Flexible Resource Allocation for Clouds and All-Optical Networks

Dmitriy Katz\(^1\), Baruch Schieber\(^1\), and Hadas Shachnai\(^2\)*

\(^1\) IBM T.J. Watson Research Center, Yorktown Heights, NY 10598.
E-mail: \{katzrog,sbar\}@us.ibm.com

\(^2\) Computer Science Department, Technion, Haifa 3200003, Israel.
E-mail: hadas@cs.technion.ac.il

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 \(r_{\text{max}}(i)\) 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.

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., [17]). We focus on scheduling intervals with resource requirements. In this

* This work was partly carried out during a visit to DIMACS supported by the National Science Foundation under grant number CCF-1144502.
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., [18,5]), and the bandwidth allocation problems (see, e.g., [3,10]). 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; and 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 (BA) problem, where we only need to allocate to each activity the required amount of the resource.

Scheduling problems of this ilk arise naturally in scenarios where activities require (non-contiguous or contiguous) portion of an available resource, with some revenue associated with the allocated amount. In the cloud computing paradigm, the resource can be servers in a large computing cluster, storage capacity, or bandwidth allocated to time-critical jobs (see, e.g. [14,20]).

In flexgrid all-optical networks, the resource is the available spectrum of light that is divided into frequency intervals of variable width, with a gap of unused frequencies between them (see, e.g., [15,13]). Several high-speed signals connecting different source-destination pairs may share a link, provided they are assigned disjoint sub-spectra (see, e.g., in [22]). Given a path network and a set of connection requests, represented by intervals, each associated with a profit per allocated spectrum unit and a maximum bandwidth requirement, we need to feasibly allocate frequencies to the requests, with the goal of maximizing the total profit. In the FBAP variant the sub-spectra allocated to each request need not be contiguous, while in the FSAP variant each request requires a contiguous frequency spectrum.

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 schedule the intervals contiguously, subject to resource availability, 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 (FBAP) problem, 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.

Throughout the paper, we use the terms ‘intervals’ and ‘activities’ interchangeably.
The goal is to determine the amount of resource allocated to each interval so as to maximize the total profit.

Formally, 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 each interval, $J_i$, which can take any value up to the maximum possible value for $J_i$, given by $r_{\text{max}}(i)$.
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. (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.

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

1.3 Our Contribution

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 Section 3). Thus, we substantially improve the running time of the best known algorithm for FBAP (due to [25]) which uses flow techniques. Our algorithm is easy to implement and is thus practical.

On the negative side, we show (in Section 5) that FSAP 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 PTAS. We also present (in the Appendix) a $\frac{2k}{2k-1}$-approximation algorithm, where $k = \lceil \frac{W}{r_{\text{max}}} \rceil$, which is of practical interest. We further show (in Section 4) that FSAP admits a $(\frac{3}{4} + \varepsilon)$-approximation algorithm, for any fixed $\varepsilon > 0$, on instances whose job intervals form a proper interval graph.

Techniques: Our Algorithm, Paging_FPA, for the non-contiguous version of the problem, uses an interesting relation to the offline paging problem. The key idea is to view the available resource as slots in fast memory, and each job (interval) $J_i$ as $r_{\text{max}}(i)$ 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 that — in case of a page fault — evicts the page that is requested furthest in the future (see Section 3). If a page remains in the fast memory between the two times it was requested, then the resource that
corresponds to 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, $0 < r_{\text{min}}(i)$ on the amount of resource $J_i$ is allocated.

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 $\beta$ 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 $\beta$ fraction of the optimum (see Section 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 $\beta$. We believe this novel technique can lead to better approximation algorithms for other problems as well.

1.4 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]. The storage allocation problem (SAP) is NP-hard, since it includes Knapsack as a special case. SAP was first studied in [3,18]. Bar-Noy et al. [3] presented an approximation algorithm that yields a ratio of 7. Chen et al. [11] presented a polynomial time exact algorithm for the special case where all resource requirements are multiples of $W/K$, for some fixed integer $K \geq 1$. Bar-Yehuda et al. [4] presented a randomized algorithm for SAP with ratio $2 + \varepsilon$, and a deterministic algorithm with ratio $\frac{2\varepsilon-1}{\varepsilon-1} + \varepsilon < 2.582$. The best known result is a deterministic $(2+\varepsilon)$-approximation algorithm due to [23].

The bandwidth allocation (BA) problem is known to be strongly NP-hard, already for uniform profits [12]. The results of Albers et al. [11] 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. [8] developed a randomized approximation algorithm for BA 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 BA due to Chekuri et al. [10].

Both BA 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, BA can be viewed as the unsplittable flow problem (UFP) on a path. The best known result is a $(2+\varepsilon)$-approximation algorithm due to Anagnostopoulos et al. [2]. Batra et al. [6] obtained approximation schemes for some spacial cases.

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

\footnote{See also the recent results on UFP with Bag constraints (BagUFP) [9].}
For SAP with non-uniform resource, the best known ratio is $2 + \varepsilon$, obtained by a randomized algorithm of Möncke and Wiese [21].

The flexible variants of SAP and BA were introduced by Shalom et al. [25]. 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 $r_{\min}(i) \leq \lceil r_{\max}(i) \rceil$, for all $1 \leq i \leq n$. The paper [21] 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 difficult even if the bounds are identical for all activities, i.e., $r_{\min}(i) = \min$ and $r_{\max}(i) = \max$, for all $i$, where $0 < \min < \max \leq W$. The authors also show 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 [24], by showing that FSAP is strongly NP-hard even if $r_{\min}(i) = 0$ and $r_{\max}(i) = \max$, for all $i$.

Finally, the paper [23] 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 representing 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 properly contained in another interval $J_j$, for all $1 \leq i, j \leq n$.

