Algorithmica (2019) 81:3217–3244
https://doi.org/10.1007/s00453-019-00582-9

Flexible Resource Allocation to Interval Jobs

Dmitriy Katz[1] · Baruch Schieber[2] · Hadas Shachnai[3]

Received: 18 January 2018 / Accepted: 23 April 2019 / Published online: 7 May 2019

Abstract
Motivated by the cloud computing paradigm, and by key optimization problems in all-optical networks, we study two variants of the classic job interval scheduling problem, where a reusable resource is allocated to competing job intervals in a *flexible* manner. Each job, J_i, requires the use of up to $r_{max}(i)$ units of the resource, with a profit of $p_i \geq 1$ accrued for each allocated unit. The goal is to feasibly schedule a subset of the jobs so as to maximize the total profit. The resource can be allocated either in *contiguous* or *non-contiguous* blocks. These problems can be viewed as flexible variants of the well known *storage allocation* and *bandwidth allocation* problems. We show that the contiguous version is strongly NP-hard, already for instances where all jobs have the same profit and the same maximum resource requirement. For such instances, we derive the best possible positive result, namely, a polynomial time approximation scheme. We further show that the contiguous variant admits a $(\frac{5}{4}+\varepsilon)$-approximation algorithm, for any fixed $\varepsilon > 0$, on instances whose job intervals form a *proper* interval graph. At the heart of the algorithm lies a non-standard parameterization of the approximation ratio itself, which is of independent interest. For the non-contiguous case, we uncover an interesting relation to the *paging* problem that leads to a simple $O(n \log n)$ algorithm for *uniform* profit instances of n jobs. The algorithm is easy to implement and is thus practical.

Keywords Elastic all-optical networks · Cloud computing · Scheduling · Paging problem · Flexible storage allocation · Bandwidth allocation · Approximation algorithms

A preliminary version of this paper appeared in Proceedings of the 28th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA), Asilomar State Beach, July 2016.

This work was partly carried out during visits to DIMACS supported by the National Science Foundation under Grants Number CCF-1144502 and CCF-1445755. Work partially supported also by the Technion V.P.R. Fund, and by the Smoler Research Fund.

Extended author information available on the last page of the article

1 Introduction

1.1 Background and Motivation

Interval scheduling is one of the basic problems in the study of algorithms, with a wide range of applications in computer science and in operations research (see, e.g., [21]). We focus on scheduling intervals with resource requirements. In this model, we have a set of intervals (or, *activities*) competing for a reusable resource. Each activity utilizes a certain amount of the resource for the duration of its execution and frees it upon completion. The problem is to find a feasible schedule of the activities that satisfies certain constraints, including the requirement that the total amount of resource allocated simultaneously to activities never exceeds the amount of resource available.

In this classic model, two well-studied variants are the storage allocation (see, e.g., [5,22]) and the bandwidth allocation problems (see, e.g., [3,11]). In the *storage allocation problem* (SAP), each activity requires the allocation of a *contiguous* block of the resource for the duration of its execution. Thus, the input is often viewed as a set of axis-parallel rectangles; the goal is to pack a maximum profit subset of rectangles into a horizontal strip of a given height, by sliding the rectangles vertically but not horizontally. When the resource can be allocated in *non-contiguous* blocks, we have the *bandwidth allocation problem* (BAP), where we only need to allocate to each activity the required amount of the resource.

Modern technologies used in scheduling jobs that require cloud services (see, e.g., [17,24]), or in spectrum assignment in elastic all-optical networks [14,33], allow *flexible* allocation of the available resources. These technologies are attracting wide interest, due to their higher efficiency and better utilization of compute and network resources. However, allocating the resources becomes even more challenging, compared to previous rigid technologies. This motivates our study of the flexible variants of the above problems.

1.2 Problem Statement

We consider a variant of SAP where each interval can be allocated any amount of the resource up to its maximum requirement, with a profit accrued from each resource unit allocated to it. The goal is to allocate to the intervals *contiguous* blocks of the resource so as to maximize the total profit. We refer to this variant below as the *flexible storage allocation problem* (FSAP).

We also consider the *flexible bandwidth allocation problem* (FBAP), where each interval specifies an upper bound on the amount of the resource it can be allocated, as well as the profit accrued from each allocated unit of the resource. The goal is to determine the amount of resource which can be feasibly allocated to each interval, so as to maximize the total profit.

In our general framework, the input consists of a set $\mathcal{J}$ of n intervals. Each interval $J_i \in \mathcal{J}$ requires the utilization of a given, limited, resource. The amount of resource available, denoted by $W > 0$, is fixed over time. Each interval J_i is defined by the following parameters.

(1) A left endpoint, $s_i \geq 0$, and a right endpoint, $e_i \geq 0$. In this case J_i is associated with the half-open interval $[s_i, e_i)$ on the real line.
(2) The amount of resource allocated to each interval, J_i, which can take any value up to the *maximum possible value* for J_i, given by $r_{max}(i)$.
(3) The profit $p_i \geq 1$ gained for each unit of the resource allocated to J_i.

A *feasible* solution has to satisfy the following conditions. (i) Each interval $J_i \in \mathcal{J}$ is allotted an amount of the resource in its given range, which does not change over time. (ii) The total amount of the resource allocated at any time does not exceed the available amount, W. In FBAP, we seek a feasible allocation which maximizes the total profit accrued by the intervals. In FSAP, we add the requirement that the allocation to each interval is a *contiguous* block of the resource.[1]

1.3 Applications

We mention below two primary applications of our problems.

Scheduling time-sensitive jobs on large computing clusters. In the cloud computing paradigm, jobs that require cloud services are often time-critical (see, e.g. [16,24]). Thus, each job is associated with arrival time, a due date, a maximum resource requirement, and the cost paid per allocated unit of the resource. Given a large computing cluster with a total available resource W (e.g. bandwidth, or storage capacity), consider job instances with strict timing constraints, where each job has to start processing upon arrival. The scheduler needs to schedule a feasible subset of the jobs, and assign to each some amount of the resource, so as to maximize the total profit. When jobs require a *contiguous* allocation of the resource, we have an instance of FSAP. When allocation may be non-contiguous, we have an instance of FBAP.

Spectrum allocation in elastic optical networks. In all-optical networks, several high-speed signals connecting different source-destination pairs may share a link, provided they are transmitted on carriers having different wavelengths of light (see, e.g., [26]). Traditionally, the spectrum of light that can be transmitted through the fiber has been divided into frequency intervals of fixed width with a gap of unused frequencies between them. In the emerging *flexgrid* technology (see, e.g., [14,18]), the usable frequency intervals are of variable width. As in the traditional model, two different signals using the same link have to be assigned disjoint sub-spectra. Thus, given a set of connection requests in a path network, each associated with a profit per allocated spectrum unit, in FSAP we need to feasibly allocate to the requests contiguous frequency intervals, with the goal of maximizing the total profit. FBAP corresponds to the model where the sub-spectra allocated to each request need not be contiguous.

1.4 Our Contribution

Given an algorithm $\mathcal{A}$, let $\mathcal{A}(\mathcal{J})$, $OPT(\mathcal{J})$ denote the profit of $\mathcal{A}$ and an optimal solution for a problem instance $\mathcal{J}$, respectively. For $\rho \geq 1$, we say that $\mathcal{A}$ is a ρ-approximation algorithm if, for any instance $\mathcal{J}$, $\frac{OPT(\mathcal{J})}{\mathcal{A}(\mathcal{J})} \leq \rho$.

[1] This is made precise in Sect. 2.

We derive both positive and negative results. On the positive side, we uncover an interesting relation of FBAP to the classic *paging* problem that leads to a simple $O(n \log n)$ algorithm for *uniform* profit instances (see Sect. 3). Thus, we substantially improve the running time of the best known algorithm for FBAP (due to [31]) which uses flow techniques. Our algorithm is easy to implement and is thus practical.

On the negative side, we show (in Sect. 5) that FSAP is strongly NP-hard, already for instances where all jobs have the same profit and the same maximum resource requirement, Max. For such instances, we derive (in Sect. 6.2) the best possible positive result, namely, a *polynomial time approximation scheme (PTAS)*. We also present (in Sect. 6.1) a $\frac{2k}{2k-1}$-approximation algorithm, where $k = \lceil \frac{W}{\texttt{Max}} \rceil$, which is of practical interest. We further show (in Sect. 4) that FSAP admits a $(\frac{5}{4} + \varepsilon)$-approximation algorithm, for any fixed $\varepsilon > 0$, on instances whose job intervals form a proper interval graph.[2]

Techniques. Our Algorithm, Paging_FBA, for the non-contiguous version of the problem, uses an interesting relation to the offline *paging* problem.[3] The key idea is to view the available resource as slots in fast memory, and each job (interval) J_i as $r_{max}(i)$ pairs of requests for pages in the main memory. Each pair of requests is associated with a distinct page: one request at s_i and one at $e_i - 1$. We apply Belady's offline paging algorithm [7], that handles a page fault by evicting the page whose next request is furthest in the future (see Sect. 3). If a page remains in the fast memory between the two times it was requested, the resource associated with its fast memory slot is allocated to the corresponding job. In fact, Paging_FBA solves the flexible bandwidth allocation problem optimally for more general instances, where each interval J_i has also a lower bound, $r_{min}(i) > 0$ on the amount of resource J_i is allocated (see Sect. 3).[4]

At the heart of our $(\frac{5}{4} + \varepsilon)$-approximation algorithm for proper instances lies a non-standard parameterization of the approximation ratio itself. Specifically, the algorithm uses a parameter $\beta \in (0, 1)$ to guess the fraction of total profit obtained by *wide* intervals, i.e., intervals with high maximum resource requirement, in some optimal solution. If the profit from these intervals is at least this fraction β of the optimum for the given instance, such a high profit subset of wide intervals is found by the algorithm; else, the algorithm proceeds to find a high profit subset of *narrow* and *wide* intervals, by solving an LP relaxation of a modified problem instance. In solving this instance, we require that the profit from *extra* units of the resource assigned to wide intervals (i.e., above certain threshold value) is bounded by a β fraction of the optimum (see Sect. 4). This tighter constraint guarantees a small loss in profit when rounding the (fractional) solution for the LP. The approximation ratio of $(\frac{5}{4} + \varepsilon)$ is attained by optimizing on the value of β.

[2] We formally define proper instances in Sect. 2.

[3] This relation was used before, also for solving unsplittable flow on paths and weighted interval scheduling (see, e.g., [26]). To the best of our knowledge, for FBAP it is used here for the first time.

[4] In obtaining all other results, we assume that $r_{min}(i) = 0$ for $1 \leq i \leq n$.

1.5 Related Work

The classic interval scheduling problem, where each interval requires all of the resource for its execution, is solvable in $O(n \log n)$ time [3]. Storage allocation problems have been studied since the 1960's (see, e.g., [20]). The traditional goal is to contiguously store a set of objects in minimum size memory. This is the well-known *dynamic storage allocation* (DSA) problem (see, e.g., [8] and the references therein). The *storage allocation problem* (SAP), the throughput version of DSA, is NP-hard since it includes Knapsack as a special case. SAP was first studied in [3,22]. Bar-Noy et al. [3] presented an approximation algorithm that yields a ratio of 7. Chen et al. [12] presented a polynomial time exact algorithm for the special case where all resource requirements are multiples of W/K, for some fixed integer $K \geq 1$. They developed an $O(n(nK)^K)$ time dynamic programming algorithm to solve this special case of SAP, and also gave an approximation algorithm with ratio $\frac{e}{e-1}+\varepsilon$, for any $\varepsilon > 0$, assuming that the maximum resource requirement of any interval is $O(W/K)$. Bar-Yehuda et al. [4] presented a randomized algorithm for SAP with ratio $2+\varepsilon$, and a deterministic algorithm with ratio $\frac{2e-1}{e-1} + \varepsilon < 2.582$. The best known result is a deterministic $(2+\varepsilon)$-approximation algorithm due to [29].

The *bandwidth allocation problem* (BAP) is known to be strongly NP-hard, already for uniform profits [13]. The results of Albers et al. [1] imply a constant factor approximation (where the constant is about 22). The ratio was improved to 3 by Bar-Noy et al. [3]. Calinescu et al. [9] developed a randomized approximation algorithm for BAP with expected performance ratio of $2+\varepsilon$, for every $\varepsilon > 0$. The best known result is an LP-based deterministic $(2+\varepsilon)$-approximation algorithm for BAP due to Chekuri et al. [11].

