

Approximation Algorithms for Bandwidth and Storage Allocation

Reuven Bar-Yehuda*

Michael Beder*

Yuval Cohen*

July 5, 2005

Abstract

We discuss two NP-hard allocation problems, the *Bandwidth Allocation Problem* (BAP) and the *Storage Allocation Problem* (SAP). In BAP we are given a set of requests, each of which is associated with a time interval, a bandwidth demand, and a profit. A feasible solution is a subset of requests such that at any given time the total bandwidth given to allocated requests does not exceed the available bandwidth. Our goal is to find a feasible solution of maximum total profit. SAP is a variation of BAP in which we are given two additional constraints: (i) the specific portion of the resource allocated to a request cannot change during the time interval of the request, and (ii) the allocation of the resource must be contiguous. We present two deterministic polynomial-time approximation algorithms one for BAP and the other for SAP. The performance guarantee of both algorithms is $(2 + 1/(e - 1))^{-1} - \epsilon \approx 1/2.582$. The best previously known deterministic approximation ratio for BAP is $1/3$ (randomized $1/2$), and the best previously known ratio for SAP is $1/7$. The algorithms are based on a new application of the *Local Ratio technique* in which a series of approximations is packed into narrow bandwidth slices.

1 Introduction

We are given a finite set $\mathcal{I} = \{I_1, \dots, I_n\}$ of requests and a *bandwidth capacity* of 1. Each request $I_j = (s_j, t_j, b_j, p_j) \in \mathbb{R}^4$ is characterized by: (i) s_j, t_j are the *start* and *end* times of the request; (ii) $0 < b_j \leq 1$ the demanded *bandwidth*; and (iii) $p_j > 0$ the *profit* for selecting the request.

In the BAP problem, we wish to select a profitable subset of requests such that at any time the total bandwidth does not exceed the bandwidth capacity. SAP is a variation of BAP with the following additional constraint: a selected request must be granted a spatial area in the bandwidth capacity that must not be changed during the time interval of the request.

A BAP application is profit maximization in network sessions selection. Each session must be scheduled at a fixed time interval and demands an amount of bandwidth. At any time, the bandwidth allocated for the selected sessions must not exceed the network's total bandwidth.

A SAP application is profit (or throughput) maximization in a multi-threaded environment where threads request allocations of memory blocks at fixed time intervals. The goal is to maximize the profit of the selected threads while reserving contiguous memory blocks for each selected thread within the memory capacity such that no address is in use by two threads at the same time. The solution would consist of the selected threads as well as the address of each allocated memory block.

A geometric representation of SAP is to interpret each request as an axis-aligned rectangle that has a fixed profit (p_j), height (b_j), a fixed length ($t_j - s_j$) and is only allowed to move

*Department of Computer Science, Technion, Haifa 32000, Israel. E-mails: {reuven@cs, sbederm1@cs, scyuval@t2}.technion.ac.il.

vertically. We wish to select a subset of the rectangles and then pack these rectangles pairwise-disjoint into a rectangular frame of a given size, so as to maximize the total profit of the selected rectangles.

A closely related problem to SAP is DYNAMIC STORAGE ALLOCATION (DSA). Similarly to SAP, in DSA we are given a set of rectangles that can only move vertically. The goal is to minimize the total height required to pack all rectangles such that no two rectangles overlap. We use previously developed algorithms for DSA presented in [6, 9].

These problems are generalizations of other main allocation problems. Both BAP and SAP in the restricted case of $\forall j : s_j = 0, t_j = 1$ become the one-dimensional knap-sack problem. Both problems in the restricted case of $\forall j : b_j = 1$ become the problem of finding a maximal independent set in a weighted interval graph.