Throughout the paper, we use coloring terminology when referring to the assignment of bandwidth 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. 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 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 \mod W) + 1, \ldots, \left(\left[\ell + |c(J_i)| - 2\right] \mod W\right) + 1\}, \text{ where } 1 \leq \ell \leq W. \text{ Given a coloring of the intervals, } c : \mathcal{J} \to 2^C, \text{ let } |c(J_i)| \text{ be the number of colors assigned to } J_i, \text{ then the total profit accrued from } c \text{ is } \sum_{i=1}^{n} p_i |c(J_i)|.

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 $S$, denoted $G_S = (S, E_S)$, 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 activity $i$ has also a lower bound $r_{\min}(i)$ on the amount of resource it is allocated.

Shalom et al.\[25\] 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 \[4\]. 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 so many colors are unavailable, 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 $r_{\text{max}}(i)$.

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 the Appendix (see Algorithm 2).

**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.** Given an instance of FBAP, where $p_i = 1$ for all $1 \leq i \leq n$, define a respective multipaging problem instance, where the size of the fast memory is $W$, as follows. (The multipaging problem is a variant of the paging problem, where more than one page is requested at the same time.) For each job $J_i \in \mathcal{J}$ define two types of page requests: feasibility requests and profit requests. For each $s_i \leq t < e_i$ there are $r_{\text{min}}(i)$ feasibility requests for the same $r_{\text{min}}(i)$ pages. In addition there are $r_{\text{max}}(i) - r_{\text{min}}(i)$ pairs of requests for $r_{\text{max}}(i) - r_{\text{min}}(i)$ distinct pages. The first request of each pair is at $s_i$ and the second at $e_i - 1$. It is not difficult to see that Belady’s algorithm is optimal also for the multipaging problem \[19\]. Consider an implementation of Belady’s algorithm for this instance where the feasibility requests are always processed before the profit requests. In this case it is easy to see that none of the pages that are requested in the feasibility requests will be evicted before the last time in which they are requested. Note that Paging\_FBA implements this variant of Belady’s algorithm, where AVAIL corresponds to the available memory slots. The optimality of Belady’s algorithm guarantees that the number of page faults is minimized. The page faults whenever a new page is requested cannot be avoided. (Note that there are $\sum_{i=1}^{n} r_{\text{max}}(i)$ such faults.) Additional page faults may occur when the second occurrence of a page in the pairs of the profit requests needs to be accessed. Minimizing the number of such page faults is equivalent to maximizing the number of pairs of profit requests for which the requested page is in memory throughout the interval $[s_i, e_i)$, that is, maximizing the profit of allocated resources in the FBAP instance. Paging\_FBA can be implemented in $O(n \log n)$ time, by noting that the total number of color reassignments is linear, and that once we sort the intervals by left endpoints, the ‘active’ intervals can be stored in a priority queue by right endpoints to implement the reassignments.

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 $\left(\frac{5}{4} + \epsilon\right)$ to optimal. Our Algorithm, Proper\_FSAP, uses the parameters $\epsilon > 0$ 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 and it is straightforward to verify by the
Let $J_{\text{wide}}$ denote the set of wide intervals $J_i$ for which $r_{\text{max}}(i) \geq \varepsilon W$. Let $J_{\text{narrow}}$ denote the complement set of narrow intervals. If the profit from the wide intervals that are actually assigned at least $\varepsilon W$ colors is large, namely, at least $\beta \cdot OPT_{\text{FSAP}}(J)$, then such a high profit subset of intervals is found and returned by the algorithm as the solution. Otherwise, Proper_FSAP calls Algorithm $A_{\text{Narrow,Color}}$ that finds a solution of high profit accrued from both narrow and wide intervals. The pseudocode for Proper_FSAP follows.

**Algorithm 1 Proper_FSAP($J, \bar{r}_{\text{max}}, \bar{p}, W, \varepsilon, \beta$)**

1. Guess the value of $OPT_{\text{FSAP}}(J)$, the optimal solution of FSAP on the input $J$.
2. Find in $J_{\text{wide}}$ 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 $P(S) < \beta OPT_{\text{FSAP}}(J)$ then
4. Let $S$ be the solution output by $A_{\text{Narrow,Color}}(J, \bar{r}_{\text{max}}, \bar{p}, W, \varepsilon, \beta OPT_{\text{FSAP}}(J))$
5. end if
6. Return $S$ and the respective contiguous coloring

We now describe Algorithm $A_{\text{Narrow,Color}}$ 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 $\beta$ fraction of the optimal solution.

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

Next, this solution is rounded to an integral solution of the FAP instance, of value at least $(1 - 2\varepsilon)OPT_{\text{FSAP}}(J)$. $A_{\text{Narrow,Color}}$ proceeds by converting the resulting (non-contiguous) coloring to a contiguous circular coloring with the same profit. Finally, this coloring is converted to a valid (non-circular) coloring. In this part, $A_{\text{Narrow,Color}}$ searches for the ‘best’ index for ‘cutting’ the circular coloring. This is done by examining a polynomial number of integral points, $\ell \in [1, (1 - \varepsilon)W]$, 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 $\ell$. The algorithm ‘cuts the circle’ in the point $\ell$ which causes the smallest harm to the total profit. For each wide interval $J_i$ crossing $\ell$, we assign the largest among its first block of colors (whose last color is $\ell$), or the second block of colors (which starts at $\ell \mod \lceil (1 - \varepsilon)W \rceil + 1$). For each narrow
interval that included ℓ, we assign the same number of new colors in the range \([(1-\varepsilon)W, W]\). We give the pseudocode of \(A_{\text{Narrow\_Color}}\) in Algorithm 3 in the Appendix.

4.1 Analysis of Proper_FSAP

Our main result is the following.

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

We prove the theorem using the following results. First, we consider the case where there exists an optimal solution of FSAP in which the profit from the intervals in \(J_{\text{wide}}\) that are assigned at least \(\varepsilon W\) colors is at least \(\beta_{\text{OPT\_FSAP}}(J)\).