Both BAP and SAP have been widely studied also in the *non-uniform* resource case, where the amount of available resource may change over time. In this setting, BAP can be viewed as the *unsplittable flow problem* (*UFP*) on a path. The best known polynomial time result is a $(\frac{5}{3}+\varepsilon)$-approximation due to Grandoni et al. [15]. Bansal et al. [2] developed a quasi-polynomial time approximation scheme for the problem. Batra et al. [6] obtained approximation schemes for some special cases.[5] For SAP with non-uniform resource, the best known ratio is $2+\varepsilon$, obtained by a randomized algorithm of Mömke and Wiese [25].

The *flexible* variants of SAP and BAP were introduced by Shalom et al. [31]. The authors study instances where each interval i has a minimum and a maximum resource requirement, satisfying $0 \leq r_{min}(i) < r_{max}(i) \leq W$, and the goal is to find a maximum profit schedule, such that the amount of resource allocated to each interval i is in $[r_{min}(i), r_{max}(i)]$. The authors show that FBAP can be optimally solved using flow techniques. The paper also presents a $\frac{4}{3}$-approximation algorithm for FSAP instances in which the input graph is proper, and $r_{min}(i) \leq \lceil \frac{r_{max}(i)}{2} \rceil$, for all $1 \leq i \leq n$. The paper [30] shows NP-hardness of FSAP instances where each interval has positive lower and upper bounds on the amount of resource it can be allocated. The problem remains intractable even if the bounds are identical for all intervals, i.e., $r_{min}(i) = \texttt{Min}$ and $r_{max}(i) = \texttt{Max}$, for all i, where $0 < \texttt{Min} < \texttt{Max} \leq W$. The authors also show

[5] See also the results on *UFP with Bag constraints* (*BagUFP*) [10].

that FSAP is NP-hard for the subclass of instances where $r_{min}(i) = 0$ and $r_{max}(i)$ is arbitrary, for all i, and present a $(2+\varepsilon)$-approximation algorithm for such instances, for any fixed $\varepsilon > 0$. We strengthen the hardness result of [30], by showing that FSAP is strongly NP-hard even if $r_{min}(i) = 0$ and $r_{max}(i) = \texttt{Max}$, for all i.

Finally, the paper [29] considers variants of FSAP and FBAP where $0 \leq r_{min}(i) < r_{max}(i) \leq W$, and the goal is to feasibly schedule a *subset* S of the intervals of maximum total profit (namely, the amount of resource allocated to each interval $i \in S$ is in $[r_{min}(i), r_{max}(i)]$). The paper presents a 3-approximation algorithm for this version of FBAP, and a $(3+\varepsilon)$-approximation for the corresponding version of FSAP, for any fixed $\varepsilon > 0$.

2 Preliminaries

We represent the input $\mathcal{J}$ as an interval graph, $G = (V, E)$, in which the set of vertices, V, represents the n jobs, and there is an edge $(v_i, v_j) \in E$ if the intervals representing the jobs J_i, J_j intersect. For simplicity, we interchangeably use J_i to denote the i-th job, and the interval corresponding to the i-th job on the real-line. We say that an input $\mathcal{J}$ is *proper*, if in the corresponding interval graph, $G = (V, E)$, no interval J_i is completely contained in another interval J_j, for all $1 \leq i, j \leq n$.[6]

Throughout the paper, we use *coloring* terminology when referring to the assignment of resource to the jobs. Specifically, the amount of available resource, W, can be viewed as the amount of available distinct colors. Thus, the demand of a job J_i for (contiguous) allocation from the resource, where the allocated amount is an integer in the range $[0, r_{max}(i)]$, can be satisfied by *coloring* J_i with a (contiguous) set of colors of size in the range $[0, r_{max}(i)]$.

Let $C = \{1, 2, \ldots, W\}$ denote the set of available colors. Given a coloring of the intervals, $c : \mathcal{J} \rightarrow 2^C$, let $|c(J_i)|$ be the number of colors assigned to J_i. The total profit accrued from c is then $P_c(\mathcal{J}) = \sum_{i=1}^{n} p_i |c(J_i)|$. Similarly, the total profit accrued for a subset of intervals $\mathcal{J}' \subseteq \mathcal{J}$ is $P_c(\mathcal{J}') = \sum_{i \in \mathcal{J}'} p_i |c(J_i)|$. Recall that in a *contiguous* coloring, c, each interval J_i is assigned a *block* of $|c(J_i)|$ consecutive colors in $\{1, \ldots, W\}$. In a *circular* contiguous coloring, c, we have the set of colors $\{0, 1, \ldots, W-1\}$ positioned consecutively on a circle. Each interval $J_i \in \mathcal{J}$ is assigned a block of $|c(J_i)|$ consecutive colors on the circle. Formally, J_i can be assigned *any* consecutive sequence of $|c(J_i)|$ indices, $\{\ell, (\ell+1) \bmod W, \ldots, (\ell + |c(J_i)| - 1) \bmod W\}$, where $0 \leq \ell \leq W-1$.

Let $S \subseteq \mathcal{J}$ be the subset of jobs J_i for which $|c(J_i)| \geq 1$ in a (contiguous) coloring C for the input graph G. We call the subgraph of G induced by S, $G_S = (S, E_S)$, the *support graph* of S.

3 The Flexible Bandwidth Allocation Problem

In this section we study FBAP, the non-contiguous version of our problem. We consider a generalized version of FBAP, where each job J_i has also a lower bound $r_{min}(i)$ on the amount of resource it is allocated.

[6] In the context of scheduling, we say that the (interval) jobs are 'agreeable', or 'similarly ordered'.

Shalom et al. [31] showed that this generalized FBAP can be solved optimally by using flow techniques. We show that in the special case where all jobs have the same (unit) profit per allocated color (i.e., resource unit), the problem can be solved by an efficient algorithm based on Belady's well known algorithm for offline paging [7]. Recall that in Belady's algorithm (also called MIN) whenever there is a need to evict a page from the fast memory to make room for a page that is being requested, the evicted page should be the one whose next request occurs furthest in the future. (Pages that will never again be requested are treated as pages whose next request is in infinity.) While Belady's algorithm is easy to state its correctness proof is rather involved. A short correctness proof appears in [27].

From now on, assume that we have a feasible instance, that is, there are enough colors to allocate at least $r_{min}(i)$ colors to each job J_i.

To gain some intuition, assume first that $r_{min}(i) = 0$ for all $i \in [1..n]$. We view the available colors as slots in fast memory, and each job (interval) J_i as $r_{max}(i)$ pairs of requests for pages in the main memory. Each pair of requests is associated with a distinct page: one request at s_i and one at $e_i - 1$. We now apply Belady's offline paging algorithm: if a page remains in the fast memory between the two times it was requested, then the color that corresponds to its fast memory slot is allocated to the corresponding job.

When $r_{min}(i) > 0$, we follow the same intuition while allocating at least $r_{min}(i)$ colors to each J_i, to ensure feasibility. We show below that the optimality of the paging algorithm implies the optimality of the solution for our FBAP instance.

The algorithm is implemented iteratively, by reassigning colors as follows. The algorithm scans the left endpoints of the intervals, from left to right. When the algorithm scans s_i, it first assigns $r_{min}(i)$ colors to J_i to ensure feasibility. The algorithm starts by assigning the available colors. If there are less than $r_{min}(i)$ colors available at s_i, the algorithm examines the intervals intersecting J_i at s_i in decreasing order of right endpoints, and feasibly decreases the number of colors assigned to these intervals and reassigns them to J_i, until J_i is allocated $r_{min}(i)$ colors. The feasibility of the instance implies that so many colors can be reassigned.

Next, the algorithm allocates up to $r_{max}(i) - r_{min}(i)$ additional colors to interval J_i, to maximize profit. If $r_{max}(i) - r_{min}(i)$ colors are available at s_i, then they are assigned to J_i. If only less colors are available, and thus J_i is assigned less than $r_{max}(i)$ colors, then the algorithm follows Belady's algorithm to potentially assign additional colors to J_i. The algorithm examines the intervals intersecting J_i at s_i, and in case there are such intervals with larger right endpoint than e_i, it feasibly decreases the number of colors assigned to such intervals with the largest right endpoints (furthest in the future), and increases the number of colors assigned to J_i, up to $r_{max}(i)$.

When the algorithm scans e_i, the right endpoint of an interval J_i, the colors assigned to J_i are released and become available. The pseudocode for Paging_FBA is given in Algorithm 3.1.

Theorem 1 Paging_FBA *is an optimal* $O(n \log n)$ *time algorithm, for any* FBAP *instance where* $p_i = 1$ *for all* $1 \leq i \leq n$.

Proof The proof is by reduction to the multipaging problem. The multipaging problem is a variant of the paging problem, where more than one page is requested at the same

Algorithm 3.1 Paging_FBA($\mathcal{J}, \bar{r}_{min}, \bar{r}_{max}, W$)

```
1: AVAIL = [1..W].
2: W.l.o.g. assume that all endpoints are distinct. Sort the endpoints of the intervals in J in increasing
   order. Let L denote the sorted list.
3: while not end-of-list do
4:     Consider the next endpoint L.
5:     if the endpoint is s_i, the left endpoint of J_i, then
6:         A_i = ∅.
7:         if |AVAIL| > r_min(i) then
8:             Move r_min(i) colors from AVAIL to A_i.
9:         else
10:            A_i = AVAIL
11:            AVAIL = ∅.
12:            while |A_i| < r_min(i) do
13:                Among all intervals such that s_k < s_i and |A_k| > r_min(k), let J_k be the interval with
                   maximum right endpoint e_k.
14:                Move min{|A_k| − r_min(k), r_min(i) − |A_i|} colors from A_k to A_i.
15:        Move min{|AVAIL|, r_max(i) − r_min(i)} colors from AVAIL to A_i.
16:        Let J_i = {J_k ∈ J | s_k < s_i ∧ e_k > e_i ∧ |A_k| > r_min(k)}.
17:        while |A_i| < r_max(i) ∧ |J_i| > 0 do
18:            Let J_k be the interval with the maximum right endpoint in J_i.
19:            Move min{|A_k| − r_min(k), r_max(i) − |A_i|} colors from A_k to A_i
20:    else
21:        {The endpoint is e_i, the right endpoint of J_i}
22:        AVAIL = AVAIL ∪ A_i
23: Return the coloring c given by the sets A_1, ..., A_n of colors assigned to the jobs in J.
```

time. Liberatore [23] proved that the multipaging version of Belady's algorithm is optimal. In this version whenever there is a need to evict $k \geq 1$ pages from the fast memory to make room for k pages that are being requested, the evicted pages should be the k pages in fast memory whose next request occur furthest in the future. (Pages that will never again be requested are treated as pages whose next request is in infinity.)

Given an instance of FBAP, where $p_i = 1$ for all $1 \leq i \leq n$, define a respective multipaging problem instance as follows. The size of the fast memory is $M = \sum_{J_i \in \mathcal{J}} r_{max}(i)$. For each job $J_i \in \mathcal{J}$ we associate $M - W + r_{max}(i)$ new pages, and define three types of page requests: *feasibility* requests, *profit* requests, and *filler* requests. At each $3s_i \leq t < 3e_i$ there are $r_{min}(i)$ feasibility requests for the same $r_{min}(i)$ of the associated pages. There are $r_{max}(i) - r_{min}(i)$ pairs of profit requests for $r_{max}(i) - r_{min}(i)$ of the associated pages. The first request of each pair is at $3s_i$ and the second at $3e_i - 1$. In addition, at $3s_i + 1$ there are $M - W$ filler requests for the remaining $M - W$ of the associated pages.

