the edge union of an interval graph and a graph that consists of a disjoint union of cliques (cliques correspond to jobs).

There are several restricted versions of WJISP that are interesting (see the applications described in Section 2). We distinguish a number of variants of WJISP. Regarding the weights we consider:

- the *unweighted case* (called JISP), which refers to the case where each interval has the same weight w,
- *WJISP with equal weights per job*, which refers to instances of WJISP in which intervals that belong to the same job have the same weight, but intervals that belong to different jobs can have different weights, and finally
- *WJISP with arbitrary weights.*

Another interesting restriction pertains to the length of the intervals. One can distinguish in a similar fashion: *WJISP with equal lengths*, which refers to the case where each interval has the same length, *WJISP with equal lengths per job*, which refers to instances of WJISP in which intervals that belong to the same job have the same length, and finally *WJISP with arbitrary lengths*.

We consider a class of simple deterministic algorithms for WJISP and investigate so-called *worst-case ratios* (or approximation ratios) that can be obtained by algorithms within this class. Using standard terminology (see, e.g., Hochbaum [17], Ausiello et al. [1]), we say that a deterministic algorithm for WJISP achieves (approximation) ratio ρ if it always outputs a feasible solution whose weight is at least as large as $1/\rho$ times the weight of an optimal solution. A randomized algorithm achieves approximation ratio ρ if, on every instance of WJISP, the expected weight of the solution computed by the algorithm is at least $1/\rho$ times the weight of an optimal solution.

A natural class of algorithms to consider for WJISP instances is the class of *single-pass* algorithms. Very generally stated, single-pass algorithms are algorithms in which a feasible solution is obtained by iteratively making a decision concerning an item or an object. The first-fit decreasing algorithm for the bin packing problem and the nearest neighbor algorithm for the traveling salesman problem are prime examples of single-pass algorithms. This kind of algorithms can be useful since they need little computing time and/or little information (i.e., they can be applied in an on-line setting). In our context, we call an algorithm a single-pass algorithm when given some sequence of the intervals, each interval is (iteratively) either rejected or accepted (selected) without considering the intervals that will be processed later. Rejections are permanent, but an accepted interval can be rejected (preempted) at a later time. At any time, the set of currently selected intervals must be a feasible solution. After the last interval is presented, the set of currently selected intervals is taken as the solution computed by the algorithm.

When considering a specific single-pass algorithm for WJISP, it is crucial to specify the mechanism that determines the sequence in which the intervals will be processed. Different choices are possible, for instance processing the intervals in order of non-increasing weight, or processing the intervals in order of non-decreasing left endpoint or non-decreasing right endpoint. However, it is easy to see that single-pass algorithms that process the intervals in order of non-increasing weight or in order of non-decreasing left endpoint do not have a finite worst-case ratio (even in the case when each job consists of one interval only, see Woeginger [28], and even if randomization is allowed, see Canetti and Irani [6]). Therefore, we investigate in this paper the special class of single-pass algorithms, which we call *myopic algorithms*, that arise when the intervals are processed in order of non-decreasing right endpoint. Thus, myopic algorithms are deterministic single-pass algorithms that process the given intervals in order of non-decreasing right endpoint. These algorithms seem to be the simplest algorithms that achieve constant approximation ratios for WJISP. Let us emphasize here that we are primarily interested in the approximation ratios that can be achieved using single-pass algorithms, not in the (better) approximation ratios that can be achieved using arbitrary polynomial-time algorithms.

Analyzing myopic algorithms for WJISP can be seen as studying an on-line problem. We study the quality of the solutions that can be obtained by myopic algorithms. Using competitive analysis we show that for most settings that we investigate the algorithms proposed here are best possible, at least in the case $m = 1$.

In applications where the intervals correspond to time periods, an on-line scenario in which the algorithm receives the intervals in order of non-decreasing right endpoint may appear unnatural. Notice however that for instances where all intervals have the same length, the order of the left endpoints and the order of the right endpoints coincide.

Therefore, the concept of myopic algorithms applies to the "real" on-line problem for such instances.

1.1. Known results

If each job consists of one interval only, WJISP$_m$ reduces to finding a maximum weight m-colorable subgraph in an interval graph. This problem was shown to be solvable in polynomial time by Yannakakis and Gavril [29]. If, in addition, $m = 1$, the problem reduces to finding a maximum weight independent set in an interval graph, for which a polynomial time algorithm was proposed by Frank [12].

The unweighted version of WJISP$_1$, called JISP$_1$, is studied by Spieksma [24]. It is shown that JISP$_1$ is MAX SNP-hard even if every job contains only two intervals and all intervals have the same length. Furthermore, it is shown that the value of the natural LP relaxation of JISP$_1$ is at most two times the value of an integral optimum, and a simple greedy algorithm that achieves approximation ratio 2 is presented.

A problem that is closely related to WJISP is the Time-Constrained Scheduling Problem (TCSP), see also Section 2. In that problem every job has a release time, a length, a deadline, and a weight. Viewed as a special case of WJISP with equal weights per job, every job in an instance of TCSP consists of all intervals of the required length between the release time and the deadline of the job. On the one hand, the structure regarding the overlap of intervals of the same job is very restricted in TCSP, but on the other hand, one must overcome the difficulty of dealing with an infinite number of intervals of a job (in particular, if release times, deadlines, and job lengths can be arbitrary real numbers). Bar-Noy et al. [3] give the following results for TCSP: for the unweighted case of TCSP$_m$, they give an iterative greedy algorithm (that does not belong to the class of single-pass algorithms) with approximation ratio

$$\rho(m) = \frac{(1+1/m)^m}{(1+1/m)^m - 1}.$$

Note that $\rho(1) = 2$ and $\rho(m)$ tends to $e/(e-1) \approx 1.582$ as $m \to \infty$. For TCSP$_m$ with equal weights per job a combinatorial algorithm called ADMISSION is described. ADMISSION consists of applying a greedy procedure m times. The time complexity of ADMISSION is $O(mn^2 \log n)$, where n is the number of jobs. An approximation ratio of $3 + 2\sqrt{2} \approx 5.828$ is proved for ADMISSION. Finally, for TCSP$_m$ with equal weights per job, an LP-based algorithm is given that achieves a ratio of $\rho(m)$ (implying a 2-approximation algorithm for TCSP$_1$).

Some of the results described above can be generalized to WJISP$_m$ with arbitrary weights. In particular, it is not difficult to verify that the LP-based algorithm and its analysis go through for our case, yielding a 2-approximation algorithm for WJISP$_1$ and a $\rho(m)$-approximation algorithm for WJISP$_m$. ADMISSION can be adapted to WJISP with equal weights per job as well, yielding a ratio of $3 + 2\sqrt{2} \approx 5.828$ for WJISP with equal weights per job. Notice that ADMISSION is not a myopic algorithm since it performs m passes over the given intervals. Improving the results of [3], Berman and DasGupta [4] and Bar-Noy et al. [2] proposed combinatorial two-phase algorithms that also achieve ratio 2 for WJISP$_1$ and, by repeated application, ratio $\rho(m)$ for WJISP$_m$. These algorithms do not belong to

the class of single-pass algorithms. For the case of JISP, Chuzhoy et al. [8] improved the known ratios further and showed that for every $\varepsilon > 0$, there is a randomized approximation algorithm with ratio $e/(e-1)+\varepsilon$.

The on-line variant of TCSP is studied by Goldman et al. [15] and Goldwasser [16] in the single-machine case. The weight of a job is equal to its length and the algorithms receive the jobs in order of non-decreasing release times. Preemption is not allowed. In [15], a deterministic algorithm with ratio 2 if all jobs have the same length and a randomized algorithm with expected ratio $O(\log c)$ if the ratio of the longest to the shortest job length is c are presented. Note that "the special case of all jobs having the same length under the arbitrary delay model is of great interest" (quoted from [15]), e.g., for scheduling packets in an ATM switch (where all packets have the same length). In [16], better bounds are derived for the case that the slack of a job is at least proportional to its length.

1.2. Our results

Before describing our results let us first make the observation that WJISP$_m$ can be reduced to WJISP$_1$. This can be done by creating m disjoint copies of the original instance (every job of the new instance consists of all m copies of all its intervals in the original instance); this amounts to *projecting* the m machines onto disjoint parts of the real line. The implication of this observation is that WJISP$_m$ instances are special WJISP$_1$ instances or, in other words, any ρ-approximation algorithm for WJISP$_1$ provides a ρ-approximation algorithm for WJISP$_m$. This partly explains the phenomenon described in [3] that it seems that more machines allow better performance guarantees. Notice, however, that myopic algorithms cannot profit from this reduction: indeed a myopic algorithm processing the resulting instance of WJISP$_1$ would correspond to an algorithm that makes m passes over the intervals in the original instance.