**Lemma 3** If there exists an optimal solution of FSAP for \(J\) in which the profit from the intervals in \(J_{\text{wide}}\) that are assigned at least \(\varepsilon W\) colors is at least \(\beta_{\text{OPT\_FSAP}}(J)\), then a solution of profit at least \(\beta_{\text{OPT\_FSAP}}(J)\) can be found in polynomial time.

**Proof.** We consider only the intervals in \(J_{\text{wide}}\) and the set of feasible solutions \(S\) with the property that each interval in \(S\) is assigned at least \(\varepsilon W\) colors. Note that for each such feasible solution, at any time \(t > 0\), the number of active intervals is bounded by \(1/\varepsilon\), which is a constant. Hence, we can use dynamic programming to find a solution of maximum profit among all these feasible solutions. By our assumption, the value of this solution is at least \(\beta_{\text{OPT\_FSAP}}(J)\).

Next, we consider the complement case. In Figure 1 we give the linear program used by Algorithm \(A_{\text{Narrow\_Color}}\). For each \(J_i \in J_{\text{wide}}\) the linear program has a variable \(x_i\) indicating the number of colors assigned to \(J_i\). For each \(J_i \in J_{\text{wide}}\), 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 \(J_i \in J_{\text{wide}}\). 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 \(|J|\), since we only need to consider the “interesting” points of time \(t\), when \(t = s_i\) for some interval \(J_i \in J\), 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 \((1-2\varepsilon)OPT_{\text{FSAP}}(J)\).

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

---

6 This is similar to the dynamic programming algorithm for bounded load instances given in [25]. Thus, we omit the details.
\[ (LP_{\text{FBA}}) : \max \sum_{J_i \in J_{\text{narrow}}} p_i x_i + \sum_{J_i \in J_{\text{wide}}} p_i (x_i + y_i) \]
\[ \text{s.t. } \sum_{J_i \in J_{\text{wide}}} p_i y_i \leq \beta (1 - \varepsilon) OPT_{\text{FBA}}(J) \]
\[ \sum_{\{J_i \in J : t \in J_i\}} x_i + \sum_{\{J_i \in J_{\text{wide}} : t \in J_i\}} y_i \leq (1 - \varepsilon) W \quad \forall t > 0 \]
\[ 0 \leq x_i \leq \min\{r_{\text{max}}(i), \lceil \varepsilon W \rceil\} \quad \text{for } 1 \leq i \leq n \]
\[ 0 \leq y_i \leq r_{\text{max}}(i) - \lceil \varepsilon W \rceil \quad \text{for } J_i \in J_{\text{wide}}. \]

Fig. 1: The linear program \(LP_{\text{FBA}}\)

Let \(S_w\) be the set of intervals in \(J_{\text{wide}}\) 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.

\[ \sum_{J_i \in S_w} p_i (x_i + \lfloor y_i \rfloor) \geq \sum_{J_i \in S_w} p_i (x_i + y_i - 1) \]
\[ \geq \sum_{J_i \in S_w} p_i (x_i + y_i) (1 - \frac{1}{x_i}) \]
\[ \geq \sum_{J_i \in S_w} p_i (x_i + y_i) (1 - \frac{1}{\lceil \varepsilon W \rceil}) \]

The last inequality follows from the fact that \(x_i = \lceil \varepsilon W \rceil\), for any \(J_i \in S_w\). Thus, for \(W \geq 1/\varepsilon^2\), we have that the total profit after rounding the \(y_i\) values is at least \((1 - 2\varepsilon) OPT_{\text{FBA}}(J)\).

Now, consider the linear program \(LP_{\text{round}}\), 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 \(LP_{\text{round}}\) with profit at least \((1 - 2\varepsilon) OPT_{\text{FBA}}(J) - \sum_{J_i \in S_w} p_i b_i\). The rows of the coefficient matrix of \(LP_{\text{round}}\) 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). It follows that we can find in polynomial time an integral solution, \(\bar{x}_I^*, \bar{z}\), of the same total profit. Thus, the integral solution \(\bar{x}_I^*, \bar{z}\) satisfies the lemma. \[\blacksquare\]
Lemma 5 Let \( P(c'(S'_w)) \), \( P(c'(S'_n)) \) be the total profit from the intervals in the sets \( S'_w \) and \( S'_n \) in the circular coloring \( c' \) generated in Step 3 of A\_Circular. Then, (i) \( P(c''(S'_w)) \geq \frac{3}{4} P(c'(S'_w)) \), and (ii) \( P(c''(S'_n)) = P(c'(S'_n)) \).

Proof. Property (i) follows from a result of [25], which implies that the non-circular contiguous coloring \( c'' \), obtained for \( S'_w \) in Steps 17–18 of A\_Circular, has a total profit at least \( \frac{3}{4} P(c'(S'_w)) \). As shown in [25], this algorithm can be implemented in polynomial time.

Also, (ii) holds since (in Steps 22–23) Algorithm A\_Circular assigns \( |c'(J_i)| \) colors to any \( J_i \in S'_n \), i.e., in the resulting non-circular contiguous coloring, \( c'' \), we have \( |c''(J_i)| = |c'(J_i)|, \forall J_i \in S'_n \). We note 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 \( \ell_{good} \) at any time \( t > 0 \). It follows, that the intervals \( J_i \in S'_n \) that contain \( c'(J_i) \) \( \ell_{good} \) form an independent set. Since, for any \( J_i \in S'_n \), \( r_{max}(i) \leq \frac{\varepsilon W}{2} \), we can assign to all of these intervals the same set of new colors in the range \( \{(1 - \varepsilon)W + 1, \ldots, W\} \).