Belady's algorithm for the multipaging instance yields an eviction policy that minimizes the number of page faults. Whenever a page that has not been requested before is requested we have an unavoidable page fault. Thus, we must have $(M - W)n + M$ page faults, $(M - W)n$ for the filler request pages, and M for the $r_{max}(i)$ feasibility and profit requests at $3s_i$, for each job $J_i \in \mathcal{J}$. The filler request pages cannot generate any additional page faults since each such page is requested only once. For any job $J_i \in \mathcal{J}$, the feasibility request pages are requested continuously from time $3s_i$ to time $3e_i - 1$. Thus, they cannot be evicted in this time interval, and cannot generate any additional page faults. It follows that additional page faults may occur only when profit request

pages are evicted at some point between the time they were requested first to the time of their second request. Minimizing the number of such page faults is equivalent to maximizing the number of pairs of profit requests whose corresponding pages stay in memory from the time of their first request to the time of their second request, that is, maximizing the sum over all jobs $J_i \in \mathcal{J}$ of the number of profit request pages associated to job J_i that stay in memory throughout the interval $[3s_i, 3e_i - 1)$.

Note that Paging_FBA simulates Belady's algorithm in the following sense. After a left endpoint s_i is scanned, the colors assigned to the jobs correspond to the memory slots in the fast memory allocated at time $3s_i + 1$ to feasibility and profit requests associated with jobs whose right endpoint is greater than s_i. The construction (and our assumption that the instance is feasible) guarantees that at least $r_{min}(i)$ colors are assigned to each job $J_i \in \mathcal{J}$. The optimality of Belady's algorithm guarantees that the total number of additional colors allocated to all jobs is maximized, thus, maximizing the profit of allocated resources in the FBAP instance.

Paging_FBA can be implemented in $O(n \log n)$ time as follows. Clearly, sorting the intervals by left endpoints can be done in $O(n \log n)$ time. To implement the color reassignments we store the intervals in a priority queue by right endpoints. An interval is inserted to the priority queue exactly once when its left endpoint is scanned. Each time the priority queue is queried for the interval J_k with the maximum right endpoint either J_k is removed from the priority queue, or we assign either $r_{min}(i)$ or $r_{max}(i)$ colors to interval J_i whose left endpoint is scanned. Additionally, an interval is removed from the priority queue when its right endpoint is scanned (if it has not been removed already). Thus, the total number of priority queue operations is linear in n. Since each priority queue operation can be implemented in $O(\log n)$ time, the total running time is $O(n \log n)$. □

4 Approximating Flexible Storage Allocation

In this section we consider the flexible storage allocation problem. We focus below on FSAP instances in which the jobs form a *proper* interval graph, and give an approximation algorithm that yields a ratio of $(\frac{5}{4} + \varepsilon)$ to optimal.

Our Algorithm, Proper_FSAP, uses the parameters $\varepsilon > 0$ and $\beta \in (0, 1)$ (to be determined). Initially, Proper_FSAP guesses the value of an optimal solution $OPT_{\text{FSAP}}(\mathcal{J})$. The guessing is done by binary search. As will be evident from the analysis of Proper_FSAP, if it fails to output a solution that is at least the value of the guess divided by $(\frac{5}{4} + \varepsilon)$, then the guess is an overestimate of the optimal value. We start the binary search with two guesses, one is zero which is guaranteed to be a lower bound on $OPT_{\text{FSAP}}(\mathcal{J})$, and one is the sum of the maximum profits of all jobs, which is guaranteed to be an upper bound on $OPT_{\text{FSAP}}(\mathcal{J})$. At each binary search iteration we run Proper_FSAP with a guess value that is equal to the median of the current lower and upper bounds. If this guess is proven to be an overestimate we set the upper bound to be the value of the guess, otherwise we update the lower bound to be the value of the guess. After a polynomial (in the input length) number of iterations we can find a lower and an upper bound whose difference is no more than the value of $\min_{i \in [1..n]} p_i$. At this point we are guaranteed that the lower bound is at least $OPT_{\text{FSAP}}(\mathcal{J})$.

Let $\mathcal{J}_{wide}$ denote the set of *wide* intervals J_i for which $r_{max}(i) \geq \varepsilon W$. Let $\mathcal{J}_{narrow}$ denote the complement set of *narrow* intervals. Algorithm Proper_FSAP handles separately two cases. (i) The profit from the *wide* intervals that are actually assigned at least εW colors is *large*, namely, at least $\beta \cdot OPT_{\text{FSAP}}(\mathcal{J})$. Then such a solution is found and returned by the algorithm. We show (in the proof of Lemma 3) that this can be done in polynomial time using dynamic programming. (ii) Any solution that achieves a profit of at least $\beta \cdot OPT_{\text{FSAP}}(\mathcal{J})$ must either include a narrow interval or assign less than εW colors to a wide interval. In this case Proper_FSAP calls Algorithm $\mathcal{A}_{Narrow\_Color}$ that finds a solution of profit at least $(\frac{4-\beta}{4} - \varepsilon) \cdot OPT_{\text{FSAP}}(\mathcal{J})$ accrued from both *narrow* and *wide* intervals, for any fixed $\varepsilon > 0$. The pseudocode for Proper_FSAP follows.

Algorithm 4.1 Proper_FSAP($\mathcal{J}, \bar{r}_{max}, \bar{p}, W, \varepsilon, \beta$)

1: Guess $OPT_{\text{FSAP}}(\mathcal{J})$, the value of an optimal solution of FSAP on $\mathcal{J}$.
2: Find in $\mathcal{J}_{wide}$ a solution S of FSAP with maximum total profit among all solutions in which intervals that are assigned colors are assigned at least $\lceil \varepsilon W \rceil$ colors.
3: **if** $P(S) < \beta OPT_{\text{FSAP}}(\mathcal{J})$ **then**
4: Let S be the solution output by
5: $\mathcal{A}_{Narrow\_Color}(\mathcal{J}, \bar{r}_{max}, \bar{p}, W, \varepsilon, \beta OPT_{\text{FSAP}}(\mathcal{J}))$
6: Return S and the respective contiguous coloring

We now describe Algorithm $\mathcal{A}_{Narrow\_Color}$ that finds an approximate solution for FSAP in case the extra profit of the *wide* intervals − above the profit of their first $\lceil \varepsilon W \rceil$ assigned colors − is bounded by β fraction of the optimal solution.

First, $\mathcal{A}_{Narrow\_Color}$ solves a linear program LP_{FBA} that finds a (fractional) maximum profit solution for FBAP on the set $\mathcal{J}$, in which the number of colors used is no more than $W' = \lfloor (1-\varepsilon)W \rfloor$. Note that, from our guess, the value of the solution is at least $(1-\varepsilon)OPT_{\text{FSAP}}(\mathcal{J})$. This holds since the value of an optimal solution for LP_{FBA} when all W colors are used is at least $OPT_{\text{FSAP}}(\mathcal{J})$. The solution needs to satisfy an upper bound on the extra profit accrued from *wide* intervals that are assigned more than εW colors.

Next, this solution is rounded to an integral solution of the FBAP instance, of value at least $(1-2\varepsilon)OPT_{\text{FSAP}}(\mathcal{J})$. $\mathcal{A}_{Narrow\_Color}$ proceeds by converting the resulting (non-contiguous) coloring c to a contiguous circular coloring c' of the same profit. Such a coloring can be obtained, e.g., by the greedy algorithm ProperToCircular proposed in [31]. Initially, the intervals are ordered in increasing order by their left endpoints. Renumber the intervals such that J_i, $1 \leq i \leq n$, is the i-th in this ordering. Then the colors $\{0, \ldots, W' - 1\}$ are positioned on a circle clockwise, starting from 0. J_1 is assigned the colors $\{0, 1, \ldots, |c(J_1)| - 1\}$. For $i > 1$, the interval J_i is assigned a contiguous set of colors $\{f_i, (f_i + 1) \bmod W', \ldots, (f_i + |c(J_i)| - 1) \bmod W'\}$, where $(f_i - 1) \bmod W'$ is the last color assigned to J_{i-1}.

Finally, the coloring c' is converted to a valid (non-circular) coloring, c''. Consider the assignment of colors to the intervals and the W' points representing the colors on the circle. Now, each point $\ell \in \{0, \ldots, W' - 1\}$ has a profit accrued from the intervals that are assigned the corresponding color. To obtain a contiguous non-circular coloring

c'' $\mathcal{A}_{Narrow\_Color}$ searches for the best index ℓ for 'cutting' the circular coloring. This is done by examining a polynomial number of integral points $\ell \in \{0, \ldots, W' - 1\}$ and calculating in each the loss in profit due to eliminating at most half of the colors for each *wide* interval whose (contiguous) color set includes color ℓ. The algorithm 'cuts the circle' in the point ℓ which causes the smallest harm to the total profit. For each *wide* interval J_i 'crossing' the point ℓ, we assign the largest among its first block of colors (whose last color is ℓ), and the second block of colors (which starts at $(\ell+1) \bmod W'$). For each *narrow* interval that included ℓ, we assign the same number of new colors in the range $\{W'+1, \ldots, W\}$. We give the pseudocode of $\mathcal{A}_{Narrow\_Color}$ in Algorithm 4.2.

Algorithm 4.2 $\mathcal{A}_{Narrow\_Color}(\mathcal{J}, \bar{r}_{max}, \bar{p}, W, \varepsilon, P)$

Solve the linear program LP_{FBA}
Round the solution to obtain a (non-contiguous) coloring c.
Find a circular contiguous coloring c' of the same total profit as c.
Let S' be the set of colored intervals in c'.
Let $S'_w \subseteq S'$ be the subset of intervals $J_i \in S'$ for which $|c'(J_i)| \geq \varepsilon W$, and let $S'_n = S' \setminus S'_w$.
for any $J_i \in S'_w$ round down $|c'(J_i)|$ to the nearest integral multiple of $\varepsilon^2 W$, and eliminate the corresponding amount of colors in $c'(J_i)$, such the first color assigned to J_i is $f_i = \lfloor r \cdot \varepsilon^2 W \rfloor$, for some integer $r \geq 0$.
for $\ell = \lfloor r \cdot \varepsilon^2 W \rfloor, r = 0, 1, \ldots, \lceil \frac{1}{\varepsilon^2} \rceil$ **do**
  Let $S'_w(\ell) = \{J_i \in S'_w | \{\ell, (\ell+1) \bmod W'\} \subseteq c'(J_i)\}$
  Let $P(S'_w(\ell)) = 0$
  for $J_i \in S'_w(\ell)$ **do**
    Suppose that $c'(J_i) = \{f_i, (f_i+1) \bmod W', \ldots, t_i\}$ for some $0 \leq f_i, t_i \leq W' - 1$.
    Partition the set of $|c'(J_i)|$ colors assigned to J_i into two contiguous blocks: $Block_1(i) = \{f_i, \ldots, \ell\}$, and $Block_2(i) = \{(\ell+1) \bmod W', \ldots, t_i\}$.
    $P(S'_w(\ell)) = P(S'_w(\ell)) + p_i \cdot \min\{|Block_1(i)|, |Block_2(i)|\}$
Let $\ell_{good} = \operatorname{argmin}_\ell P(S'_w(\ell))$.
for $J_i \in S'_w(\ell_{good})$ **do**
  Assign to J_i the larger of $Block_1(i)$ and $Block_2(i)$.
Renumber the color $\{(\ell_{good} + s) \bmod W'$ by $s+1$ for all $s \in \{0, \ldots, W' - 1\}$
Let $S'_n(\ell_{good}) = \{J_i \in S'_n | \{\ell_{good}, (\ell_{good}+1) \bmod W'\} \subseteq c'(J_i)\}$
for $J_i \in S'_n(\ell_{good})$ **do**
  Assign to J_i $|c'(J_i)|$ contiguous colors in the set $\{W'+1, \ldots, W\}$.
Return c'', the resulting coloring of S'.

4.1 Analysis of Proper _FSAP

We note that if $W \leq 1/\varepsilon^2$ then at any time $t > 0$, the number of active intervals is bounded by $\lfloor 1/\varepsilon^2 \rfloor$, which is a constant. For such instances FSAP can be solved optimally, using dynamic programming (see Lemma 5 in [31]). Thus, we assume below that $W > 1/\varepsilon^2$. Our main result is the following.

$$
\begin{aligned}
(LP_{\text{FBA}}): \quad \max \quad & \sum_{J_i \in \mathcal{J}_{narrow}} p_i x_i + \sum_{J_i \in \mathcal{J}_{wide}} p_i (x_i + y_i) \\
\text{s.t.} \quad & \sum_{J_i \in \mathcal{J}_{wide}} p_i y_i \leq \beta(1-\varepsilon) OPT_{\text{FSAP}}(\mathcal{J}) && (1) \\
& \sum_{\{J_i \in \mathcal{J}: t \in J_i\}} x_i + \sum_{\{J_i \in \mathcal{J}_{wide}: t \in J_i\}} y_i \leq (1-\varepsilon)W \quad \forall t > 0 && (2) \\
& 0 \leq x_i \leq \min\{r_{max}(i), \lceil \varepsilon W \rceil\} \qquad \text{for} \quad 1 \leq i \leq n \\
& 0 \leq y_i \leq r_{max}(i) - \lceil \varepsilon W \rceil \qquad \text{for} \quad J_i \in \mathcal{J}_{wide}
\end{aligned}
$$

.