The paper is organized as follows. In Section 2 we describe some applications of WJISP. In Section 3 we propose a myopic algorithm called GREEDY$_\alpha$ that can be implemented to run in $O(n^2)$ time (or in $O(n \log m)$ time if all intervals have the same length). Section 4 shows that with appropriate choices for the parameter α, GREEDY$_\alpha$ achieves the ratios described in Table 1 for WJISP with arbitrary lengths (the ratios for WJISP with equal lengths per job are the same) and in Table 2 for WJISP with equal lengths; each of these ratios is tight. Observe that GREEDY$_\alpha$ has the same ratio for WJISP with arbitrary lengths and equal weights per job as the (non-myopic) algorithm ADMISSION from [3], while having a lower time-complexity. Further, we prove in Section 5 that no myopic algorithm achieves better ratios than the ones described in Tables 1 and 2 under "Lower bound." Our results show that GREEDY$_\alpha$ is optimal or close to optimal in the class of myopic algorithms. In Section 5.4 we prove that even a "randomized myopic" algorithm cannot achieve a ratio better than $\frac{e}{e-1} \approx 1.582$ for JISP. This shows that the use of randomization could at best improve the ratio for JISP from 2 to approximately 1.582. Finally, in Section 6 we state our conclusions.

In order to keep this article at an adequate length, we give detailed proofs only for some of the variants of WJISP in Sections 4 and 5. The proofs for the other results can

Table 1
The results for WJISP with arbitrary lengths

Variant of WJISP	Ratio of GREEDY$_\alpha$	Lower bound
JISP	2, $\forall m \geqslant 1$	2, $\forall m \geqslant 1$
WJISP with		
equal weights per job	$3+2\sqrt{2} \approx 5.828$, $\forall m \geqslant 1$	$3+2\sqrt{2} \approx 5.828$, $m = 1$
arbitrary weights	8, $\forall m \geqslant 1$	≈ 7.103, $m = 1$

Table 2
The results for WJISP with equal lengths

Variant of WJISP	Ratio of GREEDY$_\alpha$	Lower bound
JISP	2, $\forall m \geqslant 1$	2, $\forall m \geqslant 1$
WJISP with		
equal weights per job	5, $m = 1, 2$	5, $m = 1$
arbitrary weights	≈ 6.638, $m = 1, 2$	$3+2\sqrt{2} \approx 5.828$, $m = 1$

be obtained by adapting the proofs accordingly; the basic ideas are the same. Interested readers can find detailed proofs of all our results in the technical report [10].

2. Applications of WJISP

Interval scheduling problems have numerous applications and have been studied intensively (see, for instance, Fischetti et al. [11] and Kroon et al. [19]). In the following, we outline three concrete applications of WJISP.

2.1. Combinatorial auctions

(See Rothkopf et al. [22].) Due to the ongoing sale of frequencies to providers of mobile telecommunications, and due to the ever-increasing popularity of e-commerce, the design of (combinatorial) auctions has become a popular research item. In a combinatorial auction different assets are for sale and bidders are allowed to bid for sets of assets. Given all bids from the bidders, a relevant problem is to decide what bids to accept in order to maximize total revenue (clearly, in general not all bids can be accepted since an asset can be sold at most once). In some cases the assets for sale posses a special structure, for instance, when they can be linearly ordered. A popular example is the case where frequencies are auctioned (indeed frequencies can be ordered by their magnitude), but other examples exist (see [22]). Suppose further that we allow only bids for sets of consecutive assets and allow at most one acceptance for each bidder (see [18,22]). Then the problem of maximizing total revenue for this setting (the interval auction problem) becomes an instance of WJISP$_1$: a bidder is a job, their bids are the intervals and $m = 1$ since one can sell an asset at most once.

Let us now proceed to argue that the *design* of an interval auction can give rise to instances where the on-line interpretation of WJISP becomes relevant. Suppose, as described before, that the assets can be linearly ordered, say $1, 2, \ldots, T$. Moreover, the

auction is designed in such a way that there are T rounds, and in each round t, asset t is added to the set of assets currently for sale (starting with the empty set). Again, as described before, bidders are only allowed to bid for sets of consecutive assets; moreover, in round t a bidder can only make a bid that includes asset t as the largest asset of the current bid. Of course, after each round, the bidders should receive information concerning what bids are currently active and what bids are currently rejected. Now, in principle it is possible to solve the resulting interval auction problem after round t to optimality. However, there are two arguments against such an approach. First, when playing many rounds, it may not be computationally feasible to compute the maximal revenue after each round t due to the intractability of the problem (see [24]). Second, a reasonable stipulation of such an auction would be that a bid that is rejected at some round cannot become "alive" again in later rounds (obviously this may happen when computing an optimal solution after each round). Thus, when designing an interval auction such that there is a round corresponding to each asset, we are faced with the on-line version of WJISP_1.

2.2. Caching

(See Torng [26].) A cache is a small, fast memory that can temporarily store arbitrary memory items in order to allow the CPU to access them faster than in main memory. For the purpose of analyzing cache performance, we view the execution of a program as a sequence of accesses to memory items. When a memory item x is accessed and does not yet reside in the cache, it can be (but does not have to be; we allow cache bypassing) brought into the cache. If x is still in the cache when it is accessed a second time later on, this is called a *cache hit*. With every cache hit the cost for an expensive access to main memory is avoided. We view the period between two accesses to the same item x as an interval on the real line. At the time of an access, at most one interval ends and at most one new interval begins. Selecting an interval i means that x is brought into (or remains in) the cache at the access to x at the beginning of i and stays in the cache until the access to x at the end of i, thus yielding a cache hit. If every memory item can go into an arbitrary cache location (i.e., if we have a fully associative cache) and if the cache has m locations, the problem of maximizing the cache hits can be viewed as an instance of WJISP_m where each job consists of a single interval. In this application, it seems natural to assume that the intervals are unweighted, but there may be other factors that make it more desirable to achieve cache hits for certain accesses, thus giving instances of WJISP_m with equal weights per job. If two consecutive intervals between accesses to x are selected, an additional constraint is that x must reside in the same cache location during both intervals; otherwise, we would have to assume that x can move from one cache location to another at no cost. However, this additional constraint can easily be satisfied in the procedure that colors the interval graph corresponding to the selected intervals.

Due to the high hardware cost for fully associative caches, t-way set associative caches are often used instead in practice. Here, a cache with k locations is partitioned into t direct mapped caches, each of size k/t. A memory item x is mapped to a position $p(x)$, $1 \leqslant p(x) \leqslant k/t$, and can be stored only in location $p(x)$ in each of the t direct mapped caches. Conceptually, this can be viewed as partitioning the cache into k/t sub-caches, each of size t, such that each sub-cache is fully associative and such that every memory item

can go in only one of the k/t sub-caches. The problem of maximizing cache hits in a t-way set associative cache with k locations can thus be solved by solving the subproblems for each of the k/t sub-caches independently. In terms of WJISP, this amounts to k/t disjoint instances of WJISP_t that can be combined into a single instance by projecting them onto disjoint parts of the real line.

Finally, consider the case that there can be more general restrictions on the cache locations available to a memory item (e.g., as in t-way skewed associative caches [23]). For every memory item x, there is a number of admissible locations in the cache where the item can be stored. The problem of maximizing cache hits can then be modeled as an instance of WJISP_1 as follows: project the timelines of all cache locations onto disjoint parts of the real line and add, for every period between consecutive accesses to x, an interval in those parts of the real line that correspond to cache locations that are admissible for x (all intervals for this period belong to one job). However, it should be noted that the constraint that items cannot move within the cache is ignored by this approach.

2.3. Time-constrained scheduling

(See Bar-Noy et al. [3].) Consider the following scheduling problem: we are given m machines (identical or unrelated) and n tasks, and each task has a release time, a deadline, a processing time (that can depend, in the case of unrelated machines, on the machine on which the task is executed), and a weight. We want to select a subset of the given tasks and schedule them on the machines non-preemptively such that every selected task is scheduled no earlier than its release time and finishes no later than its deadline. The goal is to maximize the sum of the weights of the scheduled tasks.

We can view this scheduling problem for m identical machines as an instance of WJISP_m with equal weights per job (tasks correspond to jobs, and every possible execution of a task corresponds to an interval) and the problem for m unrelated machines as an instance of WJISP_1 (by projecting the timelines of all m machines onto different parts of the real line). Notice that the intervals in instances of WJISP arising from this application display a special structure. If all release times, deadlines, and processing times are integers that are bounded by a polynomial in the size of the input (i.e., if we have *polynomially bounded integral input*), the resulting instances of WJISP_m and WJISP_1 have size polynomial in the original instance (only intervals with integral starting times must be considered).

Further applications of WJISP can be found in printed circuit board manufacturing (see Crama et al. [9]) and in molecular biology (see Veeramachaneni et al. [27]). Other applications are mentioned in Chuzhoy et al. [8].

3. Algorithm GREEDY_α

We propose a myopic algorithm called GREEDY_α, shown in Fig. 1, as an approximation algorithm for WJISP. It has a parameter α that can take (meaningful) values in the range $[0, 1]$. GREEDY_α considers the intervals in order of non-decreasing right endpoint. It maintains a set S of currently selected intervals. When it processes an interval i, it com-

Algorithm GREEDY$_\alpha$

$S = \emptyset$; {set of currently accepted intervals}
for all intervals, in order of non-decreasing right endpoint **do**
  $i =$ current interval;
  $C_i =$ minimum-weight subset of S such that $(S \setminus C_i) \cup \{i\}$ is feasible;
  if $w(C_i) \leqslant \alpha w(i)$ **then**
    $S = (S \setminus C_i) \cup \{i\}$;
  fi;
od;
return S;

Fig. 1. Algorithm GREEDY$_\alpha$.

putes a set $C_i \subseteq S$ such that i could be selected after preempting the intervals in C_i and such that C_i has minimum weight among all such sets. C_i is called the *cheapest conflict set* for i. The algorithm selects i only if $w(C_i) \leqslant \alpha w(i)$, i.e., if the total weight of selected intervals increases by at least $(1-\alpha)w(i)$ if i is selected and the intervals from C_i are preempted.