Proof of Theorem 2: Let \( 0 < \beta < 1 \) be the parameter used by Proper\_FSAP. For a correct guess of \( OPT_{FSAP}(\mathcal{J}) \) 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 \( J_{\text{wide}} \) that are assigned at least \( \varepsilon W \) colors is at least \( \beta OPT_{FSAP}(\mathcal{J}) \) then, by Lemma 3 Proper\_FSAP finds in Step 3 a solution of profit at least \( \beta OPT_{FSAP}(\mathcal{J}) \) in polynomial time.

(ii) Otherwise, consider the coloring \( c'' \) output by A\_Circular. Then, using Lemma 5, we have that

\[
\frac{P(c'(\mathcal{J}))}{P(c''(\mathcal{J}))} = \frac{P(c'(S'_w)) + P(c'(S'_n))}{\frac{3P(c(S'_w))}{4} + P(c'(S'_n))} \leq \max_{0 < \varepsilon \leq \beta} \frac{1}{\frac{3\varepsilon}{4} + 1 - \varepsilon} = \frac{4}{4 - \beta}
\]

Now, applying Lemma 4 and the rounding in Step 6 of A\_Circular with \( \hat{\varepsilon} \), we have that \( P(c'(\mathcal{J})) \geq (1 - 3\hat{\varepsilon})OPT_{FSAP}(\mathcal{J}) = (1 - \varepsilon)OPT_{FSAP}(\mathcal{J}) \).

The theorem follows by taking \( \beta = \frac{3\varepsilon}{4} \).

5 Uniform FSAP Instances

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

We first prove that if \( k = W / \text{Max} \) (i.e., \( W \) is an integral multiple of \( \text{Max} \)) then FSAP can be solved optimally by finding a maximum \( k \)-colorable subgraph in \( G \), and assigning to each interval in this subgraph \( \text{Max} \) contiguous colors. In contrast to this case if \( k > W / \text{Max} \), then we show that FSAP is NP-Hard, and give a polynomial approximation scheme to solve such instances.

Lemma 6 An FSAP instance where for all \( 1 \leq i \leq n \), \( r_{max}(i) = \text{Max} \), and \( W \) is multiple of \( \text{Max} \) can be solved exactly in polynomial time.
Proof. Consider an FBAP instance where for all $1 \leq i \leq n$, $r_{\max}(i) = \text{Max}$, and $W$ is multiple of $\text{Max}$. We claim that there exists an optimal solution to this FBAP instance in which each job gets either 0 or $\text{Max}$ 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 $r_{\max}(i) = \text{Max}$. Note that when we start the algorithm $|\text{AVAIL}|$ is a multiple of $\text{Max}$ and thus when the first (left) endpoint $s_1$ is considered the job $J_1$ is allocated $\text{Max}$ colors and $|\text{AVAIL}|$ remains a multiple of $\text{Max}$. Assume inductively that as the endpoints are scanned all the jobs that were allocated colors up to this endpoint were allocated $\text{Max}$ colors and that $|\text{AVAIL}|$ is a multiple of $\text{Max}$. If the current scanned endpoint is a right endpoint $e_i$, then by the induction hypothesis either 0 or $\text{Max}$ colors are added to AVAIL and nothing else is changed. Suppose that the scanned endpoint is a left endpoint $s_i$. If $|\text{AVAIL}| > 0$ then $\text{Max}$ colors from AVAIL are allocated to $J_i$. Otherwise, $J_i$ may be allocated colors that were previously allocated to another job $J_\ell$, for $\ell < i$. However, in this case because $r_{\min}(\ell) = 0$ all the $\text{Max}$ colors allocated to $J_\ell$ will be moved to $J_i$, and the hypothesis still holds.

The optimal way to assign either 0 or $\text{Max}$ colors to each job is given by computing the maximum $k = (W/\text{Max})$-colorable subgraph of $G$ and assigning $\text{Max}$ 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 $k > W/\text{Max}$.

Theorem 7. 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.

5.1 An Approximation Scheme

We now describe a PTAS for uniform instances of FSAP. Fix $\varepsilon > 0$, and let $\mathcal{J}$ be a uniform input for FSAP. Denote by $OPT_{\text{FSAP}}(\mathcal{J})$ the value of an optimal solution for instance $\mathcal{J}$ of FSAP.

The scheme handles separately two subclasses of inputs, depending on the value of $\text{Max}$. First, we consider the case where $\text{Max}$ is large relative to $W$, or more precisely $k = \lceil W/\text{Max} \rceil \leq 1/\varepsilon$. We prove in the next lemma that in this case if we partition the colors into a constant number of contiguous strips and limit our solution to always assigning all the colors in the same strip to the same job, we can find a solution that is at least $(1 - \varepsilon)OPT_{\text{FSAP}}(\mathcal{J})$. The size of each strip (except possibly the last one) is $\lceil \varepsilon \text{Max}/4 \rceil \geq 1$. Since the number of strips is $O(1/\varepsilon^2)$, we can find an approximation in this case using dynamic programming as shown in [25].