Fig. 1 The linear program LP_{FBA}

Theorem 2 *Proper_*FSAP *is a polynomial-time* $(\frac{5}{4} + \varepsilon)$*-approximation algorithm for any instance of* FSAP *in which the input graph is proper.*

We prove the theorem using the following results. First, consider the case in which a β fraction of the optimal profit comes from intervals in $\mathcal{J}_{wide}$.

Lemma 3 *If there exists a solution of* FSAP *for* $\mathcal{J}$ *in which the profit from the intervals in* $\mathcal{J}_{wide}$ *that are assigned at least* εW *colors is at least* $\beta OPT_{\text{FSAP}}(\mathcal{J})$*, then a solution of profit at least* $\beta OPT_{\text{FSAP}}(\mathcal{J})$ *can be found in polynomial time.*

Proof We consider only the intervals in $\mathcal{J}_{wide}$ and the set of feasible solutions S with the property that each interval in S is assigned at least εW colors. We can now add the requirement that the minimum number of colors assigned to any wide interval i is at least $r'_{min}(i) = \lfloor \varepsilon W \rfloor + 1$. For such instances, an optimal subset of (wide) intervals can be found in polynomial time using dynamic programming (see Lemma 4.3.1. in [34]).[7] By our assumption, the value of this solution is at least $\beta OPT_{\text{FSAP}}(\mathcal{J})$. □

Next, we consider the complement case. In Fig. 1 we give the linear program used by Algorithm $\mathcal{A}_{Narrow\_Color}$. For each $J_i \in \mathcal{J}_{narrow}$ the linear program has a variable x_i indicating the number of colors assigned to J_i. For each $J_i \in \mathcal{J}_{wide}$, the linear program has two variables: x_i and y_i, where $x_i + y_i$ is the number of colors assigned to J_i; y_i gives the number of assigned colors "over" the first $\lceil \varepsilon W \rceil$ colors.

Constraint (1) bounds the total profit from the extra allocation for each interval $J_i \in \mathcal{J}_{wide}$. The constraints (2) bound the total number of colors used at any time $t > 0$ by $(1-\varepsilon)W$. We note that the number of constraints in (2) is polynomial in $|\mathcal{J}|$, since we only need to consider the "interesting" points of time t, when $t = s_i$ for some interval $J_i \in \mathcal{J}$, i.e., we have at most n constraints.

Lemma 4 *For any* $\varepsilon > 0$*, there is an integral solution of* LP_{FBA} *of total profit at least* $(1-2\varepsilon) OPT_{\text{FSAP}}(\mathcal{J})$*, which can be found in polynomial time.*

[7] See also Lemma 5 in [31].

Proof Let $\bar{x}^*, \bar{y}^*$ be an optimal (fractional) solution of LP_{FBA}. Note that, by possibly moving value from y_i^* to x_i^*, we can always find such a solution in which for any $J_i \in \mathcal{J}_{wide}$, if $y_i^* > 0$ then $x_i^* = \lceil \varepsilon W \rceil$.

Let S_w be the set of intervals in $\mathcal{J}_{wide}$ for which $y_i^* > 0$. Consider now the solution $\bar{x}^*, \bar{z}$, where $z_i = \lfloor y_i^* \rfloor$, for all $J_i \in S_w$. We bound the loss due to the rounding down.

$$\begin{aligned}\sum_{J_i \in S_w} p_i(x_i^* + \lfloor y_i^* \rfloor) &\geq \sum_{J_i \in S_w} p_i(x_i^* + y_i^* - 1) \\ &\geq \sum_{J_i \in S_w} p_i(x_i^* + y_i^*)(1 - \tfrac{1}{x_i^*}) \\ &\geq \sum_{J_i \in S_w} p_i(x_i^* + y_i^*)(1 - \tfrac{1}{\lceil \varepsilon W \rceil})\end{aligned}$$

The last inequality follows from the fact that $x_i^* = \lceil \varepsilon W \rceil$, for any $J_i \in S_w$. We also note that an optimal solution for LP_{FBA} is at least $(1-\varepsilon)OPT_{\text{FBA}}(\mathcal{J}) \geq (1-\varepsilon)OPT_{\text{FSAP}}(\mathcal{J})$, as LP_{FBA} uses at most $(1-\varepsilon)W$ colors. Thus, for $W > 1/\varepsilon^2$, we have that the total profit after rounding the y_i^* values is at least $(1-2\varepsilon)OPT_{\text{FBA}}(\mathcal{J})$.

Now, consider the linear program LP_{round}, in which we fix $b_i = \lfloor y_i^* \rfloor$, for $J_i \in S_w$. We have that $\bar{x}^*$ is a feasible (fractional) solution of LP_{round} with profit at least

$$\begin{aligned}(LP_{round}): \quad \max \quad & \sum_{i=1}^{n} p_i x_i \\ \text{s.t.} \quad & \sum_{\{J_i \in \mathcal{J} : t \in J_i\}} x_i + \sum_{\{J_i \in S_w : t \in J_i\}} b_i \leq \lfloor (1-\varepsilon)W \rfloor \quad \forall t > 0 \\ & 0 \leq x_i \leq \min\{r_{max}(i), \lceil \varepsilon W \rceil\} \qquad \text{for } 1 \leq i \leq n\end{aligned} \tag{3}$$

$(1-2\varepsilon)OPT_{\text{FBA}}(\mathcal{J}) - \sum_{J_i \in S_w} p_i b_i$. The rows of the coefficient matrix of LP_{round} can be permuted so that the time points associated with the rows form an increasing sequence. In the permuted matrix each column has consecutive 1's, thus this matrix is *totally unimodular* (*TU*) (see, e.g., [28,32]). It follows that we can find in polynomial time an integral solution, $\bar{x}_I^*$, of the same total profit. Thus, the integral solution $\bar{x}_I^*, \bar{z}$ satisfies the lemma. □

The next result is shown in [31].

Lemma 5 *Given the subset S' of intervals colored by c in Step 2 of $\mathcal{A}_{Narrow\_Color}$, Algorithm ProperToCircular finds in polynomial time a valid coloring c' of S' satisfying $P_{c'}(S') = P_c(S')$.*

Lemma 6 *Consider the sets S'_w and S'_n defined in Step 5 of $\mathcal{A}_{Narrow\_Color}$ and the coloring c' after Step 6. Then the following hold. (i) $P_{c''}(S'_w) \geq \frac{3}{4} P_{c'}(S'_w)$, and (ii) $P_{c''}(S'_n) = P_{c'}(S'_n)$.*

Proof We first show property (i). Given the contiguous circular coloring c' after Step 6 of $\mathcal{A}_{Narrow\_Color}$, consider first a randomized algorithm which selects uniformly at random an index $\ell \in \{0, \ldots, W' - 1\}$ for "cutting" the circle. Let $J_i \in S'_w$ be a wide

interval colored by c', and let $d = |c'(J_i)|$ be the number of colors assigned to J_i after Step 6. We assume that d is even (for odd d the bound can only improve). We distinguish between three cases.

(a) If $\ell \notin c(J_i)$ then $|c''(J_i)| = d$, since the coloring c' of J_i remains contiguous after cutting the circle.
(b) $\ell = f_i + m$ and $0 \leq m \leq \frac{d}{2}$. Then $|c''(J_i)| = d - m$.
(c) $\ell = f_i + m$ and $\frac{d}{2} < m < d$. Then $|c''(J_i)| = m$.

Now, the probability that m takes any value in $\{0, \ldots, W'\}$ is $\frac{1}{W'}$. It suffices to show that $E[|c''(J_i)|] \geq \frac{3d}{4}$. By the above discussion,

$$\begin{aligned} E[|c''(J_i)|] &= \frac{1}{W'}\left(\sum_{m=1}^{d/2}(d-m) + \sum_{m=d/2+1}^{d-1} m + (W'-d+1)d\right) \\ &= \frac{1}{W'}(W'd - \frac{d^2}{4}) \\ &= d - \frac{d^2}{4W'} \geq \frac{3d}{4} \end{aligned}$$

Using linearity of expectation we have that $E[P_{c''}(S'_w)] \geq \frac{3}{4}P_{c'}(S'_w)$. Therefore there exists an index $\ell \in \{0, \ldots, W'\}$ for which property (i) is satisfied. Such an index will be found in Step 14 of $\mathcal{A}_{Narrow\_Color}$.

To show that property (ii) holds, we note that (in Steps 19–20) Algorithm $\mathcal{A}_{Narrow\_Color}$ assigns $|c'(J_i)|$ colors to any $J_i \in S'_n$. Thus, in the resulting non-circular contiguous coloring, c'', we have $|c''(J_i)| = |c'(J_i)|, \forall\, J_i \in S'_n$. Observe that c'' is a valid coloring, since at most one interval (in particular, an interval $J_i \in S'_n$) can be assigned the color ℓ_{good} at any time $t > 0$. It follows, that the intervals J_i in S'_n that contain ℓ_{good} in $c'(J_i)$ form an independent set. Since for any $J_i \in S'_n$ $r_{max}(i) \leq \lceil \varepsilon W \rceil$, we can assign to all of these intervals the same set of new colors in the range $\{W' + 1, \ldots, W\}$. □

Proof of Theorem 2: Let $0 < \beta < 1$ be the parameter used by Proper_FSAP. For a correct guess of $OPT_{\text{FSAP}}(\mathcal{J})$ and a fixed value of $\varepsilon > 0$, let $\hat{\varepsilon} = \varepsilon/3$.

(i) If there is an optimal solution in which the profit from the intervals in $\mathcal{J}_{wide}$ that are assigned at least εW colors is at least $\beta OPT_{\text{FSAP}}(\mathcal{J})$ then, by Lemma 3, Proper_FSAP finds in Step 2 a solution of profit at least $\beta OPT_{\text{FSAP}}(\mathcal{J})$ in polynomial time.
(ii) Otherwise, consider the coloring c'' output by $\mathcal{A}_{Narrow\_Color}$. Using Lemma 6, we have that

$$\frac{P_{c'}(\mathcal{J})}{P_{c''}(\mathcal{J})} = \frac{P_{c'}(S'_w) + P_{c'}(S'_n)}{\frac{3P_{c'}(S'_w)}{4} + P_{c'}(S'_n)} \leq \max_{0<x\leq\beta} \frac{1}{\frac{3x}{4} + 1 - x} = \frac{4}{4-\beta}$$

Now, by Lemma 4, taking $\varepsilon = \hat{\varepsilon}$, the (non-contiguous) coloring obtained in Step 2 of $\mathcal{A}_{Narrow\_Color}$ has a profit at least $(1 - 2\hat{\varepsilon})OPT_{\text{FSAP}}(\mathcal{J})$. Using Lemma 5, after applying the rounding in Step 6 of $\mathcal{A}_{Narrow\_Color}$ with $\hat{\varepsilon}$ we have that $P_{c'}(\mathcal{J}) \geq (1 - 3\hat{\varepsilon})OPT_{\text{FSAP}}(\mathcal{J}) = (1 - \varepsilon)OPT_{\text{FSAP}}(\mathcal{J})$.

The theorem follows by taking $\beta = \frac{4}{5}$.

For the running time of Proper_FSAP, we first note that, by Lemma 3, a solution consisting of wide intervals (only) of profit at least $\beta OPT_{\text{FSAP}}(\mathcal{J})$ can be found in polynomial time. For the complementary case, we now show that the running time of $\mathcal{A}_{Narrow\_Color}$ is polynomial as well. Indeed, LP_{FBA} has a polynomial number of constraints. Also, by Lemma 5, the contiguous circular coloring c' is found in $O(n \log n)$ steps. Then, converting c' to the contiguous (non-circular) coloring c'' requires to find the best way to cut the circle. Since the possible number of such cutting points is $O(1)$, the index ℓ_{good} is found in polynomial time. We conclude that Proper_FSAP has a polynomial running time. □

5 Complexity of FSAP on Uniform Instances

We now consider *uniform* instances of FSAP, in which $r_{max}(i) = \texttt{Max}$, for some $1 \leq \texttt{Max} \leq W$, and $p_i = 1$, for all $1 \leq i \leq n$. Let $k = \lceil W/\texttt{Max} \rceil$.