1.1 Prior Work and Our Contribution

Due to the vast applications of these problems, extensive research has been done revolving both problems. A special case of the BAP, where all requests have the same duration, has been studied by Arkin and Silverberg [1]. Philips et al. [11]. developed approximation algorithms for both BAP and SAP with performance guarantees of $1/6$ and $1/35$ respectively. Leonardi et al. [10] obtained a $1/12$ -approximation algorithm for the SAP. Bar-Noy et al. [3] improved the best known performance guarantees for these problems to $1/3$ and $1/7$ respectively using the Local Ratio technique. Chen et al. [8] developed dynamic programming algorithms solving restricted cases of both BAP and SAP. Calinescu et al. [7] also developed a dynamic programming algorithm for a restricted BAP case as well as a non-deterministic non-combinatorial approximation algorithm with expected performance guarantee of $1/2 - \epsilon$.

Our main results in this paper are presenting an iterative approximation technique using a new application of Local Ratio and two polynomial-time combinatorial deterministic approximation algorithms for BAP and SAP with performance guarantees of $(2 + 1/(e - 1))^{-1} - \epsilon \approx 1/2.6 - \epsilon$ using this technique. Hence, we provide the best known deterministic approximation algorithm for BAP and improve the best known performance guarantee for SAP.

2 Preliminaries

2.1 Problems Formulation

We refer to the requests as generalized time *intervals* characterized by bandwidth and profit. In both BAP and SAP, the input consists of a set $\mathcal{I} = \{I_1, \dots, I_n\}$ of n intervals, each interval I_j is a quadruple $(s_j, t_j, b_j, p_j) \in \mathbb{R}^4$ representing the start and end times, the demanded bandwidth and the profit of the interval respectively.

In BAP we wish to select a subset $\mathcal{S} \subseteq \mathcal{I}$, so as to maximize the profit $p(\mathcal{S}) = \sum_{I_j \in \mathcal{S}} p_j$ such that:

$$\forall t \quad \sum_{I_j \in \mathcal{S} : t \in (s_j, t_j)} b_j \leq 1$$

We define an interval I_j as *active* at time t if $t \in (s_j, t_j)$. Intuitively, at any given time the total bandwidth reserved to active intervals at that time must not exceed the bandwidth capacity. We say that a solution \mathcal{S} *fits into bandwidth of α* if $\forall t \sum_{I_j \in \mathcal{S} : t \in (s_j, t_j)} b_j \leq \alpha$.

In SAP, the solution consists of (\mathcal{S}, Y) where $\mathcal{S} \subseteq \mathcal{I}$ and $Y : \mathcal{S} \rightarrow [0, 1]$ is an assignment function such that the following constraints are satisfied:

1. $Y(I_j) + b_j \leq 1$ for every $I_j \in \mathcal{S}$

2. $(s_j, t_j) \cap (s_k, t_k) = \emptyset$ or $(Y(I_j), Y(I_j) + b_j) \cap (Y(I_k), Y(I_k) + b_k) = \emptyset$ for every $I_j, I_k \in \mathcal{S}$ such that $j \neq k$.

Intuitively, it is possible to place each “rectangle” $I_j \in \mathcal{S}$ at a fixed height $Y(I_j)$ such that no rectangle exceeds the bandwidth capacity and every two rectangles do not overlap. Note that every feasible SAP solution denotes a feasible BAP solution.

A DSA problem instance consists of a set $\mathcal{I} = \{I_1, \dots, I_n\}$, where $I_j = (s_j, t_j, b_j) \in \mathbb{R}^3$ represents the start time, the end time, and the height of the rectangle. Throughout the paper we refer to these rectangles as intervals similarly to BAP and SAP. We use the definitions *LOAD* and *cost* as seen in [6] and [9] respectively for the DSA problem. Given a DSA instance \mathcal{I} , $\text{LOAD}(\mathcal{I})$ is the maximal sum of heights of intervals active at some time. Intuitively, $\text{LOAD}(\mathcal{I})$ is the minimal bandwidth capacity such that \mathcal{I} is a feasible BAP solution with such capacity. A DSA solution is a height assignment $Y : \mathcal{I} \rightarrow \mathbb{R}$ for all intervals in \mathcal{I} that satisfies constraint (2) of SAP. In DSA we wish to minimize $\text{cost}(Y) = \max_{I_j \in \mathcal{I}} \{Y(I_j) + b_j\}$.

For any problem instance \mathcal{I} we define $\text{OPT}(\mathcal{I})$ to be the profit of an optimal solution to \mathcal{I} .