Lemma 8 Any optimal solution for a uniform input $\mathcal{J}$ for FSAP where $k = \lceil W/\text{Max} \rceil \leq 1/\varepsilon$ can be converted to a solution where each interval is assigned a number of colors that is an integral number of strips, and the total profit is at least $(1 - \varepsilon)OPT_{\text{FSAP}}(\mathcal{J})$.
Proof. W.l.o.g., we may assume that $\lfloor \varepsilon\text{Max}/4 \rfloor \geq 1$, else we have that $W$ is a constant, and we can solve $\text{FSAP}$ optimally in polynomial time (see [25]). Given an optimal solution for a uniform input $J$, 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 $S$). Using the above partition of the colors to strips, we have $1 \leq N \leq \lfloor \frac{4W}{\varepsilon\text{Max}} \rfloor$ 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., $S_j = \{J_i | c(J_i) \cap C_j \neq \emptyset\}$. Initialize for all $J_i \in S$, $c'(J_i) = 0$. (i) For all $1 \leq j \leq N$, find in $G_S$ a maximum independent set, $I_j$ of intervals $J_i \in S_j$. For any $J_i \in I_j$, 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 $|c'(J_i)| > \text{Max}$ 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 $|c'(J_i)| \leq \text{Max}$. We show below that the above *strip coloring* for $S$ is feasible, and that the total profit from the strip coloring is at least $(1 - \varepsilon)\text{OPT}_{\text{FSAP}}(J)$. 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 $S_{j+1}$, and thus any maximum independent set $I_{j+1}$ 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., $J_i \in S_j \cap S_{j+1} \cap \cdots \cap S_{j+t}$, for $t > 1$, then $J_i \in I_{j+1} \cap \cdots \cap I_{j+t-1}$. 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 $|c'(J_i)| \leq \text{Max}$. Now, consider the profit of the strip coloring. We note that after step (i), the total profit of $c'$ is at least $\text{OPT}_{\text{FSAP}}(J)$. This is because for each strip $j$, $|C_j| \cdot |Z_j|$ is an upper bound on the profit that can be obtained from this strip. We show that the harm 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 $|c(J_i)| \geq (1 - \varepsilon/2)\text{Max}$. 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 $2\lfloor \frac{\text{Max}}{2} \rfloor \leq (\varepsilon/2)\text{Max}$. Since $|c(J_i)| \geq (1 - \varepsilon/2)\text{Max}$, we have that after step (ii), $|c'(J_i)| \geq |c(J_i)| - (\varepsilon/2)\text{Max} \geq (1 - \varepsilon)\text{Max}$. (b) For any interval $J_i$ for which $|c(J_i)| < (1 - \varepsilon/2)\text{Max}$, since after step (i) $|c'(J_i)| \leq |c(J_i)| + 2\lfloor \frac{\text{Max}}{4} \rfloor$, we have that $|c'(J_i)| \leq \text{Max}$. 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 $\text{OPT}'_{\text{FSAP}}(J) \geq (1 - \varepsilon)\text{OPT}_{\text{FSAP}}(J)$. Now, consider the case where $\text{Max}$ is small, i.e., $k = \lceil W/\text{Max} \rceil > 1/\varepsilon$. In this case we consider just $(k - 1)\text{Max}$ consecutive colors and ignore the remainder up to $\varepsilon W$ colors. Recall that when the number of colors is a multiple of $\text{Max}$ we can
find an optimal solution. Let $OPT_{\text{fsap}}(\mathcal{J})$ denote the value of an optimal solution for instance $\mathcal{J}$ of $\text{fsap}$ with $W$ colors, and recall that $k = \lceil W/\text{Max} \rceil$. Since $(k-1)\text{Max} < W < k\text{Max}$, $OPT_{\text{fsap}}((k-1)\text{Max}) < OPT_{\text{fsap}}(W) < OPT_{\text{fsap}}(k\text{Max})$. The value of $OPT_{\text{fsap}}((k-1)\text{Max})$ is $\text{Max}$ times the size of the max $(k-1)$-colorable subgraph of $G$, and the value of $OPT_{\text{fsap}}(k\text{Max})$ is $\text{Max}$ times the size of the max $k$-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 $OPT_{\text{fsap}}((k-1)\text{Max})(\mathcal{J}) \geq (1-\varepsilon)OPT_{\text{fsap}}(k\text{Max})(\mathcal{J}) > (1-\varepsilon)OPT_{\text{fsap}}(W)(\mathcal{J})$. Combining the results, we have