We first prove that if $k = W/\texttt{Max}$ (i.e., W is an integral multiple of Max) then FSAP can be solved optimally by finding a maximum k-colorable subgraph in G, and assigning to each interval in this subgraph Max contiguous colors. Recall that a maximum k-colorable subgraph of an interval graph can be found in polynomial time using dynamic programming as follows. We sort the intervals by their left endpoints. Then, we scan the intervals one by one while maintaining a maximum k-colorable subgraph of the scanned intervals. When an interval is scanned it is added to the maintained subgraph. If the resulting subgraph is not k-colorable then the interval with the maximum right endpoint is removed from the subgraph. It is easy to see that the removal of this interval makes the subgraph k-colorable.

In contrast to this case if $k > W/\texttt{Max} \geq 1$, then we show that FSAP is NP-Hard, and give a polynomial approximation scheme to solve such instances.

Lemma 7 *An* FSAP *instance where W is a multiple of* Max*, and for all $1 \leq i \leq n$, $p_i = 1$ and $r_{max}(i) = \texttt{Max}$, can be solved exactly in polynomial time.*

Proof Consider an FBAP instance where for all $1 \leq i \leq n$, $r_{max}(i) = \texttt{Max}$, and W is multiple of Max. We claim that there exists an optimal solution to this FBAP instance in which each job gets either 0 or Max colors. To see that consider the behavior of the algorithm Paging_FBA on such an instance, i.e., in which $r_{min}(i) = 0$ and $r_{max}(i) = \texttt{Max}$. Note that when we start the algorithm |AVAIL| is a multiple of Max and thus when the first (left) endpoint s_1 is considered the job J_1 is allocated Max colors and |AVAIL| remains a multiple of Max. Assume inductively that as the endpoints are scanned all the jobs that were allocated colors up to this endpoint were allocated Max colors and that |AVAIL| is a multiple of Max. If the current scanned endpoint is a right endpoint e_i then by the induction hypothesis either 0 or Max colors

are added to AVAIL and nothing else is changed. Suppose that the scanned endpoint is a left endpoint s_i. If $|\texttt{AVAIL}| > 0$ then Max colors from AVAIL are allocated to J_i. Otherwise, J_i may be allocated colors that were previously allocated to another job J_ℓ, for $\ell < i$. However, in this case because $r_{min}(\ell) = 0$ all the Max colors allocated to J_ℓ will be moved to J_i, and the hypothesis still holds.

The optimal way to assign either 0 or Max colors to each job is given by computing the maximum $k = (W/\texttt{Max})$-colorable subgraph of G and assigning Max colors to each interval in the maximum k-colorable subgraph. Since the assignment of these colors can be done contiguously it follows that this is also the optimal solution to the respective FSAP instance. □

In the Appendix, we give a proof of hardness in case $k > W/\texttt{Max}$.

Theorem 8 FSAP *is strongly NP-hard even if for all* $1 \leq i \leq n$ $r_{max}(i) = 3$*, and* W *is not divisible by 3.*

6 Approximation Algorithms for Uniform FSAP

6.1 A $\frac{2k}{2k-1}$-approximation Algorithm

Suppose that $k > W/\texttt{Max}$. (Note that since $W > \texttt{Max}$, we have $k \geq 2$.) We describe below Algorithm $\mathcal{A}_{MaxSmall}$ for uniform FSAP instances. It yields solutions that are close to the optimal as $k = \lceil \frac{W}{\texttt{Max}} \rceil$ gets large. Initially, $\mathcal{A}_{MaxSmall}$ solves optimally FBAP on the input graph G. Let G' be the support graph for this solution (i.e., G' is the subgraph of G induced by the intervals that are colored in this solution). $\mathcal{A}_{MaxSmall}$ proceeds by generating a feasible solution for FSAP as follows. Consider the set of intervals $J_i \in G'$ for which $|c(J_i)| = \texttt{Max}$. Let G_2 be the subgraph of G' induced by this set of intervals. Observe that the graph G_2 is $(k-1)$-colorable, since there is no clique of size k in G_2 (as it would require $k\texttt{Max} > W$ colors). Consequently, each interval in G_2 can be assigned a set of Max contiguous colors, using a total of $(k-1)\texttt{Max}$ colors. $\mathcal{A}_{MaxSmall}$ then finds a maximum independent set of intervals in the remaining subgraph $G_1 = G' \setminus G_2$, and colors contiguously each interval in this set using the remaining W mod Max colors (see the pseudocode in Algorithm 6.1). Recall that an independent set is a maximum 1-colorable graph and thus it can be computed using the algorithm given above.

Theorem 9 *For any uniform instance of* FSAP*,* $\mathcal{A}_{MaxSmall}$ *yields an optimal solution for* FSAP *if* $k = 2$*, and a* $\frac{2k}{2k-1}$*-approximation for any* $k \geq 3$*.*

The proof of Theorem 9 uses the next lemma. Recall that $G_1 \subseteq G'$ is the subgraph induced by the intervals J_i for which $|c(J_i)| < \texttt{Max}$.

Lemma 10 *The subgraph* $G_1 \subseteq G'$ *is proper and* 2*-colorable, and for each* $J_i \in G_1$ *we have* $|c(J_i)| = W$ mod Max*.*

Proof We first note that if G_1 is not proper, then there exist two intervals, J_i and J_j, such that J_j is completely contained in J_i. By the way Paging_FBA proceeds, when it

Algorithm 6.1 $\mathcal{A}_{MaxSmall}(\mathcal{J}, \texttt{Max}, W)$

1: Find an optimal solution for FBAP on G, the interval graph of $\mathcal{J}$, using Algorithm Paging_FBA.
2: Let $S \subseteq \mathcal{J}$ be the solution set of intervals, and $G' \subseteq G$ the support graph of S.
3: Let $G_2 \subseteq G'$ be the subgraph induced by the intervals J_i for which $|c(J_i)| = \texttt{Max}$.
4: Let $G_1 \subseteq G'$ be the subgraph induced by the rest of the intervals, i.e., intervals J_i for which $|c(J_i)| < \texttt{Max}$.
5: Scan the intervals J_i in G_2 from left to right and color contiguously each interval with the lowest available Max colors.
6: Let $r = W \bmod \texttt{Max}$.
7: Let I be a maximum independent set in G_1.
8: Color each interval in I contiguously in the r colors $(k-1)\texttt{Max}+1, \ldots, W$.
9: Return the coloring of the intervals in $G_1 \cup G_2$.

colors J_j, some colors that were assigned to J_i should be assigned to J_j, until either $|c(J_j)| = \texttt{Max}$, or $|c(J_i)| = 0$. Since none of the two occurs - a contradiction.

We now show that G_1 is 2-colorable, and that for each $J_i \in G_1$ we have $|c(J_i)| = W \bmod \texttt{Max}$. Throughout the proof, we assume that there are n_1 intervals in G_1 sorted by their starting points, and numbered $1, 2, \ldots, n_1$, i.e., $s_1 < s_2, < \cdots < s_{n_1}$. For any $t \in [s_1, e_{n_1})$, say that t is *tight* if

$$\sum_{\{J_\ell \in \mathcal{J} : t \in J_\ell\}} |c(J_\ell)| = W.$$

Let $\mathcal{T}$ denote the set of tight time points. To keep a discrete set of such points, we consider only tight points t which are also the start-times of intervals, i.e., $t = s_i$ for some $1 \leq i \leq n$. We note that every interval $J_i \in G_1$ must contain at least one tight time point (otherwise, Paging_FBA would assign more colors to J_i). We complete the proof using the next claim. □

Claim 1 *Every time point $t \in \mathcal{T}$ is contained in exactly one interval $J_i \in G_1$.*

We show that Claim 1 implies that G_1 is 2-colorable. Assume that there exists in G_1 a clique of at least 3 intervals. Let J_a, J_b, J_c be three ordered intervals in this clique. We note that there is no $t \in J_b$ that is not contained in either J_a or J_c. However, J_b must contain a tight time point. Contradiction to Claim 1.

Claim 1 also implies that for each $J_i \in G_1$ we have $|c(J_i)| = W \bmod \texttt{Max}$. Consider such an interval J_i. It must contain a tight time point. Since this tight time point is not contained in any other interval in G_1, we must have $|c(J_i)| = W \bmod \texttt{Max}$.

Proof of Claim 1: First, note that since W is not a multiple of Max every tight time point has to be contained in at least one interval $J_i \in G_1$. We prove that it cannot be contained in more than one such interval by induction. Let t_1 be the earliest time point in $\mathcal{T}$. Since each $J_i \in G_1$ contains a tight time point, $t_1 \in J_1$. If $t_1 < s_2$ then clearly the claim holds for t_1. Suppose that $t_1 \geq s_2$. In this case, since there is at least one available color in $[s_1, s_2)$, and since $e_2 > e_1$, Algorithm Paging_FBA would assign at least one additional color to J_1 rather than to J_2. A contradiction.

For the inductive step, let $i > 1$, and consider $t_i \in \mathcal{T}$. Assume that the claim holds for $t_{i-1} \in \mathcal{T}$, and let $t_{i-1} \in J_\ell$. Since t_{i-1} is not contained in any other interval in G_1,

and since it is tight, we must have $|c(J_\ell)| = W \bmod \texttt{Max}$. To obtain a contradiction assume that t_i is contained in more than one interval in G_1. At least one of these intervals must be $J_{\ell+1}$ (since $t_i > e_{\ell-1}$). If t_i is not contained in J_ℓ then clearly, by our assumption, t_i must also be contained in $J_{\ell+2}$. However, even if $t_i \in J_\ell$, since $|c(J_\ell)| = W \bmod \texttt{Max}$, in order for t_i to be tight, it must be contained also in $J_{\ell+2}$. In this case, since there is at least one available color in $[s_{\ell+1}, s_{\ell+2})$, and since $e_{\ell+2} > e_{\ell+1}$, Algorithm Paging_FBA would assign at least one additional color to $J_{\ell+1}$, rather than to $J_{\ell+2}$. A contradiction. □

We are ready to show the performance ratio of Algorithm $\mathcal{A}_{MaxSmall}$.

Proof of Theorem 9: We handle separately two cases.