Let us briefly compare GREEDY$_\alpha$ to the algorithm ADMISSION of [3], which was found independently of our work. The basic spirit of the algorithms is similar: For the case of WJISP$_1$ with equal weights per job, the two algorithms are essentially identical, and the parameters α of GREEDY$_\alpha$ and β of ADMISSION are related by $\alpha = 1/\beta$. For the case $m > 1$, however, ADMISSION passes through the intervals m times (once for each machine) [3], while our algorithm GREEDY$_\alpha$ is a single-pass algorithm for any value of m. Furthermore, GREEDY$_\alpha$ can deal with the case of arbitrary weights, while ADMISSION is specified only for the case of equal weights per job.

We are interested in an efficient implementation of GREEDY$_\alpha$. In Section 3.1, we show how the cheapest conflict set C_i can be determined efficiently, which gives rise to a total running time of $O(n^2)$ of GREEDY$_\alpha$. As a byproduct of this subsection we show how knowing that an interval graph is m-colorable gives you an $O(n)$ algorithm for obtaining a maximum-weight $(m-1)$-colorable subgraph as compared to $O(mS(n))$ in the general case [7], where $S(n)$ denotes the running time for any algorithm for finding a shortest path in a directed graph with $O(n)$ arcs and positive arc weights. (With an efficient implementation of Dijkstra's algorithm, for example, $S(n)$ can be taken as $O(n \log n)$. There are algorithms that are asymptotically faster (see, e.g., [13,21] and further references given in [25]); however, these algorithms seem of theoretical interest only and do not achieve a linear running-time. The only linear-time shortest paths algorithm known so far is due to Thorup [25]; it works for undirected graphs.) In Section 4, we analyze the approximation ratio achieved by GREEDY$_\alpha$.

3.1. Determining the cheapest conflict set

Let i be the interval that is currently processed by GREEDY$_\alpha$. In the special case of $m = 1$, the set C_i is simply the set of all intervals in S that intersect i and, possibly, an interval $i' \in S$ that belongs to the same job as i. Therefore, C_i can be determined easily in this case. In the following, we show how to deal with the more complicated case of arbitrary m.

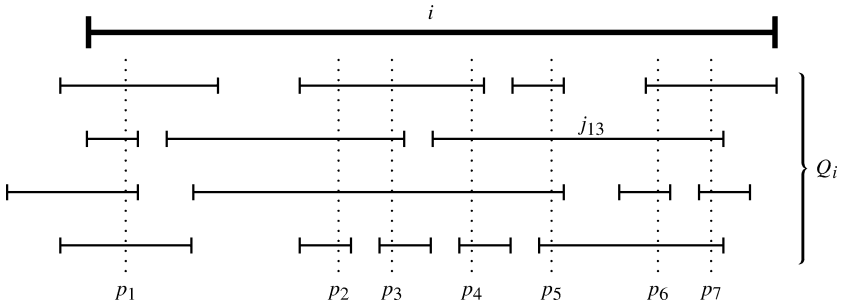

Fig. 2. Current interval i and intersecting intervals in S.

If S contains an interval i' that belongs to the same job as i, it is clear that C_i must contain i'. In that case, let $C_i' = C_i \setminus \{i'\}$, otherwise let $C_i' = C_i$. Let $Q_i \subseteq S$ be the subset of currently selected intervals that intersect the interval i and that do not belong to the same job as i. See Fig. 2. Obviously, C_i' is a minimum-weight subset of Q_i such that $Q_i \setminus C_i'$ is $(m-1)$-colorable. Hence, the set $Q_i \setminus C_i'$ is a maximum-weight $(m-1)$-colorable subset of Q_i. Thus, the problem of determining the cheapest conflict set is equivalent to the problem of finding a maximum weight $(m-1)$-colorable subgraph in an m-colorable interval graph. Therefore, $Q_i \setminus C_i'$ and C_i' could be determined in polynomial time using an algorithm that solves the maximum-weight k-colorable subgraph problem in interval graphs for arbitrary values of k. However, since our problem is more specific (we know that Q_i is m-colorable), we can compute the set C_i' in time $O(n)$ using a dynamic programming approach, as witnessed by the following theorem.

Theorem 1. *The cheapest conflict set can be computed in $O(n)$ time.*

Proof. For every interval $j \in Q_i$, consider a point p just before the right endpoint of j. If p is contained in m intervals of Q_i, at least one of these intervals must be in C_i'. Let $p_1, \ldots, p_r$ be all such points (i.e., points just before the right endpoint of an interval in Q_i) that are contained in m intervals of Q_i. It is clear that C_i' is a minimum-weight subset of Q_i such that every point p_h, $1 \leqslant h \leqslant r$, is covered by that subset. (We say that a set of intervals *covers* a point if the point is contained in at least one interval of that set.)

Let $j_1, j_2, \ldots, j_s$ denote the intervals in Q_i in order of non-decreasing right endpoint. For every interval $j_\ell \in Q_i$, let *first*(j_ℓ) and *last*(j_ℓ) be the smallest (respectively largest) index of a point p_h that is contained in j_ℓ. (If j_ℓ does not contain any point p_h, *first*(j_ℓ) and *last*(j_ℓ) are undefined.) In Fig. 2, we have *first*$(j_{13}) = 4$ and *last*$(j_{13}) = 7$, for example. The points $p_1, \ldots, p_r$, the intervals $j_1, \ldots, j_s$, and the tables *first* and *last* can all be constructed in time $O(n)$. We use the dynamic programming procedure shown in Fig. 3 to compute values $C(h)$, $1 \leqslant h \leqslant r$.

Claim 1. *After ℓ iterations of the outer for-loop, it holds for $1 \leqslant h \leqslant r$ that $C(h)$ is the weight of a minimum-weight subset of $\{j_1, \ldots, j_\ell\}$ that covers $p_1, \ldots, p_h$ (or ∞ if no such subset exists).*

```
C(0) = 0;
for h = 1 to r do C(h) = ∞; od;
for ℓ = 1 to s do
    if first(j_ℓ) is not undefined then
        for h = first(j_ℓ) to last(j_ℓ) do
            C(h) = min{C(h), C(first(j_ℓ) − 1) + w(j_ℓ)};
        od;
    fi;
od;
```

Fig. 3. Dynamic programming algorithm.

Obviously, Claim 1 implies that $C(r) = w(C'_i)$ at the end of the execution of the procedure, and additional bookkeeping will allow to construct C'_i at no extra cost. Thus, the correctness of the algorithm follows from Claim 1, which can be proved easily by induction on ℓ. The running-time of the procedure sketched in Fig. 3 is $O(s+rm) = O(nm)$, because the body of the inner for-loop is executed exactly rm times. (For each of the r points p_h, the body of the inner for-loop is executed only for the m intervals containing p_h.)

Let us now describe an implementation of the algorithm in Fig. 3 that runs in $O(n)$ time. To begin with, notice that the costs $C(h)$, $0 \leqslant h \leqslant r$, are non-decreasing with h at any stage of the algorithm. We will maintain disjoint consecutive subsets of $\{0, 1, \ldots, r\}$ with the property that for each subset X, the value $C(h)$ is the same for all $h \in X$. Elements with $C(h) = \infty$ are kept in singleton subsets. For maintaining the subsets, we use the data structure due to Gabow and Tarjan [14]. We may assume that this data structure provides the following operations:

INITIALIZE(r): initialize the data structure with $r+1$ singleton sets $\{0\}, \{1\}, \ldots, \{r\}$.
FIND(x): return the first and last element of the subset that currently contains x, i.e., if x is currently in the set $\{a, a+1, \ldots, b\}$, then return the pair (a, b).
UNION(x, y): merge the sets containing x and y. Precondition: x and y are in different sets, x is the largest element in the set containing x, and y is the smallest element in the set containing y.

The precondition of the UNION operation ensures that the structure of all potential UNION operations is a chain and thus the data structure of [14] is applicable (that data structure requires that the potential UNION operations form a tree). The total running-time for the initialization of the data structure and a sequence consisting of $O(s)$ operations FIND(x) and up to r operations UNION(x, y) is then $O(r+s)$.

For a subset X maintained by the data structure, we store the common value $C(h)$ for all $h \in X$ with the largest element in X (the second component of the pair returned by FIND(h) for any $h \in X$). For an element h that is not the largest element of its subset, the value of the variable $C(h)$ can be arbitrary.