**Theorem 9.** The above algorithm is a PTAS for uniform $\text{fsap}$ instances.

**Acknowledgments:** We thank Magnús Halldórsson and Viswanath Nagarajan for valuable discussions.

**References**

1. S. Albers, S. Arora, and S. Khanna. Page replacement for general caching problems. In SODA, pages 31–40, 1999.
2. A. Anagnostopoulos, F. Grandoni, S. Leonardi, and A. Wiese. A mazing $2+\varepsilon$ approximation for unsplittable flow on a path. In SODA, pages 26–41, 2014.
3. A. Bar-Noy, R. Bar-Yehuda, A. Freund, J. Naor, and B. Schieber. A unified approach to approximating resource allocation and scheduling. JACM, 48(5):735–744, 2000.
4. R. Bar-Yehuda, M. Beder, Y. Cohen, and D. Rawitz. Resource allocation in bounded degree trees. Algorithmica, 54(1):89–106, 2009.
5. R. Bar-Yehuda, M. Beder, and D. Rawitz. A constant factor approximation algorithm for the storage allocation problem: extended abstract. In SPAA, pages 204–213, 2013.
6. J. Batra, N. Garg, A. Kumar, T. Mömke, and A. Wiese. New approximation schemes for unsplittable flow on a path. In SODA, pages 47–58, 2015.
7. L. A. Belady. A study of replacement algorithms for virtual-storage computer. IBM Systems Journal, 5(2):78–101, 1966.
8. G. Călinescu, A. Chakrabarti, H. J. Karloff, and Y. Rabani. An improved approximation algorithm for resource allocation. ACM Transactions on Algorithms, 7(4), 2011.
9. V. T. Chakaravarthy, A. R. Choudhury, S. Gupta, S. Roy, and Y. Sabharwal. Improved algorithms for resource allocation under varying capacity. In ESA, pages 222–234, 2014.
10. C. Chekuri, M. Mydlarz, and F. B. Shepherd. Multicommodity demand flow in a tree and packing integer programs. ACM Transactions on Algorithms, 3(3), 2007.
11. B. Chen, R. Hassin, and M. Tzur. Allocation of bandwidth and storage. IIE Transactions, 34(5):501–507, 2002.
12. M. Chrobak, G. J. Woeginger, K. Makino, and H. Xu. Caching is hard - even in the fault model. Algorithmica, 63(4):781–794, 2012.
13. O. Gerstel. Flexible use of spectrum and photonic grooming. In Photonics in Switching, 2010.
14. N. Jain, I. Menache, J. Naor, and J. Yaniv. A truthful mechanism for value-based scheduling in cloud computing. In SACT, pages 178–189, 2011.
15. M. Jinno, H. Takara, B. Kozicki, Y. Tsukishima, Y. Sone, and S. Matsuoka. Spectrum-efficient and scalable elastic optical path network: architecture, benefits, and enabling technologies. Comm. Mag., 47:66–73, 2009.
16. R. M. Karp. Reducibility among combinatorial problems. In Proc. Complexity of Computer Computations, pages 85–103, 1972.
17. A. W. Kolen, J. K. Lenstra, C. H. Papadimitriou, and F. C. Spieksma. Interval scheduling: A survey. Naval Research Logistics, 54(5):530–543, 2007.
18. S. Leonardi, A. Marchetti-Spaccamela, and A. Vitaletti. Approximation algorithms for bandwidth and storage allocation problems under real time constraints. In FSTTCS, pages 409–420, 2000.
19. V. Liberatore. Uniform multipaging reduces to paging. Inf. Process. Lett., 67(1):9–12, 1998.
20. M. Mao and M. Humphrey. Auto-scaling to minimize cost and meet application deadlines in cloud workloads. In SC, 2011.
21. T. Mömke and A. Wiese. A (2+ε)-approximation algorithm for the storage allocation problem. In ICALP, pages 973–984, 2015.
22. R. Ramaswami, K. N. Sivarajan, and G. H. Sasaki. Optical Networks: A Practical Perspective. Morgan Kaufmann Publisher Inc., San Francisco, 2009.
23. H. Shachnai, A. Voloshin, and S. Zaks. Flexible bandwidth assignment with application to optical networks. In MFCS, 2014.
24. H. Shachnai, A. Voloshin, and S. Zaks. Optimizing bandwidth allocation in flex-grid optical networks with application to scheduling. In IPDPS, 2014.
25. M. Shalom, P. Wong, and S. Zaks. Profit maximization in flex-grid all-optical networks. In SIROCCO, 2013.

A Algorithm Paging_{FBA}

Algorithm 2 gives the pseudocode of Paging_{FBA}.

B Hardness of FSAP for Uniform Instances

Proof of Theorem 7. 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 [16] 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 to each interval in [9..16]. Assuming that three colors are indeed
Algorithm 2 Paging_FBA($\mathcal{J}, \bar{r}_{\min}, \bar{r}_{\max}, W$)

1: $SOL = 0.$
2: AVAIL = [1..W].
3: W.l.o.g. assume that all endpoints are distinct. Sort the left endpoints of the intervals in $\mathcal{J}$ in non-decreasing order. Let $L$ denote the sorted list.
4: while not end-of-list do
5:   Consider the next endpoint $L_i$.
6:   if the endpoint is $s_i$, the left endpoint of $J_i$, then
7:     $SOL+ = r_{\min}(i)$.
8:     $A_i = \emptyset$.
9:     if $|AVAIL| > r_{\min}(i)$ then
10:        Move $r_{\min}(i)$ colors from AVAIL to $A_i$.
11:     else
12:         $A_i = AVAIL$.
13:         AVAIL = \emptyset.
14:     end if
15:     while $|A_i| < r_{\min}(i)$ do
16:        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 $e_k$.
17:        Move $\min\{|A_k| - r_{\min}(k), r_{\min}(i) - |A_i|\}$ colors from $A_k$ to $A_i$.
18:     end while
19:     end if
20: Move $\min\{|AVAIL|, r_{\max}(i) - r_{\min}(i)\}$ colors from AVAIL to $A_i$.
21: $SOL+ = \min\{|AVAIL|, r_{\max}(i) - r_{\min}(i)\}$.
22: Let $\mathcal{J}_i = \{J_k \in \mathcal{J}| s_k < s_i \land e_k > e_i \land |A_k| > r_{\min}(k)\}$.
23: while $|A_i| < r_{\max}(i) \land |\mathcal{J}_i| > 0$ do
24:    Let $J_k$ be the interval with the maximum right endpoint in $\mathcal{J}_i$.
25:    Move $\min\{|A_k| - r_{\min}(k), r_{\max}(i) - |A_i|\}$ colors from $A_k$ to $A_i$.
26: end while
27: else
28:    {The endpoint is $e_i$, the right endpoint of $J_i$}
29:    AVAIL = AVAIL $\cup A_i$.
30: end if
31: end while
32: Return $SOL$ and the sets $A_1, \ldots A_n$ of colors assigned to the jobs in $\mathcal{J}$.

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 \([t_2, t_2 + 1]\).
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 \( i \in [1..3n] \),

\[
L_i = [0, i), \text{ for } i \in [3n + 1..3m], \quad L_i = [0, 4n), \text{ and for } i \in [3m + 1..3m + 3],
\]

\[
L_i = [0, 4n + 15m).
\]

(ii) **Right border**: \( 3m + 3 \) intervals \( R_1, \ldots, R_{3m+3} \) each of profit \( P^2 \). For \( i \in [1..3n] \),

\[
R_i = [5n + 45m + i, 8n + 45m + 1), \text{ for } i \in [3n + 1..3m], \quad R_i = [4n + 45m, 8n + 45m + 1), \text{ and for } i \in [3m + 1..3m + 3],
\]

\[
R_i = [4n + 30m, 8n + 45m + 1).
\]

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

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 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 9, 10, 15, 9 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.)