2.2 The Local Ratio Technique

We use the Local Ratio technique in our algorithms. It was first developed by Bar-Yehuda and Even [5], extended by Bafna et al. [2], and later expanded by Bar-Yehuda [4]. The Local-Ratio technique originated in minimization problems and later was adapted to maximization problems in the following form, by Bar-Noy et al. [3].

Let $p \in \mathbb{R}^n$ be a profit vector and let F be a set of feasibility constraints on vectors $x \in \mathbb{R}^n$. A vector $x \in \mathbb{R}^n$ is a feasible solution to a given problem instance (F, p) if it satisfies all of the constraints in F . Its profit is the inner product $p \cdot x$. An optimal solution is a feasible solution with the highest profit among feasible solutions. A feasible solution x is an r -approximate solution or an r -approximation if $p \cdot x \geq r \cdot p \cdot x^*$, where x^* is an optimal solution to (F, p) . An algorithm is said to have a performance guarantee of r if it always computes r -approximate solutions.

Theorem 1 (Local Ratio [3]). *Let F be a set of constraints and let p, p_1 and p_2 be profit vectors such that $p = p_1 + p_2$. Then if x is an r -approximation for (F, p_1) and for (F, p_2) then it is an r -approximation for (F, p) .*

Proof. Let x_1^*, x_2^* and x^* be optimal solutions to $(F, p_1), (F, p_2)$ and (F, p) respectively. Thus $p \cdot x = (p_1 + p_2) \cdot x = p_1 \cdot x + p_2 \cdot x \geq r \cdot p_1 \cdot x_1^* + r \cdot p_2 \cdot x_2^* = r(p_1 \cdot x_1^* + p_2 \cdot x_2^*) \geq r(p_1 \cdot x^* + p_2 \cdot x^*) = r \cdot (p_1 + p_2) \cdot x^* = r \cdot p \cdot x^*$. \square

The theorem is formulated for maximization problems. Similarly it holds for minimization problems as well.

2.3 Combining two approximations

Lemma 1. *Let \mathcal{I} be a problem instance and $\mathcal{I} = \mathcal{I}_1 \cup \mathcal{I}_2$ be a disjoint partition of \mathcal{I} . Let $\mathcal{S}_1, \mathcal{S}_2$ be r_1, r_2 -approximations to $\mathcal{I}_1, \mathcal{I}_2$ respectively. Then the most profitable solution among $\mathcal{S}_1, \mathcal{S}_2$ is an $(r_1^{-1} + r_2^{-1})^{-1}$ -approximation to \mathcal{I} .*

Proof. Let \mathcal{S}^* be an optimal solution to \mathcal{I} . Thus either $p(\mathcal{S}^* \cap \mathcal{I}_1) \geq r_1^{-1}/(r_1^{-1} + r_2^{-1})p(\mathcal{S}^*) = r_1^{-1}/(r_1^{-1} + r_2^{-1})\text{OPT}(\mathcal{I})$ and thus $p(\mathcal{S}_1) \geq r_1\text{OPT}(\mathcal{I}_1) \geq r_1p(\mathcal{S}^* \cap \mathcal{I}_1) \geq (r_1^{-1} + r_2^{-1})^{-1}\text{OPT}(\mathcal{I})$ or $p(\mathcal{S}^* \cap \mathcal{I}_2) \geq r_2^{-1}/(r_1^{-1} + r_2^{-1})p(\mathcal{S}^*)$ and by similar reasoning $p(\mathcal{S}_2) \geq (r_1^{-1} + r_2^{-1})^{-1}\text{OPT}(\mathcal{I})$. \square

This lemma holds for any problem in which a sub-solution of a feasible solution is also a feasible solution, particularly for BAP and SAP. We will use this lemma for partitioning a general problem instance into two sub-problems of “wide” and “narrow” intervals, computing an approximation for them and returning the most profitable solution as the final approximation.

3 BAP approximation algorithm

Fact 1. For a BAP instance in which all the bandwidths are higher than $0 < \epsilon < 1$, there is a polynomial-time algorithm of complexity $O(n^{2/\epsilon+O(1)})$ that solves the problem optimally. A dynamic-programming algorithm is presented in [7].