(i) $k \geq 3$. Consider the graphs G' and G_1, G_2, as defined in Steps 4, and 3 of $\mathcal{A}_{MaxSmall}$, respectively. Let $OPT_{\text{FSAP}}(G)$ and $\mathcal{A}(G)$ be the value of an optimal solution and the solution output by $\mathcal{A}_{MaxSmall}$, respectively, for an input graph G.
Since G_1 is 2-colorable, $\mathcal{A}(G) \geq OPT_{\text{FBA}}(G) - \frac{1}{2}(W \bmod \texttt{Max}) \cdot |G_1|$. To get the approximation ratio $\frac{2k}{2k-1}$, we need to show that $(W \bmod \texttt{Max}) \cdot |G_1| \leq \frac{1}{k} OPT_{\text{FBA}}(G)$. Let $\lambda\texttt{Max} = W \bmod \texttt{Max}$. Note that $0 < \lambda < 1$. In fact, we prove a slightly better bound, as we show that

$$\frac{\lambda \texttt{Max}|G_1|}{OPT_{\text{FBA}}(G)} \leq \frac{\lambda}{k-1+\lambda} < \frac{1}{k}. \tag{4}$$

Before we prove the inequality we show how it implies the approximation ratio. Clearly, $OPT_{\text{FSAP}}(G) \leq OPT_{\text{FBA}}(G)$. By inequality (4)

$$\begin{aligned}
\mathcal{A}(G) &\geq OPT_{\text{FBA}}(G) - \frac{1}{2}(W \bmod \texttt{Max}) \cdot |G_1| \\
&\geq OPT_{\text{FBA}}(G)\left(1 - \frac{1}{2} \cdot \frac{\lambda}{k-1+\lambda}\right) \\
&= OPT_{\text{FBA}}(G)\frac{(2k-2+\lambda)}{2(k-1+\lambda)} > OPT_{\text{FBA}}(G)\frac{2k-1}{2k}.
\end{aligned}$$

Thus, $\frac{OPT_{\text{FSAP}}(G)}{\mathcal{A}(G)} < \frac{2k}{2k-1}$.
We now prove inequality (4). To obtain a contradiction, assume that inequality (4) does not hold. By Lemma 10 we have

$$OPT_{\text{FBA}}(G) = (W \bmod \texttt{Max}) \cdot |G_1| + \texttt{Max} \cdot |G_2|,$$

where $|G_2| = |G'| - |G_1|$. We get

$$\begin{aligned}
\frac{(W \bmod \texttt{Max}) \cdot |G_1|}{OPT_{\text{FBA}}(G)} &= \frac{\lambda \texttt{Max}|G_1|}{\texttt{Max}|G'| - \texttt{Max}(1-\lambda)|G_1|} = \frac{\lambda|G_1|}{|G'| - (1-\lambda)|G_1|} \\
&> \frac{\lambda}{k-1+\lambda}.
\end{aligned}$$

This implies $k|G_1| > |G'|$, and thus

$$OPT_{\text{FBA}}(G) = \lambda \texttt{Max}|G_1| + \texttt{Max}(|G'| - |G_1|) = \texttt{Max}|G'| - (1-\lambda)\texttt{Max}|G_1|$$
$$< (1 - \frac{1}{k}(1-\lambda))\texttt{Max}|G'| = \frac{k-1+\lambda}{k} \cdot \texttt{Max}|G'|.$$

Note than G', the support graph of the solution obtained by Paging_FBA, is k-colorable, since it cannot contain any clique of size $k+1$. Indeed, such a clique can have at most 2 intervals from G_1, and at least $k-1$ intervals from G_2, and thus requires more than W colors (since Algorithm Paging_FBA assigns W mod Max colors to each interval in G_1). We can use the k coloring to obtain a solution of the FBAP instance, by assigning Max colors to intervals in the $k-1$ largest color classes, and W mod Max to the remaining color class. Thus, $OPT_{\text{FBA}}(G) \geq \frac{1}{k} \cdot \lambda \texttt{Max}|G'| + \frac{k-1}{k} \cdot \texttt{Max}|G'| = \frac{k-1+\lambda}{k} \cdot \texttt{Max}|G'|$. A contradiction.

(ii) $k = 2$. Then a polynomial time algorithm $\mathcal{A}$ is obtained by modifying Algorithm $\mathcal{A}_{MaxSmall}$. We first compute the graphs G' and G_1, G_2, as defined in Steps 4, and 3 of $\mathcal{A}_{MaxSmall}$, respectively. Note that no two intervals in G_2 intersect, since coloring two intersecting intervals in G_2 would require $2\texttt{Max} > W$ colors. It follows that G_2 is an independent set. We claim that G' is 2-colorable. Suppose that this is not the case, then G' must have a clique of size at least 3. Since G_1 is 2-colorable and G_2 is an independent set, the only way to have such a clique is if an interval from G_2 intersects an intersection point of two intervals of G_1, say J_{i-1} and J_i. However, in this case a tight time point is contained in two intervals in G_1 contradicting Claim 1.

We color G' in two "shades" a and b. (We call it *shades* rather than *colors* to distinguish from the colors that are used for allocation.) Let $r = W$ mod Max. We now assign the colors as follows: if an interval from G_1 has shade a color it in the colors $1, \ldots, r$, otherwise color it in the colors $\texttt{Max}+1, \ldots W$. If an interval from G_2 has shade a color it in the colors $1, \ldots, \texttt{Max}$, otherwise color it in the colors $r+1, \ldots W$. The shades for the intervals can be determined by any 2-coloring of G'. It is easy to verify that no two intersecting intervals are colored in the same color. It follows that in this case $\mathcal{A}(G) = OPT_{\text{FBA}}(G) = OPT_{\text{FSAP}}(G)$.

□

6.2 An Approximation Scheme

We now describe a PTAS for uniform instances of FSAP. Denote by $OPT_{\text{FSAP}}(\mathcal{J})$ the value of an optimal solution for an instance $\mathcal{J}$ of FSAP. Fix $\varepsilon \in (0, \frac{1}{2})$. W.l.o.g., we may assume that $W > 4/\varepsilon^2$, else W is a constant, and in this case FSAP can be solved optimally in polynomial time (see [31]).

Let $\mathcal{J}$ be a uniform input for FSAP. The scheme handles separately two subclasses of inputs, depending on the value of Max. First, we consider the case where Max is *large* relative to W, or more precisely $k = \lceil W/\texttt{Max} \rceil \leq 1/\varepsilon$. In this case we partition the colors into $O(1/\varepsilon^2)$ strips of contiguous colors, each of size at most $\lfloor \varepsilon^2 W/4 \rfloor \geq 1$. Note that by our assumption $\lfloor \varepsilon^2 W/4 \rfloor \leq \lfloor \varepsilon \texttt{Max}/4 \rfloor$. We consider only

feasible solutions that for each strip assign either all the colors in the strip or none of the colors in the strip to any job, and find an optimal solution among these solutions. Since the number of strips is $O(1/\varepsilon^2)$, this can be done in polynomial time using dynamic programming, as shown in [31]. In the next lemma we prove that the profit of this solution is at least $(1-\varepsilon)OPT_{\text{FSAP}}(\mathcal{J})$.

Lemma 11 *For any uniform instance $\mathcal{J}$ of* FSAP *where $\lceil W/\texttt{Max}\rceil \leq 1/\varepsilon$, there exists a feasible solution of total profit is at least $(1-\varepsilon)OPT_{\text{FSAP}}(\mathcal{J})$, where each job is assigned complete strips of colors (possibly zero).*

Proof Given an optimal solution for a uniform input $\mathcal{J}$, let S be the subset of intervals J_i for which $|c(J_i)| > 0$, and let G_S be the support graph for S (i.e., the subgraph of the original interval graph induced by the intervals in S). We show how to convert this solution to a solution whose total profit is at least $(1-\varepsilon)OPT_{\text{FSAP}}(\mathcal{J})$ in which each interval $J_i \in S$ is assigned complete strips of colors (possibly zero), and each interval $J \notin S$ is not assigned any color.

Using the above partition of the colors to strips, we have $1 \leq N \leq \lceil \frac{8}{\varepsilon^2} \rceil$ strips. Denote by C_j the subset of colors of strip j. We obtain the *strip structure* for the solution as follows. Let $S_j \subseteq S$ be the subset of intervals with colors in strip j, i.e., $S_j = \{J_i | c(J_i) \cap C_j \neq \emptyset\}$.

We now define the coloring c' of the converted solution. Initialize for all $J_i \in S$, $c'(J_i) = \emptyset$.

(i) For all $1 \leq j \leq N$, find in G_S a maximum independent set, $\mathcal{I}_j$ of intervals $J_i \in S_j$. For any $J_i \in \mathcal{I}_j$, assign to J_i all colors in C_j, i.e., $c'(J_i) = c'(J_i) \cup C_j$.
(ii) For any $J_i \in S$, if $|c'(J_i)| > \texttt{Max}$ then omit from the coloring of J_i a consecutive subset of strips, starting from the highest $1 \leq j \leq N$, such that $C_j \subseteq c'(J_i)$, until for the first time $|c'(J_i)| \leq \texttt{Max}$.

We show below that the above *strip coloring* for S is feasible, and that the total profit from the strip coloring is at least $(1-\varepsilon)OPT_{\text{FSAP}}(J)$. To show feasibility, note that if $J_i \in S_j$ and $J_i \in S_{j+2}$ then, because the coloring is contiguous J_i is allocated all the colors in S_{j+1}; thus, any maximum independent set $\mathcal{I}_{j+1}$ will contain J_i, since it has no conflicts with other jobs in S_{j+1}. It follows that if a job J_i is allocated colors in more than two consecutive strips, i.e., $J_i \in S_j \cap S_{j+1} \cap \cdots \cap S_{j+\ell}$, for $\ell > 1$, then $J_i \in \mathcal{I}_{j+1} \cap \cdots \cap \mathcal{I}_{j+\ell-1}$. Thus, each interval $J_i \in S$ will be assigned in step (i) a consecutive set of strips. Hence, c' is a contiguous coloring. In addition, after step (ii), for all $J_i \in S$ we have that $|c'(J_i)| \leq \texttt{Max}$.

Now, consider the profit of the strip coloring. We note that after step (i), the total profit of c' is at least $OPT_{\text{FSAP}}(J)$. This is because for each strip j, $|C_j| \cdot |\mathcal{I}_j|$ is an upper bound on the profit that can be obtained from this strip. (This is true since, by Lemma 7, this is the optimal profit even when all the colors C_j can be assigned to every interval in S_j.) We show that the impact of reducing the number of colors in step (ii) is small. We distinguish between two type of intervals in S.

(a) Intervals J_i for which $|c(J_i)| \geq (1-\varepsilon/2)\texttt{Max}$. Since coloring c is valid, it follows that $|c'(J_i)|$ is reduced in step (ii) only if before this step $|c'(J_i)| > |c(J_i)|$. Consider all the strips that contain colors assigned to J_i in the original coloring

c. Note that in all such strips, except at most two, no colors are assigned to any interval that intersects J_i. Thus, $|c'(J_i)|$ is reduced in step (ii) by at most $2\lfloor\frac{\varepsilon \texttt{Max}}{4}\rfloor \leq (\varepsilon/2)\texttt{Max}$. Since $|c(J_i)| \geq (1-\varepsilon/2)\texttt{Max}$, we have that after step (ii), $|c'(J_i)| \geq |c(J_i)| - (\varepsilon/2)\texttt{Max} \geq (1-\varepsilon)\texttt{Max}$.

(b) For any interval J_i for which $|c(J_i)| < (1-\varepsilon/2)\texttt{Max}$, since after step (i) $|c'(J_i)| \leq |c(J_i)| + 2\lfloor\frac{\varepsilon \texttt{Max}}{4}\rfloor$, we have that $|c'(J_i)| \leq \texttt{Max}$. Thus, no colors are omitted from J_i in step (ii).

From (a) and (b), we have that the total profit from the strip coloring satisfies $OPT'_{\text{FSAP}}(\mathcal{J}) \geq (1-\varepsilon)OPT_{\text{FSAP}}(\mathcal{J})$. □

Now, consider the case where Max is *small*, i.e., $k = \lceil W/\texttt{Max}\rceil > 1/\varepsilon$. In this case we consider just $(k-1)\texttt{Max}$ consecutive colors and ignore the remainder up to εW colors. Let $OPT_{\text{FSAP}(W)}(\mathcal{J})$ denote the value of an optimal solution for instance $\mathcal{J}$ of FSAP with W colors, and recall that $k = \lceil W/\texttt{Max}\rceil$. Since $(k-1)\texttt{Max} < W < k\texttt{Max}$, $OPT_{\text{FSAP}((k-1)\texttt{Max})}(\mathcal{J}) < OPT_{\text{FSAP}(W)}(\mathcal{J}) < OPT_{\text{FSAP}(k\texttt{Max})}(\mathcal{J})$.

Recall that by Lemma 7 when the number of colors W is a multiple of Max we can find an optimal solution in polynomial time, and that by the proof of the lemma the value of this optimal solution is Max times the size of the maximum $W/\texttt{Max}$-colorable subgraph of G. Thus, the value of $OPT_{\text{FSAP}((k-1)\texttt{Max})}$ is Max times the size of the maximum $(k-1)$-colorable subgraph of G, and the value of $OPT_{\text{FSAP}(k\texttt{Max})}$ is Max times the size of the maximum k-colorable subgraph of G. Clearly, the ratio of the sizes of these subgraphs and thus the ratio of the two optimal values is bounded by $1-1/k > 1-\varepsilon$. It follows that $OPT_{\text{FSAP}((k-1)\texttt{Max})}(\mathcal{J}) > (1-\varepsilon)OPT_{\text{FSAP}(k\texttt{Max})}(\mathcal{J}) \geq (1-\varepsilon)OPT_{\text{FSAP}(W)}(\mathcal{J})$. Combining the results, we have

Theorem 12 *The above algorithm is a PTAS for uniform* FSAP *instances.*

Acknowledgements We thank Magnús Halldórsson and Viswanath Nagarajan for valuable discussions. We also thank two anonymous referees for many insightful comments on the paper.