**Fig. 5:** Assigning intervals related to \(S_i\)

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 $S_l$ is in the cover. Let $S_l = \{c_a, c_b, c_c\}$, where \{a, b, c\} ⊆ [1..3n]. Choose the second option for the “two choice” gadget associated with $S_l$. 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 $L_{4m+2}, R_{4m+2}$ 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 $24nP^2 + 7m$ 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, $3n \cdot 3n$ “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...
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 $O^L_1, \ldots, O^L_{3m}$ and $O^R_1, \ldots, O^R_{3m}$.

Claim 1 For $i \in [1..3m]$, if $O^L_i$ and $O^R_i$ 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 $O^L_i$ and $O^R_i$ 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 $O^R_i$ contains time $t$, $t$ is contained also in $O^R_{i+1}, \ldots, O^R_{3m}$. Similarly, since $O^L_i$ contains time $t$, $t$ is contained also in $O^L_{i+1}, \ldots, O^L_{3m}$. 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 $O^L_i$ and $O^R_i$ 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 $405m^2 + 45m^2$ terms in the profit, we must have that for all $i \in [1..3m]$, intervals $O^L_i$ and $O^R_i$ are an overlapping pair, and 3n out of these overlapping pairs are element overlapping pairs. This will contribute $405m^2 + 45m^2 + 9m$ 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.

C Algorithm $A_{Narrow\_Color}$

The pseudocode of $A_{Narrow\_Color}$ is given in Algorithm 3.
Algorithm 3 \( A_{\text{Narrow\_Color}}(J, \bar{f}_\text{max}, \bar{p}, W, \epsilon, P) \)

1: Solve the linear program \( LP_{\text{fba}} \).
2: Round the solution to obtain a (non-contiguous) coloring \( c \).
3: Find a circular contiguous coloring \( c' \) of the same total profit as \( c \). Such a coloring exists since the input graph is proper, and is obtained by scanning the intervals from left to right and assigning the colors in a fixed circular order.
4: Let \( S' \) be the set of colored intervals in \( c' \).
5: Let \( S'_w \subseteq S' \) be the subset of intervals \( J_i \in S' \) for which \( |c'(J_i)| \geq \epsilon W \), and let \( S'_w = S' \setminus S'_w \).
6: For any \( J_i \in S'_w \), round down \( |c'(J_i)| \) to the nearest integral multiple of \( \epsilon^2 W \), and eliminate the corresponding amount of colors in \( c'(J_i) \), such that the first color assigned to \( J_i \) is \( f_i = \lfloor r \cdot \epsilon^2 W \rfloor \), for some integer \( r \geq 0 \).
7: For \( \ell = \lfloor r \cdot \epsilon^2 W \rfloor \), \( r = 0, 1, \ldots, \lfloor \frac{1}{\epsilon^2} \rfloor \) do
8: Let \( S'_w(\ell) = \{ J_i \in S'_w | \ell, |\ell \mod \lfloor (1 - \epsilon) W \rfloor + 1 \} \subseteq c'(J_i) \}
9: Let \( P(S'_w(\ell)) = 0 \)
10: For \( J_i \in S'_w(\ell) \) do
11: Suppose that \( c'(J_i) = \{ f_i, [f_i \mod \lfloor (1 - \epsilon) W \rfloor + 1, \ldots, t_i \} \), for some \( 1 \leq f_i, t_i \leq \lfloor (1 - \epsilon) W \rfloor \).
12: Partition the set of \( |c'(J_i)| \) colors assigned to \( J_i \) into two contiguous blocks:
   \( \text{Block}_1(i) = \{ f_i, \ldots, \ell \} \), and \( \text{Block}_2(i) = \{ \ell \mod \lfloor (1 - \epsilon) W \rfloor + 1, \ldots, t_i \} \).
13: Let \( P(S'_w(\ell)) = p_i \cdot \min \{ |\text{Block}_1(i)|, |\text{Block}_2(i)| \} \)
14: End for
15: End for
16: Let \( \ell_{\text{good}} = \arg \min_{\ell \leq \lfloor (1 - \epsilon) W \rfloor} P(S'_w(\ell)) \).
17: For \( J_i \in S'_w(\ell_{\text{good}}) \) do
18: Assign to \( J_i \) the larger of \( \text{Block}_1(i) \) and \( \text{Block}_2(i) \).
19: End for
20: Renumber the first \( \lfloor (1 - \epsilon) W \rfloor \) colors starting at \( \ell_{\text{good}} \mod \lfloor (1 - \epsilon) W \rfloor + 1 \).
21: Let \( S'_w(\ell_{\text{good}}) = \{ J_i \in S'_w | \lfloor \ell_{\text{good}} \mod \lfloor (1 - \epsilon) W \rfloor + 1 \} \subseteq c'(J_i) \}
22: For \( J_i \in S'_w(\ell_{\text{good}}) \) do
23: Assign to \( J_i \) \( |c'(J_i)| \) contiguous colors in the set \( \{ \lfloor (1 - \epsilon) W \rfloor + 1, \ldots, W \} \).
24: End for
25: Return \( c'' \) the resulting coloring of \( S' \).