Lemma 2. For a BAP instance in which all the bandwidths are less than or equal to $0 < \epsilon < \alpha$, for a fixed $\alpha \leq 1$, there is a polynomial-time algorithm that computes an $((\alpha - \epsilon)^{-1} + 1)^{-1}$ -approximation that fits into bandwidth of α .

Proof. We use the unified approach for BAP presented in [3] based on the Local Ratio technique, with a profit function that assures both approximation guarantee and the solution fitting into bandwidth of α . The algorithm is denoted by $LR(\mathcal{I}, p)$ where \mathcal{I} is the BAP instance and p is the profit function. The initial profit function is defined by $p(I_j) = p_j$. $LR(\mathcal{I}, p)$ performs the following operations:

Algorithm 1 $LR(\mathcal{I}, p)$

```

1: if  $\mathcal{I} = \emptyset$  then
2:   return  $\emptyset$ 
3: end if
4: if exists  $I_j \in \mathcal{I}$  such that  $p(I_j) \leq 0$  then
5:   return  $LR(\mathcal{I} \setminus \{I_j\}, p)$ 
6: end if
7:  $I_f \leftarrow$  interval in  $\mathcal{I}$  such that  $t_f = \min\{t_j | I_j \in \mathcal{I}\}$ 
8:  $C \leftarrow \{I_j \in \mathcal{I} | (s_j, t_j) \cap (s_f, t_f) \neq \emptyset\} \setminus \{I_f\}$ 
9:  $\mathcal{S} \leftarrow LR(\mathcal{I}, p - p')$ 
10: if  $\sum_{I_j \in \mathcal{S} \cap C} b_j \geq \alpha - \epsilon$  then
11:   return  $\mathcal{S}$ 
12: else
13:   return  $\mathcal{S} \cup \{I_f\}$ 
14: end if

```

Where p' is the local profit function defined by:

$$p'(I_j) = p_f \cdot \begin{cases} 1 & j = f \\ \frac{b_j}{\alpha - \epsilon} & I_j \in C \\ 0 & \text{otherwise} \end{cases}$$

A graphical presentation of p' is presented in Figure 1. Define $r \triangleq ((\alpha - \epsilon)^{-1} + 1)^{-1}$. We now prove two properties of the algorithm.

r -approximation guarantee. By the induction hypothesis, the computed solution in Step 9 \mathcal{S} is an r -approximation with respect to $p - p'$. If condition 10 holds then $p'(\mathcal{S}) \geq p'(\mathcal{S} \cap C) \geq p_f(\alpha - \epsilon)^{-1} \sum_{I_j \in \mathcal{S} \cap C} b_j \geq p_f$. Otherwise $p'(\mathcal{S} \cup \{I_f\}) \geq p'(I_f) = p_f$. In both cases, the returned

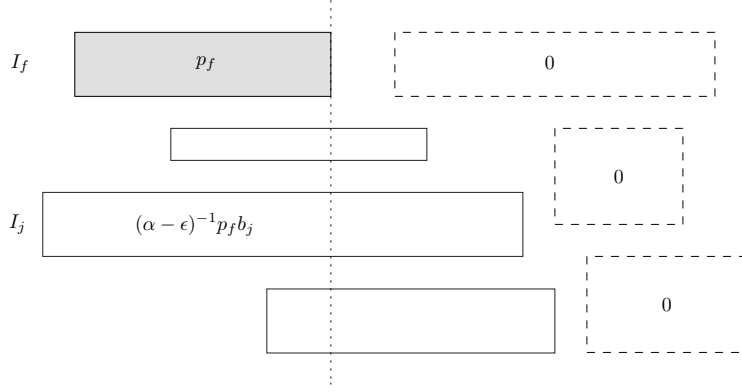


Figure 1: The grey interval is I_f , below are C intervals and to the right is the rest of \mathcal{I} .

solution gains at least profit of p_f with respect to p' . In addition, the optimal solution to \mathcal{I} with respect to p' is of profit at most $p_f((\alpha - \epsilon)^{-1} + 1) = p_f/r$ because $C \cup \{I_f\}$ induces a clique in the time-intersection interval graph. Thus, the returned solution is an r -approximation with respect to p' and thus, by the Local Ratio theorem, it is an r -approximation with respect to p .