The pseudo-code of the resulting implementation is shown in Fig. 4. For each interval j_ℓ, the algorithm first determines the value $C(\mathit{first}(j_\ell) - 1)$ by executing $(a, b) =$ FIND($\mathit{first}(j_\ell) - 1$) and then accessing $C(b)$. This allows to compute the value $w = C(b) + w(j_\ell)$, which is the cost of the new candidate set that covers all p_i with $i \leqslant \mathit{last}(j_\ell)$ and has j_ℓ as its rightmost interval. If the elements in the set containing $\mathit{last}(j_\ell)$ have a cost

```
INITIALIZE(r);
C(0) = 0;
for h = 1 to r do C(h) = ∞; od;
for ℓ = 1 to s do
    if first(j_ℓ) is not undefined then
        (a, b) = FIND(first(j_ℓ) − 1);
        w = C(b) + w(j_ℓ);
        (c, d) = FIND(last(j_ℓ)); /* will give d = last(j_ℓ) */
        if C(d) > w then
            C(d) = w;
            (e, f) = FIND(c − 1); /* will give f = c − 1 */
            while C(f) > w do
                UNION(f, c);
                c = e;
                (e, f) = FIND(c − 1); /* will give f = c − 1 */
            od;
        fi;
    fi;
od;
```

Fig. 4. Linear-time implementation of the dynamic programming algorithm.

larger than w, their cost is updated to w by setting $C(d) = w$. (Note that the set containing $last(j_\ell)$ must have $last(j_\ell)$ as its largest element, because all elements $h > last(j_\ell)$ still have cost $C(h) = \infty$.) In this case, the algorithm then checks repeatedly whether the elements in the set that is just before the set containing $last(j_\ell)$ have a cost larger than w and, if so, merges that set with the set containing $last(j_\ell)$. The effect of each such UNION operation is that the elements h in the set with smaller elements implicitly receive the same value $C(h) = w$ as the elements in the set containing $last(j_\ell)$.

It is not difficult to see that this is a correct implementation of the dynamic programming algorithm of Fig. 3. It remains to analyze the running-time. For each interval j_ℓ, the number of FIND operations executed in the body of the loop is $3 + k$, where k is the number of iterations of the inner while loop. Since each iteration of the inner while loop executes a UNION operation and there can be at most r such operations, the total number of FIND operations can be bounded by $3s + r$. Hence, the total running-time of the algorithm is $O(s + r) = O(n)$. Moreover, the cheapest conflict set itself can again be computed easily in the same running-time by storing with each value $C(h)$ also the index of the rightmost interval of the solution covering $p_1, \ldots, p_h$ that has cost $C(h)$. □

Theorem 1 leads to the following corollaries.

Corollary 2. *A maximum weight $(m-1)$-colorable subgraph in an m-colorable interval graph, given by the sorted list of interval endpoints, can be obtained in $O(n)$ time.*

Corollary 3. *GREEDY$_\alpha$ runs in $O(n^2)$ time.*

For WJISP with equal lengths, computing the cheapest conflict set is much easier, since at most m intervals in S can overlap the current interval i. The cheapest conflict set C_i contains the interval in S that belongs to the same job as i (if such an interval exists)

as well as the cheapest interval overlapping i (if there are m intervals in S that overlap i and belong to a different job). We maintain a balanced search tree T such that, if the current interval has left endpoint p, T stores all intervals in S overlapping p, sorted by their weights. Determining the cheapest conflict set and updating T takes time $O(\log m)$, giving a total running time of $O(n \log m)$ for GREEDY$_\alpha$ in the case of WJISP with equal lengths.

4. Analysis of approximation ratio

In this section, we give tight bounds on the approximation ratio of GREEDY$_\alpha$ for the different variants of WJISP. The section is divided into two parts: in Section 4.1 we investigate the case where intervals have arbitrary lengths, and in Section 4.2 we deal with the case where all intervals have the same length. Each subsection deals with the three possibilities concerning the weights of the intervals: equal weights, equal weights per job, and arbitrary weights.

Let us now introduce some notation. We use A to denote the set of intervals that is returned by GREEDY$_\alpha$, we use T for the set of intervals that were selected at least at some time by GREEDY$_\alpha$, and we use OPT for some set of intervals that constitutes an optimal solution. Their values are referred to as $w(A)$, $w(T)$, and $w(OPT)$, respectively. Further, as mentioned before, the set S is the set of selected intervals at some point during the execution of the algorithm. The basic idea of the analysis is to charge the weight of the intervals in an optimal solution to the intervals selected by GREEDY$_\alpha$ (the set T) and next to derive bounds on the amount of charge received by intervals in A. An inequality that is fundamental in the analysis is

$$w(A) \geqslant (1-\alpha)w(T). \tag{1}$$

This inequality holds because GREEDY$_\alpha$ selects a new interval i only if the total weight of currently selected intervals increases by at least $(1-\alpha)w(i)$. Thus, first we bound $w(OPT)$ in terms of $w(T)$ and, using (1), in terms of $w(A)$.

4.1. WJISP with arbitrary lengths

Here, we analyze the approximation ratio of GREEDY$_\alpha$ in the case that the given intervals have arbitrary lengths. For the case of arbitrary weights, we have the following theorem.

Theorem 2 (arbitrary lengths). *For WJISP$_m$ with arbitrary weights, GREEDY$_\alpha$ achieves approximation ratio* $\frac{2}{\alpha(1-\alpha)}$.

Proof. Consider an interval $i \in OPT$. If $i \in T$, we charge $w(i)$ to i. If $i \in OPT \setminus T$, consider the instant when GREEDY$_\alpha$ processed interval i. Let C_i denote the minimum-weight set of intervals whose removal from S would have allowed to accept i. If S contains an interval from the same job as i, denote that interval by i'; otherwise, let i' be an imaginary interval with zero weight (just to simplify the formulas).

Let Q_i denote the set of all intervals in S that intersect i and that do not belong to the same job as i. As S is feasible, Q_i can be partitioned into m sets $Q_{i1}, \ldots, Q_{im}$ of intervals such that the intervals in each set $Q_{i\ell}$ are pairwise disjoint. Note that

$$w(i') + w(Q_{i\ell}) > \alpha w(i) \quad \text{for } 1 \leqslant \ell \leqslant m, \tag{2}$$

because of the definition of C_i and because GREEDY$_\alpha$ did not accept i.

We charge $\min\{w(i), (1/\alpha)w(i')\}$ to i'. If the remaining weight $w(i) - \min\{w(i), (1/\alpha)w(i')\}$ is positive, we divide it into m equal parts and distribute each part among the intervals in one set $Q_{i\ell}$ such that an interval $j \in Q_{i\ell}$ is charged

$$\frac{w(i) - \min\{w(i), \frac{1}{\alpha}w(i')\}}{m} \cdot \frac{w(j)}{w(Q_{i\ell})}.$$

(2) implies that every interval $j \in Q_i$ is charged by i for at most $(1/\alpha m)w(j)$ in this way. Altogether, every interval $j \in T$ can receive charge at most $(2/\alpha)w(j)$ (charge at most $(1/\alpha)w(j)$ from an interval belonging to the same job and charge at most $(1/\alpha)w(j)$ from intervals in *OPT* that overlap the right endpoint of j), implying that

$$w(OPT) \leqslant \frac{2}{\alpha}w(T) \leqslant \frac{2}{\alpha(1-\alpha)}w(A). \qquad \square$$

It is not difficult to see that the ratio in Theorem 2 is tight (even in the case of equal lengths per job). Hence, we get the following corollary.

Corollary 4 (arbitrary lengths). *For WJISP$_m$ with arbitrary weights, GREEDY$_\alpha$ performs best for $\alpha = 1/2$, in which case it achieves approximation ratio* 8.

In the cases of unit weights and of equal weights per job, the proof technique of Theorem 2 can be adapted to obtain the following results (see [10] for details).

Theorem 3 (arbitrary lengths). *For JISP$_m$, GREEDY$_\alpha$ achieves approximation ratio* 2 *for any α in the range* $[0, 1)$.

Theorem 4 (arbitrary lengths). *For WJISP$_m$ with equal weights per job, GREEDY$_\alpha$ achieves approximation ratio* $(1+\alpha)/(\alpha(1-\alpha))$.

Again, it is not difficult to construct examples showing that the ratios in Theorems 3 and 4 are tight, even in the case of equal lengths per job (see [10] for details).

Corollary 5 (arbitrary lengths). *For WJISP$_m$ with equal weights per job, GREEDY$_\alpha$ performs best for $\alpha = \sqrt{2} - 1 \approx 0.414$, in which case it achieves approximation ratio* $3 + 2\sqrt{2} \approx 5.828$.

Note that this result is consistent with the result of Bar-Noy et al. [3], who proved independently that ADMISSION performs best for TCSP with parameter $\beta = 1 + \sqrt{2}$ and achieves ratio $3 + 2\sqrt{2}$ in this case (their parameter β corresponds to $1/\alpha$ in our setting).

4.2. WJISP with equal lengths

Now we consider WJISP in the case where all intervals of all jobs have the same length. In applications where the intervals correspond to time intervals, myopic algorithms are "real" on-line algorithms in this case, as the order of left endpoints and the order of right endpoints coincide.