D A \( \frac{2k}{2k-1} \)-approximation Algorithm for Uniform FSAP

We describe below Algorithm \( A_{\text{Max\_Small}} \) for uniform FSAP instances. It yields solutions that are close to the optimal as \( k = \lceil \frac{W_{\text{fba}}}{W_{\text{fba}}} \rceil \) gets large. Initially, \( A_{\text{Max\_Small}} \) solves optimally \( \text{FBAP} \) on the input graph \( G \). Let \( G' \) be the support graph for this solution. \( A_{\text{Max\_Small}} \) proceeds by generating a feasible solution for FSAP as
follows. Consider the set of intervals \( J_i \in G' \) for which \( |c(J_i)| = \text{Max} \). Denote the support subgraph of this set of intervals \( G_2 \). 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 \( k\text{Max} > W \) colors). Consequently, each interval in \( G_2 \) can be assigned a set of \( \text{Max} \) contiguous colors, using a total of \((k-1)\text{Max}\) colors. \( A_{\text{MaxSmall}} \) 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 \( \text{Max} \) colors (see the pseudocode in Algorithm 4).

```
Algorithm 4 \( A_{\text{MaxSmall}}(J, \text{Max}, W) \)

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

**Theorem 11.** For any uniform instance of fsap, \( A_{\text{MaxSmall}} \) yields an optimal solution for fsap, if \( k \in \{1, 2\} \), and a \( \frac{2k}{2k-1} \)-approximation for any \( k \geq 3 \).

The proof of Theorem 11 uses the next lemma. Recall that \( G_1 \subseteq G' \) is the subgraph induced by the intervals \( J_i \) for which \( |c(J_i)| < \text{Max} \).

**Lemma 12** The subgraph \( G_1 \subseteq G' \) is proper and 2-colorable.

**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 properly contained in \( J_i \). By the way \text{Paging\_FBA} proceeds, when it colors \( J_j \), some colors that were assigned to \( J_i \) should be assigned to \( J_j \), until either \( |c(J_j)| = \text{Max} \), or \( |c(J_i)| = 0 \). Since none of the two occurs - a contradiction.

We now show that \( G_1 \) is 2-colorable. 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 \( \sum_{(J_i \in J, t \in J_i)} |c(J_i)| = W \). Let \( T \) 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 $J_i \in G_1$ contains 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 2** Every time point $t \in \mathcal{T}$ is contained in exactly one interval $J_i \in G_1$.

We now show that Claim 2 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 2.

**Proof of Claim 2.** First, note that since $W$ is not a multiple of $\lambda$ 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 $t_i \in J_i$. Since $t_{i-1}$ is not contained in any other interval in $G_1$, and since it is tight, we must have $|c(J_i)| = W$ mod $\lambda$. 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 $J_{i+1}$ (since $t_i > e_{i-1}$). If $t_i$ is not contained in $J_i$ then clearly, by our assumption, $t_i$ must also be contained in $J_{i+2}$. However, even if $t_i \in J_i$, since $|c(J_i)| = W$ mod $\lambda$, in order for $t_i$ to be tight, it must be contained also in $J_{i+2}$. In this case, since there is at least one available color in $[s_{i+1}, s_{i+2}]$, and since $e_{i+2} > e_{i+1}$, Algorithm Paging_FBA would assign at least one additional color to $J_i$, rather than to $J_{i+2}$. A contradiction. □

We are ready to show the performance ratio of Algorithm $A_{\text{MaxSmall}}$.

**Proof of Theorem 11.** Given the graphs $G'$ and $G_1, G_2$, as defined in Steps 4 and 5 of $A_{\text{MaxSmall}}$, respectively. Let $OPT_{\text{FSA}}(G)$ and $A(G)$ be the value of an optimal solution and the solution output by $A_{\text{MaxSmall}}$, respectively, for an input graph $G$. Clearly, $OPT_{\text{FSA}}(G) \leq OPT_{\text{FBA}}(G)$.

Case 1: $k < 3$. Since in this case $G_1$ is an independent set, $A(G) = OPT_{\text{FBA}}(G)$, and thus $A(G) = OPT_{\text{FSA}}(G)$.

Case 2: $k \geq 3$. Since in this case $G_1$ is 2-colorable, $A(G) \geq OPT_{\text{FBA}}(G) - \frac{k}{2} \cdot (W \mod \lambda) \cdot |G_1|$. Hence, to get the approximation ratio $\frac{2k}{2k-1}$, we need to show that $(W \mod \lambda) \cdot |G_1| \leq \frac{1}{k} OPT_{\text{FBA}}(G)$. Let $\lambda \lambda = W \mod \lambda$. Note that $0 < \lambda < 1$. In fact, we prove a slightly better bound, as we show that

$$\frac{(W \mod \lambda) \cdot |G_1|}{OPT_{\text{FBA}}(G)} \leq \frac{\lambda}{k-1+\lambda} < \frac{1}{k}.$$  \hspace{1cm} (3)

This implies the approximation ratio $\frac{2k-2+2\lambda}{2k-2+\lambda} < \frac{2k}{2k-1}$. To obtain a contradiction, assume that inequality (3) does not hold. Recall that $OPT_{\text{FBA}}(G) = (W \mod
Max) · |G_1| + Max · |G_2|, and that |G_2| = |G'| − |G_1|. We get

\[
\frac{(W \mod \text{Max}) \cdot |G_1|}{OPT_{fba}(G)} = \frac{\lambda \text{Max}|G_1|}{\text{Max}|G'| - \text{Max}(1 - \lambda)|G_1|} = \frac{\lambda |G_1|}{|G'| - (1 - \lambda)|G_1|} > \frac{\lambda}{k - 1 + \lambda}.
\]

This implies \( k|G_1| > |G'| \), and thus

\[
OPT_{fba}(G) = \lambda \text{Max}|G_1| + \text{Max}(|G' − |G_1|) = \text{Max}|G' − (1 - \lambda)\text{Max}|G_1|
\]

\[
< (1 - \frac{1}{k}(1 - \lambda))\text{Max}|G'| = \frac{k - 1 + \lambda}{k} \cdot \text{Max}|G'|.
\]

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 \( W \mod \text{Max} \) colors to each interval in \( G_1 \)). We can use the \( k \) coloring to obtain a solution of the FBAP instance, by assigning \( \text{Max} \) colors to intervals in the \( k - 1 \) largest color classes, and \( W \mod \text{Max} \) to the remaining color class. Thus, \( OPT_{fba}(G) > \frac{1}{k} \cdot \lambda \text{Max}|G'| + \frac{k - 1}{k} \cdot \text{Max}|G'| = \frac{k - 1 + \lambda}{k} \cdot \text{Max}|G'|. \) A contradiction. \( \blacksquare \)