A Hardness of FSAP for Uniform Instances

Proof of Theorem 8: The reduction is from of the 3-Exact-Cover (3XC) problem defined as follows. Given a universal set $U = \{e_1, \ldots, e_{3n}\}$ and a collection $S = \{S_1, \ldots, S_m\}$ of 3 element subsets of U, is there a sub-collection $S' \subseteq S$ such that each element of U occurs in exactly one member of S'. Note that $|S'| = n$, if such exists. Recall that Karp showed in his seminal paper [19] that 3XC is NP-Complete.

To simplify the presentation, we first assume that the intervals have different profits per allocated unit. For the reduction, we use several sets of intervals. One such set is shown in Fig. 2. It consists of 16 intervals whose lengths and relative positions are given in the figure. Assume that the profit of each of the intervals $9, \ldots, 16$ is higher than the sum of the profits of intervals $1, \ldots, 8$, and that the profit of each of the intervals 2 to 7 is higher than the profit of intervals 1 and 8.

Suppose that we are given two "banks" of contiguous colors to allocate to this set of intervals: one bank consists of four contiguous colors and one consists of three contiguous colors. Given that $r_{max}(i) = 3$, our first priority is to allocate three colors

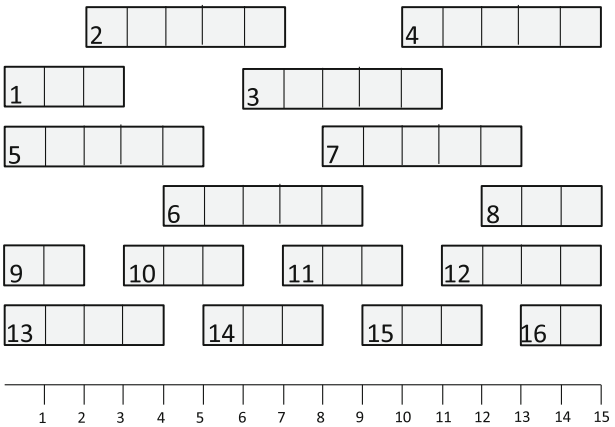

Fig. 2 The "two-choice" set of intervals

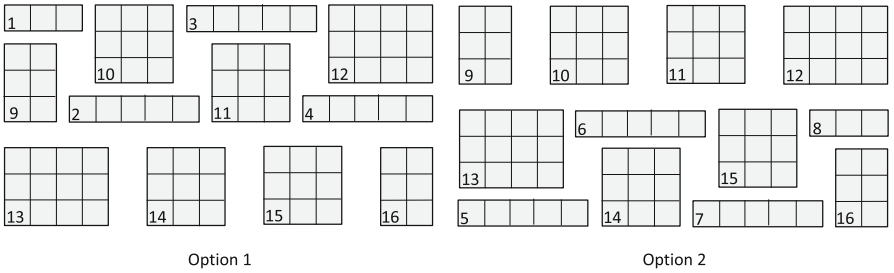

Fig. 3 Two possibilities for allocating the two banks of colors

to each interval in [9..16]. Assuming that three colors are indeed allocated to each interval in [9..16], note that any other interval can be allocated at most one color. This is since any other interval intersects two intervals from [9..16] at a point.

We say that two intervals from [1..8] *conflict* if both cannot be allocated colors simultaneously. Note that two such intervals conflict if both intersect two intervals from [9..16] at the same time point because only 7 colors are available. Define the conflict graph to be a graph over the vertices [1..8], where two vertices are connected if the respective intervals conflict. It is easy to see that the conflict graph is the path 1–5–2–6–3–7–4–8. Since the profit of intervals [2..7] is higher than the profit of intervals 1 and 8, the best strategy is to allocate one color to three of the intervals in [2..7] and one color to either interval 1 or 8. The only two possibilities for allocating the two banks of colors in such a way are illustrated in Fig. 3. Because of this property, we call this set of intervals a "two-choice" gadget. Note that, in the first option, the bank of 3 colors is unassigned at times: 5, 9 and 13, while in the second option, it is unassigned at times 3, 7 and 11.

We need to define also a pair of intervals called an "overlapping" pair of intervals, illustrated in Fig. 4. Note that to be able to allocate 3 colors to both intervals, we need one bank of 3 contiguous colors at time interval $[t_1, t_2+1)$ and another at time interval $[t_2, t_3)$; that is, we need both banks at time interval $[t_2, t_2+1)$.

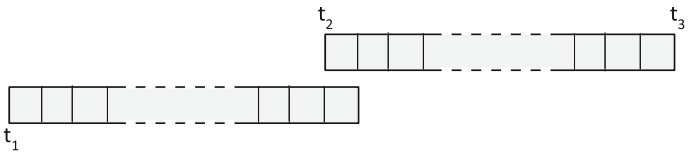

Fig. 4 Overlapping intervals

We now describe the reduction from the 3XC problem. For each set $S_i \in S$, associate a "two-choice" gadget. The "two choices" correspond to the decision whether to include S_i in the cover or not. For each element $e \in U$, and for each set S_i such that $e \in S_i$, we associate a pair of overlapping intervals, where the overlap is in one of the times in which the "two choice" gadget corresponding to set S_i has an unassigned bank of 3 colors. In addition, we define some extra intervals as detailed below.

Given a 3XC problem instance, set $W = 9m + 7$ and $r_{max} = 3$ for all intervals. Let $P = 8n + 45m$. In the reduction, we have 5 groups of intervals defined as follows.

(i) Left border: $3m + 3$ intervals $L_1, \ldots, L_{3m+3}$ each of profit P^2. For $i \in [1..3n]$, $L_i = [0, i)$, for $i \in [3n + 1..3m]$, $L_i = [0, 4n)$, and for $i \in [3m + 1..3m + 3]$, $L_i = [0, 4n + 15m)$.

(ii) Right border: $3m + 3$ intervals $R_1, \ldots R_{3m+3}$ each of profit P^2. For $i \in [1..3n]$, $R_i = [5n+45m+i, 8n+45m+1)$, for $i \in [3n+1..3m]$, $R_i = [4n+45m, 8n+45m+1)$, and for $i \in [3m + 1..3m + 3]$, $R_i = [4n + 30m, 8n + 45m + 1)$.

(iii) "two choice" gadgets: m copies of the "two choice" gadget, one for each set $S_i \in S$. The gadget associated with set S_i starts at time $4n + 15m + 15(i - 1)$ and its length is 15 time units. The profit of intervals 1 and 8 in each copy is 1, the profit of intervals 2 to 7 is 2, and the profit of intervals 9 to 16 is P^2.

(iv) element overlapping pairs: $3m$ overlapping pairs of intervals, one per occurrence of an element in a set. For each $S_i \in S$, let $S_i = \{e_a, e_b, e_c\}$, where $\{a, b, c\} \subseteq [1..3n]$. The respective three overlapping pairs are (1) $[a, 4n+15m+15(i-1)+3)$ and $[4n+15m+15(i-1)+2, 5n+45m+a)$, (2) $[b, 4n+15m+15(i-1)+7)$ and $[4n+15m+15(i-1)+6, 5n+45m+b)$, and (3) $[c, 4n+15m+15(i-1)+11)$ and $[4n+15m+15(i-1)+10, 5n+45m+c)$. The profit of each such interval is its length. Note that the profit of any pair of overlapping intervals is $5n+45m+1$.

(v) "filler" overlapping pairs: $3m$ overlapping pairs, three per set. For each $S_i \in S$, the respective three overlapping pairs are (1) $[4n, 4n + 15m + 15(i - 1) + 5)$ and $[4n + 15m + 15(i - 1) + 4, 4n + 45m)$, (2) $[4n, 4n + 15m + 15(i - 1) + 9)$ and $[4n + 15m + 15(i - 1) + 8, 4n + 45m)$, and (3) $[4n, 4n + 15m + 15(i - 1) + 13)$ and $[4n + 15m + 15(i - 1) + 12, 4n + 45m)$. The profit of each such interval is its length. Note that the profit of any pair of overlapping intervals is $45m + 1$.

□

Lemma 13 *The 3XC instance has an exact cover if and only if the associated* FSAP *instance has profit* $(18m+14+24m)P^2+7m+9n(5n+45m+1)+(9m-9n)(45m+1) = (42m + 14)P^2 + 405m^2 + 45n^2 + 16m$.

Proof Assume that the 3XC instance has an exact cover. We show an assignment of the intervals in the FSAP instance that achieves the desired profit. First, assign color

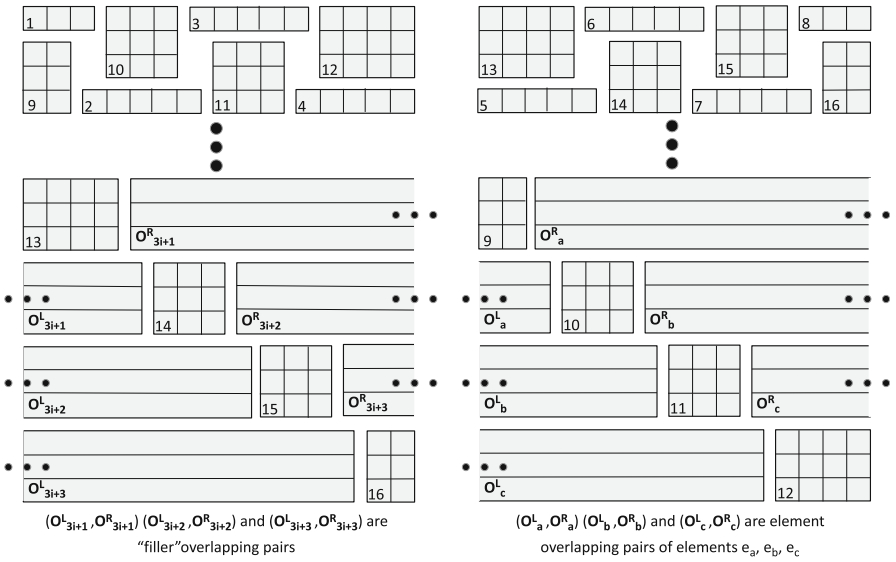

Fig. 5 Assigning intervals related to S_i

1 to L_{3m+2} and R_{3m+2} and colors 2, 3, 4 to L_{3m+3} and R_{3m+3}. Also assign colors 5, 6, 7 to L_{3m+1} and colors $9m+5, 9m+6, 9m+7$ to R_{3m+1}. Now, consider S_i, for $i \in [1..m]$. Assume that $0 \leq k < i$ sets S_j, for $j < i$ are in the cover. (See also Fig. 5.)

If S_i is not in the cover choose the first option for the "two choice" gadget associated with S_i. Namely, assign colors [1..4] to intervals [1..4] and [9..12] in the gadget, and for $j \in [13..16]$ assign colors $9i+3j-43, 9i+3j-42, 9i+3j-41$ to interval j. Also, assign colors $9i-4, 9i-3, 9i-2$ to $R_{3(n+i-k)-2}$; for $j \in \{2,3\}$, assign colors $9i+3j-7, 9i+3j-6, 9i+3j-5$ to $L_{3(n+i-k)+j-4}$ and $R_{3(n+i-k)+j-3}$, and assign colors $9i+5, 9i+6, 9i+7$ to $L_{3(n+i-k)}$. Finally, assign 3 colors to the 3 "filler" overlapping pairs corresponding to S_i as follows: colors $9i-4, 9i-3, 9i-2$ to interval $[4n+15m+15(i-1)+4, 4n+45m)$, colors $9i-1, 9i, 9i+1$ to intervals $[4n, 4n+15m+15(i-1)+5)$ and $[4n+15m+15(i-1)+8, 4n+45m)$, colors $9i+2, 9i+3, 9i+4$ to intervals $[4n, 4n+15m+15(i-1)+9)$ and $[4n+15m+15(i-1)+12, 4n+45m)$, and colors $9i+5, 9i+6, 9i+7$ to interval $[4n, 4n+15m+15(i-1)+13)$.