For JISP with equal lengths, the approximation ratio of GREEDY$_\alpha$ is 2, the same as in the case of arbitrary lengths (Theorem 3), and this is again tight. Therefore, we need to consider here only WJISP with arbitrary weights (Section 4.2.1) and WJISP with equal weights per job (Section 4.2.2). It turns out that for these cases GREEDY$_\alpha$ achieves better ratios for equal lengths than for arbitrary lengths. The analysis, however, gets a bit more complicated, and we do not have tight results for all values of m.

4.2.1. Arbitrary weights

We consider WJISP with equal lengths and arbitrary weights. First, we consider the case $m = 1$.

Theorem 5 (equal lengths). *For WJISP$_1$ with arbitrary weights, GREEDY$_\alpha$ achieves approximation ratio* $(2+\alpha)/(\alpha(1-\alpha^2))$.

Proof. We distinguish *job charge* and *overlap charge*. Consider an interval $i \in OPT$ and the instant when GREEDY$_\alpha$ processed that interval. If $i \in T$, charge $w(i)$ to i and call this charge *overlap charge*. Now assume that $i \notin T$. Let Q_i be the set of intervals in S that are in conflict with i when GREEDY$_\alpha$ processes i. Note that Q_i contains at most one interval intersecting i (because all intervals have the same length) and at most one additional interval belonging to the same job as i. Therefore, we have $|Q_i| \leqslant 2$. Each interval $j \in Q_i$ is charged

$$\frac{w(j)}{w(Q_i)} \cdot w(i).$$

As GREEDY$_\alpha$ did not select i, we have $w(Q_i) > \alpha w(i)$, so each interval $j \in Q_i$ receives charge at most $w(j)/\alpha$ from i. If j intersects i, we call the charge that j receives from i *overlap charge*, otherwise *job charge*.

In order to facilitate a tight analysis, we redistribute some of the job charge. Consider an interval $j \in T$ that receives job charge from an interval $i \in OPT$ belonging to the same job. If $j \in A$, we do nothing. If $j \in T \setminus A$, this means that j was preempted by GREEDY$_\alpha$ at some later time in favor of an interval k belonging to the same job as j. Observe that k cannot have received any job charge, as the interval $i \in OPT$ that belongs to the same job as j and k was processed by GREEDY$_\alpha$ before k. Now we redistribute the total charge that j and k have received, which is bounded from above by $\frac{2}{\alpha}w(j) + \frac{1}{\alpha}w(k)$, proportionally among these two intervals. Since $w(k) \geqslant w(j)/\alpha$, we can bound the resulting charge c_j for interval j as follows:

$$c_j \leqslant \frac{w(j)}{w(j)+w(k)}\left(\frac{2}{\alpha}w(j) + \frac{1}{\alpha}w(k)\right) = w(j)\left(\frac{1}{\alpha} + \frac{\frac{1}{\alpha}w(j)}{w(j)+w(k)}\right)$$

$$\leqslant w(j)\left(\frac{1}{\alpha}+\frac{1}{\alpha\left(1+\frac{1}{\alpha}\right)}\right)=w(j)\left(\frac{1}{\alpha}+\frac{1}{\alpha+1}\right).$$

In the same way, we obtain that the resulting charge c_k for interval k is bounded by

$$c_k \leqslant w(k)\left(\frac{1}{\alpha}+\frac{1}{\alpha+1}\right).$$

Putting everything together, we get that every interval $j \in A$ receives charge at most $\frac{2}{\alpha}w(j)$, while every interval $k \in T \setminus A$ receives charge at most

$$\left(\frac{1}{\alpha}+\frac{1}{\alpha+1}\right)w(k).$$

We obtain:

$$\begin{aligned} w(OPT) &\leqslant w(A)\cdot\frac{2}{\alpha}+w(T\setminus A)\left(\frac{1}{\alpha}+\frac{1}{\alpha+1}\right)\\ &= w(T)\left(\frac{1}{\alpha}+\frac{1}{\alpha+1}\right)+w(A)\left(\frac{1}{\alpha}-\frac{1}{\alpha+1}\right)\\ &\leqslant w(A)\frac{1}{1-\alpha}\left(\frac{1}{\alpha}+\frac{1}{\alpha+1}\right)+w(A)\left(\frac{1}{\alpha}-\frac{1}{\alpha+1}\right)\\ &= w(A)\frac{2+\alpha}{\alpha(1-\alpha^2)}. \qquad \square \end{aligned}$$

Furthermore, for every $m \geqslant 1$, one can construct examples showing that the approximation ratio of GREEDY$_\alpha$ for WJISP$_m$ with arbitrary weights is not better than $(2+\alpha)/(\alpha(1-\alpha^2))$ (see [10]). Therefore, Theorem 5 is tight for WJISP$_1$.

Corollary 6 (equal lengths). *For WJISP$_1$ with arbitrary weights, GREEDY$_\alpha$ performs best for $\alpha = 2\cos(2\pi/9)-1 \approx 0.5321$, in which case it achieves approximation ratio*

$$\frac{1+2\cos\left(\frac{2}{9}\pi\right)}{7+6\cos\left(\frac{4}{9}\pi\right)-10\cos\left(\frac{2}{9}\pi\right)} \approx 6.638.$$

By a very detailed and extensive analysis, we can extend Theorem 5 to the case $m = 2$, giving the following theorem (the long proof can be found in [10]).

Theorem 6 (equal lengths). *For WJISP$_2$ with arbitrary weights, GREEDY$_\alpha$ achieves approximation ratio $(2+\alpha)/(\alpha(1-\alpha^2))$ and thus performs best for $\alpha = 2\cos(2\pi/9)-1 \approx 0.5321$, in which case it achieves approximation ratio*

$$\frac{1+2\cos\left(\frac{2}{9}\pi\right)}{7+6\cos\left(\frac{4}{9}\pi\right)-10\cos\left(\frac{2}{9}\pi\right)} \approx 6.638.$$

Now we consider the case of arbitrary $m > 1$.

Theorem 7 (equal lengths). *For WJISP$_m$, $m \geqslant 2$, with arbitrary weights, the approximation ratio of GREEDY$_\alpha$ is at most $(2m-\alpha)/(m\alpha(1-\alpha))$.*

Proof. Charge the weight of intervals in OPT to intervals in T as in the proof of Theorem 2. We know that an interval $j \in T$ receives charge at most $\frac{2}{\alpha}w(j)$. Assume that an interval $j \in T \setminus A$ receives a charge of more than

$$\left(\frac{m-1}{m}+1\right)\frac{1}{\alpha}w(j).$$

This implies that j receives charge from m intervals overlapping the right endpoint of j (each of these charges at most $\frac{1}{\alpha m}w(j)$) and from an interval $i \in OPT$ that belongs to the same job as j, that lies strictly to the right of j, and that is disjoint from j. As $j \in T \setminus A$ and j was in S when GREEDY$_\alpha$ processed interval i, this means that GREEDY$_\alpha$ selected an interval k that was processed after i and that belongs to the same job as j. Note that we must have $w(k) \geqslant w(j)/\alpha$. Distributing the total charge received by j and k proportionally among these two intervals, we can bound the resulting charge c_j and c_k for interval j and interval k, respectively, as in the proof of Theorem 5:

$$c_j \leqslant w(j)\left(\frac{1}{\alpha}+\frac{1}{\alpha+1}\right), \qquad c_k \leqslant w(k)\left(\frac{1}{\alpha}+\frac{1}{\alpha+1}\right).$$

So every interval $j \in T \setminus A$ that is not involved in the redistribution of charge receives charge at most

$$\left(\frac{m-1}{m}+1\right)\frac{1}{\alpha}w(j),$$

while an interval $j \in T \setminus A$ that is involved in redistribution receives charge at most

$$w(j)\left(\frac{1}{\alpha}+\frac{1}{\alpha+1}\right).$$

For $m \geqslant 2$ the former bound is larger, so that we can bound $w(OPT)$ as follows:

$$\begin{aligned} w(OPT) &\leqslant w(A)\cdot\frac{2}{\alpha}+w(T\setminus A)\frac{2m-1}{m\alpha} = w(A)\cdot\frac{1}{m\alpha}+w(T)\frac{2m-1}{m\alpha} \\ &\leqslant w(A)\cdot\frac{1}{m\alpha}+w(A)\frac{2m-1}{m\alpha(1-\alpha)} = w(A)\cdot\frac{2m-\alpha}{m\alpha(1-\alpha)}. \end{aligned}$$ □

We do not now whether our analysis of Theorem 7 is tight. Nevertheless, we can determine the value of α that optimizes the obtained bound and get the following corollary.

Corollary 7 (equal lengths). *For WJISP$_m$, $m \geqslant 3$, with arbitrary weights, our analysis of GREEDY$_\alpha$ gives the best ratio for $\alpha = 2m-\sqrt{4m^2-2m}$, in which case the bound on the approximation ratio of GREEDY$_\alpha$ is*

$$\frac{4m-1+2\sqrt{4m^2-2m}}{m}.$$

Some bounds on the ratio of GREEDY$_\alpha$ resulting from Corollary 6, Theorem 6, and Corollary 7 are as follows:

m	1	2	3	4	5	6	7	8	9	10
ratio	6.638	6.638	7.318	7.491	7.594	7.663	7.712	7.748	7.776	7.799

Note that for every $m \geqslant 1$ these bounds are better than the ratio 8 that GREEDY$_\alpha$ achieves for WJISP$_m$ with arbitrary weights and arbitrary lengths. Furthermore, our bounds for the equal-lengths case converge to 8 as m goes to infinity.