Fitting into bandwidth of α . By induction hypothesis, \mathcal{S} fits into bandwidth of α . If \mathcal{S} is returned in Step 11, then this property is preserved. Otherwise, there is at least $\epsilon \geq b_f$ free bandwidth in order to insert I_f into \mathcal{S} so the returned solution $\mathcal{S} \cup \{I_f\}$ fits into bandwidth of α .

As described in [3] the algorithm can be implemented in $O(n \log n)$. □

Corollary 1. *For a BAP instance in which all the bandwidths are less than or equal to $0 < \epsilon < 1/k$, for a fixed $k \geq 1$, there is a polynomial-time algorithm that computes an $((1/k - \epsilon)^{-1} + 1)^{-1}$ -approximation that fits into bandwidth of $1/k$.*

3.1 Iterative approximation of narrow intervals

Lemma 3. *For every $\delta > 0$ exists $0 < \epsilon < 1$ such that for a BAP instance in which all the bandwidths are less than or equal to ϵ , there is a polynomial-time $(1 - e^{-1} - \delta)$ -approximation algorithm.*

Proof. Let $\delta > 0$ be a constant. Let k and $\epsilon < 1/k$ be constants determined later. Suppose we have a BAP instance in which all the bandwidths are less than or equal to ϵ . We apply Corollary 1 with ϵ and k , for k times iteratively for computing k solutions $(\mathcal{S}_{i=1}^k)$, each iteration computes an approximation from the remaining intervals $(\mathcal{I}_{i=1}^k)$ that fits into bandwidth of $1/k$ and accumulates a partial solution $(\mathcal{F}_{i=1}^k)$. The final approximation becomes a sum of a series of all approximations, with respect to $\text{OPT}(\mathcal{I})$, accumulated throughout the iterations. We will see that asymptotically the limit of this sum is $1 - e^{-1}$. On a BAP instance \mathcal{I} , the algorithm performs the following operations:

We denote by $LR(\mathcal{I}, p)$ the algorithm of Corollary 1. We now prove the approximation guarantee of the algorithm. Let $r_i \triangleq p(\mathcal{S}_i)/\text{OPT}(\mathcal{I}_i)$ be the *relative* approximation gained in the i -th iteration.

Claim 1. *For every $0 \leq i \leq k$,*

$$\text{OPT}(\mathcal{I}_{i+1}) \geq (\prod_{j=1}^i (1 - r_j)) \text{OPT}(\mathcal{I}) \geq \text{OPT}(\mathcal{I}) - p(\mathcal{F}_{i+1}).$$

Algorithm 2 BAP Iterative approximation of narrow intervals

```

1:  $\mathcal{I}_1 \leftarrow \mathcal{I}, \mathcal{F}_1 \leftarrow \emptyset, p$  is the initial profit function defined by  $p(I_j) = p_j$ 
2: for  $i = 1$  to  $k$  do
3:    $\mathcal{S}_i \leftarrow LR(\mathcal{I}_i, p)$ 
4:    $\mathcal{F}_{i+1} \leftarrow \mathcal{F}_i \cup \mathcal{S}_i$ 
5:    $\mathcal{I}_{i+1} \leftarrow \mathcal{I}_i \setminus \mathcal{S}_i$ 
6: end for
7: return  $\mathcal{F}_{k+1}$ 

```

Proof. We prove the claim by induction on i . The induction case ($i = 0$) is trivial. For the inductive step, let \mathcal{S}_i^* be an optimal solution to \mathcal{I}_i . Thus

$$p(\mathcal{S}_i^* \setminus \mathcal{S}_i) = p(\mathcal{S}_i^*) - p(\mathcal{S}_i^* \cap \mathcal{S}_i) \geq p(\mathcal{S}_i^*) - p(\mathcal{S}_i) = (1 - r_i)\text{OPT}(\mathcal{I}_i).$$