Suppose that S_i is in the cover. Let $S_i = \{e_a, e_b, e_c\}$, where $\{a,b,c\} \subseteq [1..3n]$. Choose the second option for the "two choice" gadget associated with S_i. Namely, assign colors [1..4] to intervals [5..8] and [13..16] in the gadget, and for $j \in [9..12]$ assign colors $9i+3j-31, 9i+3j-30, 9i+3j-29$ to interval j. The respective element overlapping pairs and the border intervals are assigned colors as follows: colors $9i-4, 9i-3, 9i-2$ to $[4n+15m+15(i-1)+2, 5n+45m+a)$ and R_a, colors $9i-1, 9i, 9i+1$ to L_a, $[a, 4n+15m+15(i-1)+3)$, $[4n+15m+15(i-1)+6, 5n+45m+b)$ and R_b, colors $9i+2, 9i+3, 9i+4$ to L_b, $[b, 4n+15m+15(i-1)+7)$, $[4n+15m+15(i-1)+10, 5n+45m+c)$ and R_c, and colors $9i+5, 9i+6, 9i+7$ to L_c and $[c, 4n+15m+15(i-1)+11)$.

Note that the assignment is valid, that is, no two overlapping intervals are assigned the same color. To calculate the total profit of the assignment note that L_{3m+1}, L_{3m+3}, R_{3m+1} and R_{3m+3} are each assigned 3 colors and L_{3m+2}, R_{3m+2} are assigned a single color. This contributes $14P^2$ to the profit. Also, intervals [9..16] in all the "two choice" gadgets are assigned 3 colors, and either intervals [1..4] or [5..8] are assigned a single color. This contributes $24mP^2+7m$ to the profit. Since we start from a cover, all the element overlapping pairs as well as the corresponding left and right border intervals are assigned 3 colors each. This contributes $3\cdot 6nP^2+3\cdot 3n(5n+45m+1)$ to the profit. Since the cover is exact, $3m-3n$ "filler" overlapping pairs as well as the corresponding left and right border intervals are assigned 3 colors each. This contributes $3(6m-6n)P^2+3(3m-3n)(45m+1)$ to the profit. Overall, the profit is $(42m+14)P^2+405m^2+45n^2+16m$.

We now prove the other direction. Suppose that we find an assignment of colors to intervals with total profit $(42m+14)P^2+405m^2+45n^2+16m$. The only way to get total profit of at least $(42m+14)P^2$ is by assigning 3 colors to all intervals with P^2 profit per allocated unit in the "two choice" gadgets, and in addition, by assigning 3 colors to all but one left (and right) border intervals, and by assigning the 1 remaining color to the remaining left (and right) border interval.

Consider the (element and "filler") overlapping pairs. At most $3m$ left (and right) intervals out of these overlapping pairs can be assigned 3 colors each, since any additional assignment would conflict with the assignment of colors to the border intervals. Out of these $3m$ left and right intervals, at most $3n$ left (and right) intervals can be element overlapping intervals.

Since all intervals with P^2 profit per allocated unit in the "two choice" gadgets are assigned 3 colors, there are $3m$ remaining 3 color blocks throughout the interval $[4n+15m, 4n+30m)$ and at most one more block of 3 colors is available in each of the $6m$ time units when some of the intervals with P^2 profit per allocated unit in the "two choice" gadgets are not assigned any color (see Fig. 3). The maximum profit that can be attained by assigning these colors to the unassigned intervals in the "two choice" gadgets is at most $3\cdot(2\cdot 6+2)m=42m$. Thus the only way to achieve the $405m^2$ term in the profit (for large enough m) is by actually assigning 3 colors to $3m$ left and $3m$ right intervals out of the overlapping pairs.

Consider the $3m$ left intervals of the overlapping pairs that are assigned 3 colors in increasing length order and the right intervals of the overlapping pairs that are assigned 3 colors in decreasing length order. Denote these two sequences by $O_1^L,\ldots,O_{3m}^L$ and $O_1^R,\ldots,O_{3m}^R$.

Claim 2 *For $i\in[1..3m]$, if O_i^L and O_i^R overlap, they cannot overlap by more than one time unit.*

Proof Suppose the claim does not hold, and let i be the minimum index for which O_i^L and O_i^R overlap by more than one time unit. However, in this case O_i^L and O_i^R must overlap in at least one time unit t when the intervals with P^2 profit per allocated unit in the "two choice" gadgets are assigned two blocks of 3 colors. Since O_i^R contains time t, t is contained also in $O_1^R,\ldots,O_{i-1}^R$. Similarly, since O_i^L contains time t, t is contained also in $O_{i+1}^L,\ldots,O_{3m}^L$. But this implies that $3m+1$ intervals out of the

overlapping pairs and 2 intervals from the "two choice" gadget are each assigned 3 colors. This is impossible, since there are only $9m + 7$ colors. □

From the discussion above, it follows that if O_i^L and O_i^R overlap they must be an overlapping pair. The maximum profit that can be attained from the intervals in the "two choice" gadgets that do not have P^2 profit per allocated unit is $14m$. Thus, to achieve the additional $405m^2 + 45n^2$ terms in the profit, we must have that for all $i \in [1..3m]$, intervals O_i^L and O_i^R are an overlapping pair, and $3n$ out of these overlapping pairs are element overlapping pairs. This will contribute $405m^2 + 45n^2 + 9m$ to the profit. The extra $7m$ profit needs to be attained by assigning colors to the intervals in the "two choice" gadgets that do not have P^2 profit per allocated unit. It follows that each "two choice" gadget needs to be colored using one of the two options described above and exactly n of them have to be colored using the second option. These n gadgets correspond to the exact cover. □

Finally, we note how the reduction can be modified to include only intervals of identical profit per allocated unit. The idea is to "slice" each interval in the original reduction to smaller intervals whose number is the profit per allocated unit of the original interval. When doing so, we need to ensure that it is not beneficial to move from a "slice" of one interval to a "slice" of another interval. This is done by assigning different displacements to the slices in different intervals, so that whenever we attempt to gain from a move from a "slice" of one interval to a "slice" of another interval, we lose at least one slice due to the different displacements. Thus, the same set of colors will be used for all slices associated with the original interval. This completes the proof of the theorem. □

References

1. Albers, S., Arora, S., Khanna, S.: Page replacement for general caching problems. In: SODA, pp. 31–40 (1999)
2. Bansal, N., Chakrabarti, A., Epstein, A., Schieber, B.: A quasi-PTAS for unsplittable flow on line graphs. In: STOC, pp. 721–729 (2006)
3. Bar-Noy, A., Bar-Yehuda, R., Freund, A., Naor, J., Schieber, B.: A unified approach to approximating resource allocation and scheduling. JACM **48**(5), 735–744 (2000)
4. Bar-Yehuda, R., Beder, M., Cohen, Y., Rawitz, D.: Resource allocation in bounded degree trees. Algorithmica **54**(1), 89–106 (2009)
5. Bar-Yehuda, R., Beder, M., Rawitz, D.: A constant factor approximation algorithm for the storage allocation problem. Algorithmica **77**(4), 1105–1127 (2017)
6. Batra, J., Garg, N., Kumar, A., Mömke, T., Wiese, A.: New approximation schemes for unsplittable flow on a path. In: SODA, pp. 47–58 (2015)
7. Belady, L.A.: A study of replacement algorithms for virtual-storage computer. IBM Syst. J. **5**(2), 78–101 (1966)
8. Buchsbaum, A.L., Karloff, H.J., Kenyon, C., Reingold, N., Thorup, M.: OPT versus LOAD in dynamic storage allocation. SIAM J. Comput. **33**(3), 632–646 (2004)
9. Călinescu, G., Chakrabarti, A., Karloff, H.J., Rabani, Y.: An improved approximation algorithm for resource allocation. ACM Trans. Algorithms **7**(4), 48 (2011)
10. Chakaravarthy, V.T., Choudhury, A.R., Gupta, S., Roy, S., Sabharwal, Y.: Improved algorithms for resource allocation under varying capacity. In: ESA, pp. 222–234 (2014)
11. Chekuri, C., Mydlarz, M., Shepherd, F.B.: Multicommodity demand flow in a tree and packing integer programs. ACM Trans. Algorithms **3**(3), 27 (2007)
12. Chen, B., Hassin, R., Tzur, M.: Allocation of bandwidth and storage. IIE Trans. **34**(5), 501–507 (2002)

13. Chrobak, M., Woeginger, G.J., Makino, K., Xu, H.: Caching is hard—even in the fault model. Algorithmica **63**(4), 781–794 (2012)
14. Gerstel, O.: Flexible use of spectrum and photonic grooming. In: Photonics in Switching (2010)
15. Grandoni, F., Mömke, T., Wiese, A., Zhou, H.: A (5/3+ϵ) approximation for unsplittable flow on a path: placing small tasks into boxes. In: STOC, pp. 607–619 (2018)
16. Jain, N., Menache, I., Naor, J., Yaniv, J.: A truthful mechanism for value-based scheduling in cloud computing. In: SAGT, pp. 178–189 (2011)
17. Jain, N., Menache, I., Naor, J., Yaniv, J.: A truthful mechanism for value-based scheduling in cloud computing. Theory Comput. Syst. **54**(3), 388–406 (2014)
18. Jinno, M., Takara, H., Kozicki, B., Tsukishima, Y., Sone, Y., Matsuoka, S.: Spectrum-efficient and scalable elastic optical path network: architecture, benefits, and enabling technologies. Comm. Mag. **47**, 66–73 (2009)
19. Karp, R.M.: Reducibility among combinatorial problems. In: Proceedings of Complexity of Computer Computations, pp. 85–103 (1972)
20. Knuth, D.: The Art of Computer Programming, Vol. 1: Fundamental Algorithms, 2nd edn. Addison-Wesley, Boston (1973)
21. Kolen, A.W., Lenstra, J.K., Papadimitriou, C.H., Spieksma, F.C.: Interval scheduling: a survey. Naval Res. Logist. **54**(5), 530–543 (2007)
22. Leonardi, S., Marchetti-Spaccamela, A., Vitaletti, A.: Approximation algorithms for bandwidth and storage allocation problems under real time constraints. In: FSTTCS, pp. 409–420 (2000)
23. Liberatore, V.: Uniform multipaging reduces to paging. Inf. Process. Lett. **67**(1), 9–12 (1998)
24. Mao, M., Humphrey, M.: Auto-scaling to minimize cost and meet application deadlines in cloud workflows. In: SC (2011)
25. Mömke, T., Wiese, A.: A $(2+\epsilon)$-approximation algorithm for the storage allocation problem. In: ICALP, pp. 973–984 (2015)
26. Ramaswami, R., Sivarajan, K.N., Sasaki, G.H.: Optical Networks: A Practical Perspective. Morgan Kaufmann Publisher Inc., San Francisco (2009)
27. Roy, B.V.: A short proof of optimality for the MIN cache replacement algorithm. Inf. Process. Lett. **102**(2–3), 72–73 (2007)
28. Schrijver, A.: Theory of Linear and Integer Programming. Wiley, Hoboken (1998)
29. Shachnai, H., Voloshin, A., Zaks, S.: Flexible bandwidth assignment with application to optical networks. J. Scheduling **21**(3), 327–336 (2018)
30. Shachnai, H., Voloshin, A., Zaks, S.: Optimizing bandwidth allocation in elastic optical networks with application to scheduling. J. Discrete Algorithms **45**, 1–13 (2017)
31. Shalom, M., Wong, P., Zaks, S.: Profit maximization in flex-grid all-optical networks. In: SIROCCO (2013)
32. Tutte, W.T.: Lectures on matroids. J. Res. Natl. Bur. Stand. (B) **69**, 1–47 (1965)
33. Velasco, L., Klinkowski, M., Ruiz, M., Comellas, J.: Modeling the routing and spectrum allocation problem for flexgrid optical networks. Photonic Netw. Commun. **24**(3), 177–186 (2012)
34. Voloshin, A.: Flexible resource allocation for network problems. Ph.D. Thesis, Computer Science Department, Technion (2017)

Affiliations

Dmitriy Katz[1] · Baruch Schieber[2] · Hadas Shachnai[3]

✉ Hadas Shachnai
hadas@cs.technion.ac.il

Dmitriy Katz
katzrog@us.ibm.com

Baruch Schieber
sbar@njit.edu

1 IBM T.J. Watson Research Center, P.O. Box 218, Yorktown Heights, NY 10598, USA

2 Department of Computer Science, New Jersey Institute of Technology, Newark, NJ 07102, USA

3 Computer Science Department, Technion, 3200003 Haifa, Israel