4.2.2. Equal weights per job

The analysis for the case of WJISP$_m$ with equal lengths and equal weights per job is similar to the previous section. The key observation that leads to improved results is that an interval $k \in T$ can receive job charge only if $k \in A$ and that the job charge received by k is either 0 or $w(k)$. No redistribution of charge is necessary. For $m = 1$, this yields the following theorem.

Theorem 8 (equal lengths). *For WJISP$_1$ with equal weights per job, GREEDY$_\alpha$ achieves approximation ratio* $(1 + \alpha(1-\alpha))/(\alpha(1-\alpha))$.

It is not difficult to construct examples showing that for every $m \geqslant 1$, the approximation ratio of GREEDY$_\alpha$ for WJISP$_m$ with equal weights per job and equal lengths is not better than $(1 + \alpha(1-\alpha))/(\alpha(1-\alpha))$ (see [10] for details). This shows that Theorem 8 is tight.

Corollary 8 (equal lengths). *For WJISP$_1$ with equal weights per job, GREEDY$_\alpha$ performs best for* $\alpha = 1/2$, *in which case it achieves approximation ratio* 5.

We will prove in Theorem 11 that no deterministic myopic algorithm for WJISP$_1$ with equal weights per job and arbitrary lengths can have approximation ratio better than approximately 5.828. This shows that the case of WJISP$_1$ with equal lengths is provably easier to approximate for myopic algorithms than the case of arbitrary lengths.

Now we consider the case $m > 1$. By adapting the proof of Theorem 7, the following theorem is obtained. The main observation is that an interval $j \in T$ that receives a charge of more than

$$w(j) \cdot \max\left\{\frac{1}{\alpha},\ 1 + \frac{m-1}{m\alpha}\right\}$$

must receive charge from an interval $i \in OPT$ that belongs to the same job, is disjoint from j, and lies strictly to the right of j (which in turn implies that $j \in A$). Redistribution of charges is not necessary.

Theorem 9 (equal lengths). *For WJISP$_m$, $m \geqslant 2$, with equal weights per job, the approximation ratio of GREEDY$_\alpha$ is not worse than*

$$\frac{1+\alpha(1-\alpha)}{\alpha(1-\alpha)} \quad \text{for } \alpha \leqslant \frac{1}{m} \quad \text{and} \quad \frac{m\alpha + m - \alpha}{m\alpha(1-\alpha)} \quad \text{for } \alpha \geqslant \frac{1}{m}.$$

We do not know whether our analysis of Theorem 9 is tight. Nevertheless, we can determine the value of α that optimizes the obtained bound and get the following corollary.

Corollary 9 (equal lengths). *For WJISP$_m$ with equal weights per job, the following bounds for the ratio of GREEDY$_\alpha$ are the best bounds obtainable from our analysis*:

- $m = 2$: *ratio* 5, *obtained for* $\alpha = 1/2$ (*derived from Theorem* 9).
- $3 \leqslant m \leqslant 67$: *ratio*

$$\frac{5m^3 - 3m^2 - 6m + 2 + \sqrt{5m^2 - 4m}(3m^2 + m - 2)}{2m(m^2 - 1)}$$

$$\text{for } \alpha = \frac{3m - \sqrt{5m^2 - 4m}}{2m + 2}$$

  (*derived from Theorem* 9).
- $m \geqslant 68$: *ratio* $3 + 2\sqrt{2}$ *for* $\alpha = \sqrt{2} - 1$ (*derived from Corollary* 5).

Some bounds on the ratio of GREEDY$_\alpha$ resulting from Corollaries 8 and 9 are as follows:

m	1	2	3	4	5	6	7	8	9	10
ratio	5	5	5.268	5.417	5.505	5.564	5.606	5.637	5.661	5.681

Note that our bounds for the equal-lengths case are better than the ratio $3 + 2\sqrt{2} \approx 5.828$ that GREEDY$_\alpha$ achieves for WJISP$_m$ with arbitrary lengths and equal weights per job for $1 \leqslant m \leqslant 67$.

5. Competitive lower bounds

In this section, we are interested in lower bounds on the best approximation ratio that can be achieved by any myopic algorithm for WJISP, and we use competitive analysis [5] to answer this question. We phrase our arguments in terms of an adversary who constructs worst-case instances for a myopic algorithm incrementally, depending on previous decisions made by the algorithm. In our illustrations, intervals belonging to the same job are always drawn in the same row. In Section 5.1 we deal with the unweighted case, Section 5.2 treats the case of equal weights per job, Section 5.3 investigates the case of arbitrary weights, and Section 5.4 gives a lower bound for randomized myopic algorithms for JISP.

5.1. The unweighted version of WJISP (JISP)

In the case of equal weights, it is easy to show that no myopic algorithm for WJISP$_m$ can achieve approximation ratio better than 2.

Theorem 10. *No myopic algorithm for JISP$_m$ can achieve an approximation ratio better than* 2*, even if all intervals have equal length.*

Proof. Initially, the adversary presents a set Q of $2m$ intervals that belong to different jobs and that have the same right endpoint p. Let S be the set of at most m intervals that are selected by the algorithm at this time. Let S' be a subset of Q that contains all intervals in S and that has cardinality m. For every interval in S', the adversary presents an interval that belongs to the same job and that is to the right of p. All these new intervals have the same right endpoint. The optimal solution contains $2m$ intervals, while the algorithm accepts at most m intervals. Notice that the lengths of all intervals in this construction can be set equal. □

5.2. WJISP$_1$ with equal weights per job

Consider WJISP$_1$ with equal weights per job. In the following, we prove tight bounds on the best approximation ratio that can be achieved by any (deterministic) myopic algorithm for the case of arbitrary lengths or equal lengths per job and for the case of equal lengths. We view the construction of worst-case examples for a myopic algorithm as a game played by the algorithm with an adversary. In a move, the adversary presents a new interval, and in response the algorithm accepts this interval or rejects.

We begin with the case of equal lengths per job. For every $\varepsilon > 0$ that is sufficiently small (e.g., take $1/8 \geqslant \varepsilon > 0$), the adversary has a strategy that forces the algorithm to create a solution of weight $(3 + 2\sqrt{2})/(1 + \varepsilon)^2$ times smaller than the optimum.

The strategy of the adversary has the following properties (some of them hold after the first two moves).

- The right endpoints of intervals presented by the adversary are strictly increasing.
- The tentative solution of the algorithm consists of exactly one interval called c (the current one).
- A set of intervals, P, forms a part of the eventual solution of the adversary, all elements of P have right endpoint smaller than the right endpoint of c and none belongs to the same job as c.
- Whenever the algorithm accepts a new interval, the ratio $w(P)/w(c)$ increases at least by a $1 + \varepsilon$ factor.
- If the algorithm rejects intervals sufficiently many times, the adversary wins by exhibiting a solution R such that $(1 + \varepsilon)^2 w(R)/w(c) \geqslant 3 + 2\sqrt{2}$.

The strategy of the adversary can be described as a set of prescriptions how to move in different states.

State 0: *initial*. In the first move the adversary presents interval i_0 with weight 1, and the algorithm has to accept i_0. In the second move, the adversary presents interval i_1 that belongs to a new job, intersects i_0 and has a larger right endpoint; $w(i_1) = 3 + 2\sqrt{2}$. If the algorithm does not accept i_1, the adversary exhibits $R = \{i_1\}$ and wins. Otherwise, we have $c = i_1$ and $P = \{i_0\}$.

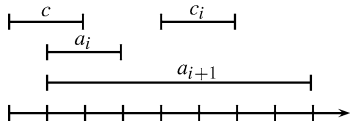

Fig. 5. Illustration of intervals a_i, c_i, and a_{i+1} played by the adverser.

The remaining states depend on the algorithm's response in the previous move. Intervals of the form a_i belong to new jobs, have weight $w(a_i) = w(c)(1+\varepsilon)^i$, and every a_i has the same left endpoint such that a_i intersects c but none of the intervals in P. An interval of the form c_i belongs to the same job as c (thus it has the same weight and length) and is to the right of a_i (so that it does not intersect a_i). See Fig. 5 for an illustration.

State 1: *the algorithm accepted a new interval*. The adversary presents an interval of a new job, a_{-1}. If the algorithm accepts a_{-1}, then c becomes a_{-1}, P remains unchanged and $w(P)/w(c)$ increases by the factor of $w(c)/w(a_{-1}) = 1+\varepsilon$.

State 2: *the algorithm rejected* a_i. The adversary presents c_i. Even if the algorithm accepts c_i, its solution still consists of only one interval, of the same weight as before. The adversary checks $R_i = P \cup \{a_i, c_i\}$. If $(1+\varepsilon)^2 w(R_i)/w(c) \geqslant 3+2\sqrt{2}$, it wins. Otherwise, one can see that $w(P)/w(c) < 4$. If the algorithm accepts c_i, the adversary inserts a_i into P and the ratio $w(P)/w(c)$ increases by a factor larger than $6/5$.