By the induction hypothesis, $\text{OPT}(\mathcal{I}_i) \geq (\prod_{j=1}^{i-1} (1 - r_j))\text{OPT}(\mathcal{I})$, and thus $p(\mathcal{S}_i^* \setminus \mathcal{S}_i) \geq (\prod_{j=1}^i (1 - r_j))\text{OPT}(\mathcal{I})$. Because $\mathcal{S}_i^* \setminus \mathcal{S}_i$ is a feasible solution to \mathcal{I}_{i+1} we have

$$\text{OPT}(\mathcal{I}_{i+1}) \geq p(\mathcal{S}_i^* \setminus \mathcal{S}_i) \geq (\prod_{j=1}^i (1 - r_j))\text{OPT}(\mathcal{I}).$$

Because $p(\mathcal{F}_{i+1}) = p(\mathcal{F}_i) + r_i\text{OPT}(\mathcal{I}_i)$ and by the induction hypothesis,

$$\begin{aligned} \text{OPT}(\mathcal{I}) - p(\mathcal{F}_{i+1}) &= \text{OPT}(\mathcal{I}) - p(\mathcal{F}_i) - r_i\text{OPT}(\mathcal{I}_i) \\ &\leq (\prod_{j=1}^{i-1} (1 - r_j))\text{OPT}(\mathcal{I}) - r_i(\prod_{j=1}^{i-1} (1 - r_j))\text{OPT}(\mathcal{I}) \\ &= (\prod_{j=1}^i (1 - r_j))\text{OPT}(\mathcal{I}). \end{aligned}$$

□

From Corollary 1 it follows that each $r_i \geq ((1/k - \epsilon)^{-1} + 1)^{-1} \triangleq f(k, \epsilon)$. Therefore from Claim 1 it follows that $\text{OPT}(\mathcal{I}) - p(\mathcal{F}_{k+1}) \leq (\prod_{j=1}^k (1 - r_j))\text{OPT}(\mathcal{I}) \leq (1 - f(k, \epsilon))^k \text{OPT}(\mathcal{I})$. Thus $p(\mathcal{F}_{k+1}) \geq \text{OPT}(\mathcal{I})(1 - (1 - f(k, \epsilon))^k)$. Because $\lim_{k \rightarrow \infty} (\lim_{\epsilon \rightarrow 0} (1 - f(k, \epsilon))^k) = e^{-1}$, we choose sufficiently large fixed k and then sufficiently small ϵ such that $(1 - f(k, \epsilon))^k \leq e^{-1} + \delta$. Thus the approximation guarantee is at least $(1 - e^{-1} - \delta)\text{OPT}(\mathcal{I})$. Note that \mathcal{F}_{k+1} is a feasible solution because it is composed of k solutions, each of them fits into bandwidth of $1/k$.

The running time of the algorithm is polynomial because for a given δ , all parameters except n are constants. □

We now combine Fact 1 for wide intervals and Lemma 3 for sufficiently narrow intervals for getting the final approximation.

Theorem 2. *For every $\delta > 0$, there is a polynomial-time $((2 + 1/(e - 1))^{-1} - \delta)$ -approximation algorithm for BAP.*

Proof. Let $\delta > 0$ be a constant. Let δ' be a constant determined later. By Lemma 3, let ϵ' be a suitable constant in relation to δ' . We partition the BAP instance \mathcal{I} into two disjoint sets of wide intervals $\mathcal{I}_w = \{I_j \in \mathcal{I} | b_j > \epsilon'\}$ and narrow intervals $\mathcal{I}_n = \mathcal{I} - \mathcal{I}_w$. For \mathcal{I}_w we compute an optimal solution, i.e. 1-approximation using Fact 1. For \mathcal{I}_n we compute an $(1 - e^{-1} - \delta')$ -approximation using Lemma 3. Thus from Lemma 1 the most profitable one among the two solutions is an $(1 + (1 - e^{-1} - \delta')^{-1})^{-1}$ -approximation to \mathcal{I} . We choose sufficiently small δ' such that $(1 + (1 - e^{-1} - \delta')^{-1})^{-1} \geq (1 + (1 - e^{-1})^{-1})^{-1} - \delta = (2 + 1/(e - 1))^{-1} - \delta$.