State 3: *the algorithm rejected* c_i. The adversary presents a_{i+1}. If the algorithm accepts a_{i+1}, the adversary replaces P with R_i.

The adversary wins in State 2 under the condition that $w(R_i) = w(P) + w(c) + w(c)(1+\varepsilon)^i$ is sufficiently large in comparison with $w(c)$, or, more precisely,

$$(1+\varepsilon)^2 \frac{w(P) + w(c)(1+\varepsilon)^i + w(c)}{w(c)} \geqslant 3+2\sqrt{2}.$$

We can note that as i grows, the left-hand side increases and eventually the adversary has to win if the algorithm keeps rejecting intervals a_i. Let $y = w(P)/w(c)$ and $z = (1+\varepsilon)^i$. If the adversary did not win, we have $(1+\varepsilon)^2(y+z+1) < 3+2\sqrt{2}$.

Now suppose that the algorithm accepted a_{i+1} in State 3. Then the ratio $w(P)/w(c)$ increases by

$$\frac{w(R_i)}{w(c)(1+\varepsilon)^{i+1}} \cdot \frac{w(c)}{w(P)} = \frac{y+z+1}{(1+\varepsilon)zy} \geqslant \frac{4(y+z+1)}{(1+\varepsilon)(y+z)^2} = \frac{4(x+1)}{(1+\varepsilon)x^2}$$

where $x = y+z$. Note that the last expression is a decreasing function of x. Because the adversary did not win when the algorithm was rejecting c_i, we had $(1+\varepsilon)^2(x+1) < 3+2\sqrt{2}$ and thus our estimate for the increase of $w(P)/w(c)$ is the smallest for $x+1 = (3+2\sqrt{2})(1+\varepsilon)^{-2}$:

$$\frac{4(x+1)}{(1+\varepsilon)x^2} \geqslant \frac{4(3+2\sqrt{2})(1+\varepsilon)^{-2}}{(1+\varepsilon)((3+2\sqrt{2})(1+\varepsilon)^{-2}-1)^2}$$

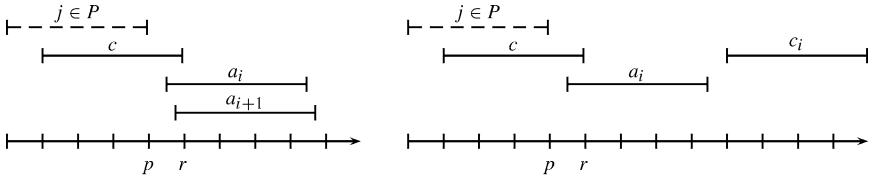

Fig. 6. Illustration of the intervals used by the adversary in the case of equal lengths.

$$= \frac{4(3+2\sqrt{2})(1+\varepsilon)}{(3+2\sqrt{2}-(1+\varepsilon)^2)^2} = \frac{4(3+2\sqrt{2})(1+\varepsilon)}{(2+2\sqrt{2}-2\varepsilon-\varepsilon^2)^2}$$
$$> \frac{4(3+2\sqrt{2})(1+\varepsilon)}{(2+2\sqrt{2})^2} = 1+\varepsilon.$$

As we can choose ε arbitrarily small, we obtain the following theorem.

Theorem 11. *No myopic algorithm for WJISP$_1$ with equal weights per job can achieve approximation ratio better than* $3+2\sqrt{2} \approx 5.828$. *This lower bound applies in the case of arbitrary lengths and in the case of equal lengths per job.*

Now consider WJISP$_1$ with equal weights per job and equal lengths. We can prove that no myopic algorithm can achieve approximation ratio better than 5. The adversary's strategy and its analysis is analogous to the one used to obtain Theorem 11. The only difference is that an interval of type c_i is played only at the end of the game. This means that, after the algorithm has rejected a_i, the adversary plays c_i only if $(w(P)+w(a_i)+w(c_i))/w(c)$ is large enough to win the game; otherwise, it plays a_{i+1}. Fig. 6 gives an illustration. This yields the following theorem (see [10] for a detailed proof).

Theorem 12. *No myopic algorithm for WJISP$_1$ with equal weights per job and equal lengths can achieve approximation ratio better than* 5.

Corollary 8 showed that the myopic algorithm GREEDY$_\alpha$ with $\alpha = 1/2$ achieves approximation ratio 5 for WJISP$_1$ with equal weights per job and equal lengths. Thus, Theorem 12 implies that GREEDY$_\alpha$ is optimal within the class of myopic algorithms in this case. Note that GREEDY$_\alpha$ decides whether it accepts a new interval without taking into account the amount of overlap between a currently accepted interval and the new interval. Intuitively, one might think that considering the amount of overlap would give an advantage to a myopic algorithm, but our tight lower bound shows that this is not true in the worst case.

5.3. WJISP$_1$ with arbitrary weights

To obtain lower bounds in the case of WJISP$_1$ with arbitrary weights, we can employ essentially the same adversary strategy as in the previous section. The only difference is that the interval c_i, which belongs to the same job as c, is now played with weight

$w(c_i) = w(c)(1 + w(a_i)/w(P))/(1 + \varepsilon)$. This ensures that, if the algorithm accepts c_i, the ratio increases from $w(P)/w(c)$ to $w(P \cup \{a_i\})/w(c_i) = (1 + \varepsilon)w(P)/w(c)$. The detailed calculations (see [10]) lead to the following theorems.

Theorem 13. *No myopic algorithm for WJISP$_1$ with arbitrary weights can achieve approximation ratio better than* 7.103. *This lower bound applies in the case of arbitrary lengths and in the case of equal lengths per job.*

The value 7.103 is obtained by minimizing the function

$$f(x) = \frac{x^2 + x}{x - \frac{1}{x} - 1}$$

in the range $((1 + \sqrt{5})/2, \infty)$. The minimum is attained at

$$x = \hat{x} = \frac{2}{3} + \sqrt[3]{\frac{71}{27} + \sqrt{\frac{35}{27}}} + \frac{16}{9\sqrt[3]{\frac{71}{27} + \sqrt{\frac{35}{27}}}} \approx 3.365$$

with $f(\hat{x}) \geqslant 7.103$.

In the case of equal lengths, interval c_i is played only at the end of the game, i.e., if $(w(P) + w(a_i) + w(c_i))/w(c)$ is large enough for the adversary to win. This modification leads to the following theorem.

Theorem 14. *No myopic algorithm for WJISP$_1$ with arbitrary weights and equal lengths can achieve approximation ratio better than* $3 + 2\sqrt{2} \approx 5.828$.

5.4. A randomized lower bound

In Section 5.1 we have shown that no (deterministic) myopic algorithm for JISP can have approximation ratio better than 2. Here we derive a bound on the best approximation ratio that can possibly be achieved by a *randomized myopic* algorithm against an oblivious adversary [5]. The randomized lower bound that we obtain is weaker than the deterministic bound, and it remains an open problem whether randomized myopic algorithms can in fact beat deterministic myopic algorithms for JISP. The following theorem shows that the use of randomization could at best improve the approximation ratio from 2 to approximately 1.582.

Theorem 15. *No randomized myopic algorithm for JISP can achieve an approximation ratio better than* $\frac{e}{e-1} \approx 1.582$.

Proof. We specify a probability distribution on instances with n jobs and show that the expected number of intervals accepted by a deterministic myopic algorithm is at most $\frac{e-1}{e}n + 1$, while an optimal solution consists of n intervals. By Yao's principle (see, e.g., [20, Sections 2.2.2 and 13.3]), this implies the lower bound for randomized myopic algorithms against an oblivious adversary.

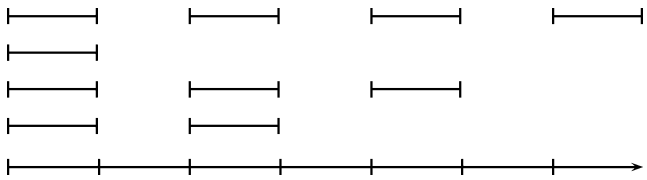

Fig. 7. Example of JISP instance used in randomized lower bound.

We consider a very restricted subset of instances of JISP, defined as follows. If such an instance consists of n jobs, then all its intervals have unit length and any two intervals either have identical endpoints or are disjoint. Initially, one interval with left endpoint 0 and right endpoint 1 is presented from each of the n jobs. Further intervals have left endpoints 2, 4, 6, …, $2n-2$. We refer to the presentation of intervals with left endpoint $2i$ as *round* i. Let J_i denote the set of jobs from which intervals are presented in round i. The instances we construct always satisfy $J_{i+1} \subseteq J_i$ and $|J_{i+1}| = |J_i| - 1$. We say that the unique job in $J_i \setminus J_{i+1}$ *dies* after round i. An example of such an instance is shown in Fig. 7. Note that an optimal solution to such an instance always contains n intervals.

The probability distribution is defined by choosing the job that dies after round i uniformly at random among the $n-i$ jobs from which an interval was presented in round i.

Consider any deterministic myopic algorithm A. If the set of intervals that are currently selected by A does not contain an interval from job j, we say that job j is *available* to A. We can assume without loss of generality that A always selects an interval in round i if at least one interval from an available job is presented in that round.