The running time of the algorithm is polynomial because for a given δ , all parameters except n are constants. □

4 SAP approximation algorithm

Definition 1. A SAP instance is c -restricted if it has an additional constraint that at any time, the number of active intervals at that time must not exceed c .

Fact 2. For a c -restricted SAP instance in which all the bandwidths are integers and the bandwidth capacity is an integer K , there is a pseudo-polynomial-time algorithm of complexity $O(n(nK)^c)$ that solves the problem optimally. A dynamic-programming algorithm is presented in [8].

The assumption that all the bandwidths are integers is used for limiting the number of possible assignments of an interval from continuum to K , because in this case there always exists an optimal solution in which intervals are assigned to integer heights.

Lemma 4. For a SAP instance in which all the bandwidths are higher than $0 < \epsilon < 1$ there is a polynomial-time algorithm that computes an optimal solution.

Proof. Let (\mathcal{S}^*, Y^*) be an optimal solution to \mathcal{I} such that each interval in \mathcal{S} is supported by some interval underneath it, as if gravity was applied to it. Thus each interval in \mathcal{S}^* is assigned to a height that is the sum of heights of a subset of \mathcal{I} . Because all bandwidths are higher than ϵ , the size of the subset is at most $z = \lfloor 1/\epsilon \rfloor$. The number of such heights is bounded by the number of subsets of \mathcal{I} of size at most z , that is $O(n^z)$. We use a dynamic programming algorithm similar to Fact 2 where instead of an integer height grid of size K we use a non-homogeneous height grid mentioned above of size $O(n^z)$ to compute an optimal solution to \mathcal{I} as z -restricted SAP instance. Thus the complexity of the algorithm is $O(n(n^z)^z) = O(n^{1/\epsilon^2+1})$. \square

We use the following results for DSA in our next algorithm for sufficiently narrow intervals.

Fact 3. For DSA problem there is a polynomial-time algorithm that computes a solution of cost at most $3L$, where L is the LOAD of a problem instance. An algorithm of complexity $O(n \log n)$ is presented in [9].

Fact 4. For DSA problem in which all the heights are integers, there is a polynomial-time algorithm that computes a solution of cost at most $(1 + O((h_{\max}/L)^{1/7}))L$, where L and h_{\max} are the LOAD and maximal height of an interval in a problem instance respectively. An algorithm is presented in [6].

Lemma 5. For every $\delta > 0$ exists $0 < \epsilon < 1$ such that for a SAP problem in which all the bandwidths are less than or equal to ϵ , there is a polynomial-time $(1 - e^{-1} - \delta)$ -approximation algorithm.

Proof. Let $\delta > 0$ be a constant. Let $C \geq 1, R \leq 1$ be constants such that the algorithm of Fact 4 computes a DSA solution of cost at most $(1 + C(h_{\max}/L)^{1/7})L$ for every $h_{\max}/L \leq R$. Let δ', ϵ be constants determined later. Suppose we have a SAP instance in which all the bandwidths are less than or equal to ϵ . On a problem instance \mathcal{I} the algorithm performs the following operations:

If the condition of Step 2 is true, (\mathcal{S}, Y) is a feasible SAP solution. In case the condition is false, we use Fact 4 after standard scaling for DSA to integer bandwidths, that asymptotically does not affect the computed cost. Because the maximal bandwidth of an interval in \mathcal{S} is at most ϵ and the LOAD of \mathcal{S} is at least $1/3$, the ratio between the former and the later is less than or equal to 3ϵ . $\text{LOAD}(\mathcal{S}) \leq 1 - \beta$, thus the computed assignment Y in Step 5 satisfies $\text{cost}(Y) \leq (1 + C(3\epsilon)^{1/7})(1 - \beta) = 1$. Therefore, the returned solution is a feasible SAP solution. Finally, in both cases the returned solution is feasible.