Let $E_{k,\ell}$ denote the expected number of intervals added to the solution by algorithm A until the end of the game provided that the current round consists of k intervals and ℓ of these k intervals belong to available jobs. Note that $E_{n,n}$ is just the expected number of intervals in the solution computed by A.

The following equations hold:

$$E_{k,0} = 0, \tag{3}$$

$$E_{k,1} = 1, \tag{4}$$

$$E_{k,\ell} = 1 + \frac{\ell-1}{k} E_{k-1,\ell-2} + \frac{k-\ell+1}{k} E_{k-1,\ell-1} \quad \text{for } 1 < \ell \leqslant k. \tag{5}$$

We need to explain why (5) holds. If $\ell > 0$, the algorithm always selects an interval in the current round. After that, among the k intervals of the current round there are $\ell - 1$ remaining intervals belonging to available jobs. With probability $(\ell-1)/k$, one of these $\ell - 1$ jobs dies, and with probability $(k-\ell+1)/k$, one of the other jobs dies.

One can prove by induction that (3)–(5) imply

$$E_{k,\ell} \leqslant k\bigl(1 - e^{-\ell/k}\bigr) + \frac{\ell}{k} \tag{6}$$

for all $0 \leqslant \ell \leqslant k$ (detailed calculations can be found in [10]). For $k = \ell = n$, (6) becomes $E_{n,n} \leqslant \frac{e-1}{e} n + 1$. This shows that the expected number of intervals in the solution computed by A cannot be greater than γn for any $\gamma > \frac{e-1}{e}$, thus establishing the theorem. □

By considering only instances of JISP like the ones constructed in the proof of Theorem 15 we obtain a restricted version of JISP. An application of this restricted version could be as follows. A shop sells n different items. After each week, one of the items becomes unavailable, i.e., after i weeks only $n - i$ of the original items can be bought. Assume that you can afford to buy at most one item per week and that you would like to buy as many different items as possible. This problem is just the restricted version of JISP.

Note that no deterministic myopic algorithm can buy more than $n/2$ items in the worst case: every week, the adversary decides that one of the items that was not yet bought by the algorithm becomes unavailable. After $n/2$ weeks, no items are left to buy for the algorithm. So the deterministic lower bound of 2 from Theorem 10 applies even to this restricted version of JISP.

6. Conclusions

The weighted job interval selection problem has applications in diverse areas. We have studied several variants of this problem: we distinguished the case of arbitrary lengths and the case of equal lengths, and for each of these cases we investigated the setting with all weights equal, equal weights per job, and arbitrary weights. The case of equal lengths can be seen as a natural online scheduling problem. We showed that a simple algorithm called GREEDY$_\alpha$, which belongs to the class of so-called myopic algorithms, outputs a solution with a value that is within a constant factor of the value of an optimal solution. Together with our lower bound results, this implies that for the case of arbitrary lengths no myopic algorithm can do better than GREEDY$_\alpha$ (with respect to approximation ratio) in the unweighted case (for all m) and in the case of equal weights per job (for $m = 1$). In case of arbitrary weights a (small) gap remains. For the case of equal lengths, our results show that GREEDY$_\alpha$ is best possible among the myopic algorithms in case of equal weights per job (for $m = 1$).

An interesting question for future research is whether the use of randomization in myopic algorithms for WJISP can lead to improved approximation ratios. For the case of JISP$_1$, we showed that randomization could at best improve the approximation ratio from 2 to $e/(e-1) \approx 1.582$. For the weighted versions of WJISP, we do not have any strong randomized lower bounds. However, we can remark that all our lower bounds for deterministic myopic algorithms hold also for randomized algorithms against an *adaptive* adversary (i.e., an adversary that can react to the random choices of the algorithm).

Acknowledgments

We are deeply indebted to the anonymous referees for numerous comments and suggestions that helped to improve the paper. In particular, the first referee proposed the use of a union-find data structure to speed up our algorithm for computing the cheapest conflict set (Theorem 1) and found a substantial simplification for the proofs of our lower bound results in Section 5.2.

Furthermore, we thank Eric Torng for bringing the caching application to our attention. We also thank Baruch Schieber and Seffi Naor for discussions regarding the relationship between WJISP and TCSP. Finally, we thank Piotr Berman for supplying preliminary versions of the papers [4,27].

This research was partially supported by EU Thematic Network APPOL II, IST-2001-32007, with funding for the Swiss partners provided by the Swiss Federal Office for Education and Science (BBW).

References

[1] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, M. Protasi, Complexity and Approximation. Combinatorial Optimization Problems and their Approximability Properties, Springer, Berlin, 1999.
[2] A. Bar-Noy, R. Bar-Yehuda, A. Freund, J.S. Naor, B. Schieber, A unified approach to approximating resource allocation and scheduling, J. ACM 48 (5) (2001) 1069–1090.
[3] A. Bar-Noy, S. Guha, J.S. Naor, B. Schieber, Approximating the throughput of multiple machines in real-time scheduling, SIAM J. Comput. 31 (2) (2001) 331–352.
[4] P. Berman, B. DasGupta, Multi-phase algorithms for throughput maximization for real-time scheduling, J. Comb. Optim. 4 (2000) 307–323.
[5] A. Borodin, R. El-Yaniv, Online Computation and Competitive Analysis, Cambridge University Press, 1998.
[6] R. Canetti, S. Irani, Bounding the power of preemption in randomized scheduling, SIAM J. Comput. 27 (4) (1998) 993–1015.
[7] M.C. Carlisle, E.L. Lloyd, On the *k*-coloring of intervals, Discrete Appl. Math. 59 (1995) 225–235.
[8] J. Chuzhoy, R. Ostrovsky, Y. Rabani, Approximation algorithms for the job interval selection problem and related scheduling problems, in: Proc. 42nd Ann. Symp. on Foundations of Computer Science, FOCS'01, 2001.
[9] Y. Crama, O. Flippo, J. van de Klundert, F. Spieksma, The assembly of printed circuit boards: a case with multiple machines and multiple board types, Eur. J. Oper. Res. 98 (1997) 457–472.
[10] T. Erlebach, F. Spieksma, Interval selection: Applications, algorithms, and lower bounds, TIK-Report 152, Computer Engineering and Networks Laboratory, ETH Zurich, October 2002.
[11] M. Fischetti, S. Martello, P. Toth, Approximation algorithms for fixed job schedule problems, Oper. Res. 40 (1992) S96–S108.
[12] A. Frank, Some polynomial algorithms for certain graphs and hypergraphs, in: Proc. 5th British Combinatorial Conference, 1975, pp. 211–226.
[13] M.L. Fredman, D.E. Willard, Trans-dichotomous algorithms for minimum spanning trees and shortest paths, J. Comput. System Sci. 48 (3) (1994) 533–551.
[14] H.N. Gabow, R.E. Tarjan, A linear-time algorithm for a special case of disjoint set union, J. Comput. System Sci. 30 (1985) 209–221.
[15] S.A. Goldman, J. Parwatikar, S. Suri, Online scheduling with hard deadlines, J. Algorithms 34 (2) (2000) 370–389.
[16] M.H. Goldwasser, Patience is a virtue: The effect of slack on competitiveness for admission control, in: Proc. 10th Ann. ACM–SIAM Symp. on Discrete Algorithms, SODA'99, 1999, pp. 396–405.
[17] D. Hochbaum, Approximation Algorithms for NP-hard Problems, PWS, Boston, 1997.
[18] S.v. Hoesel, R. Müller, Optimization in electronic markets: Examples in combinatorial auctions, Netnomics 3 (1) (2001) 23–33.
[19] L. Kroon, M. Salomon, L. van Wassenhove, Exact and approximation algorithms for the tactical fixed interval scheduling problem, Oper. Res. 45 (1997) 624–638.
[20] R. Motwani, P. Raghavan, Randomized Algorithms, Cambridge University Press, 1995.
[21] R. Raman, Priority queues: Small, monotone and trans-dichotomous, in: Proc. 4th Ann. European Symp. on Algorithms, ESA'96, in: Lecture Notes in Comput. Sci., Vol. 1136, 1996, pp. 121–137.

[22] M.H. Rothkopf, A. Pekeč, R.M. Harstad, Computationally manageable combinational auctions, Manag. Sci. 44 (1998) 1131–1147.
[23] A. Seznec, A new case for skewed-associativity, Technical Report RR-3208, INRIA, Rennes, France, July 1997.
[24] F. Spieksma, On the approximability of an interval scheduling problem, J. Sched. 2 (1999) 215–227.
[25] M. Thorup, Undirected single-source shortest paths with positive integer weights in linear time, J. ACM 46 (1999) 362–394.
[26] E. Torng, A unified analysis of paging and caching, Algorithmica 20 (1) (1998) 175–200.
[27] V. Veeramachaneni, P. Berman, W. Miller, Aligning two fragmented sequences, Discrete Appl. Math. (2002), to appear.
[28] G. Woeginger, On-line scheduling of jobs with fixed start and end times, Theoret. Comput. Sci. 130 (1994) 5–16.
[29] M. Yannakakis, F. Gavril, The maximum k-colorable subgraph problem for chordal graphs, Inform. Process. Lett. 24 (1987) 133–137.