Algorithm 3 SAP narrow intervals approximation

- 1: Apply Lemma 3 to compute a $(1 - e^{-1} - \delta')$ -approximation \mathcal{S} to \mathcal{I} as BAP instance with bandwidth capacity $1 - \beta = (1 + C(3\epsilon)^{1/7})^{-1}$
 - 2: **if** $\text{LOAD}(\mathcal{S}) \leq 1/3$ **then**
 - 3: Compute an assignment Y for \mathcal{S} using Fact 3
 - 4: **else**
 - 5: Compute an assignment Y for \mathcal{S} using Fact 4
 - 6: **end if**
 - 7: **return** (\mathcal{S}, Y)
-

The profit of \mathcal{S} , $p(\mathcal{S}) \geq (1 - e^{-1} - \delta')\text{OPT}(\mathcal{I}, 1 - \beta)$, where $\text{OPT}(\mathcal{I}, 1 - \beta)$ is the optimal profit that can be achieved for a BAP instance \mathcal{I} with bandwidth capacity $1 - \beta$. $\text{OPT}(\mathcal{I}, 1 - \beta)$ for \mathcal{I} as BAP instance is at least as profitable as $\text{OPT}(\mathcal{I}, 1 - \beta)$ for \mathcal{I} as SAP instance, thus the computed approximation in Step 1 is also an approximation for SAP instance \mathcal{I} . Because $\epsilon < \beta$ for sufficiently small ϵ ($\epsilon < 1/2$ is sufficient), $\text{OPT}(\mathcal{I}, 1 - \beta) \geq (1 - 2/\lfloor 1/\beta \rfloor)\text{OPT}(\mathcal{I})$ and thus $p(\mathcal{S}) \geq (1 - e^{-1} - \delta')(1 - 2/\lfloor 1/\beta \rfloor)\text{OPT}(\mathcal{I})$.

Therefore we choose sufficiently small δ' and then sufficiently small ϵ such that

1. $3\epsilon \leq R$, for successful application of Fact 4 in Step 5.
2. $\epsilon < \beta$, for assuring that $\text{OPT}(\mathcal{I}, 1 - \beta)$ is asymptotically close to $\text{OPT}(\mathcal{I})$.
3. $\epsilon < \epsilon'$, where ϵ' is a suitable constant in relation to δ' as described in Lemma 3, for successful application of Lemma 3 in Step 1.
4. $(1 - e^{-1} - \delta')(1 - 2/\lfloor 1/\beta \rfloor) \geq 1 - e^{-1} - \delta$, for achieving the wanted approximation.

The running time of the algorithm is polynomial because for a given δ , all parameters except n are constants. □

Theorem 3. *For every $\delta > 0$, there is a polynomial-time $((2 + 1/(e - 1))^{-1} - \delta)$ -approximation algorithm for SAP.*

Proof. The proof is very similar to proof of Theorem 2, because Lemmas 4 and 5 are equivalent to Fact 1 and Lemma 3 respectively for SAP. □

Acknowledgment

We would like to thank Dror Rawitz for his helpful suggestions.

References

- [1] E. M. Arkin and E. B. Silverberg. Scheduling jobs with fixed start and end times. *Discrete Applied Mathematics*, 18:1–8, 1987.
- [2] V. Bafna, P. Berman, and T. Fujito. A 2-approximation algorithm for the undirected feedback vertex set problem. *SIAM Journal on Discrete Mathematics*, 12(3):289–297, 1999.
- [3] A. Bar-Noy, R. Bar-Yehuda, A. Freund, J. Naor, and B. Shieber. A unified approach to approximating resource allocation and scheduling. *Journal of the ACM*, 48(5):1069–1090, 2001.

- [4] R. Bar-Yehuda. One for the price of two: A unified approach for approximating covering problems. *Algorithmica*, 27(2):131–144, 2000.
- [5] R. Bar-Yehuda and S. Even. A local-ratio theorem for approximating the weighted vertex cover problem. *Annals of Discrete Mathematics*, 25:27–46, 1985.
- [6] A. L. Buchsbaum, H. Karloff, C. Kenyon, N. Reingold, and M. Thorup. OPT versus LOAD in dynamic storage allocation. In *35th ACM Symposium on the Theory of Computing*, pages 556–564, 2003.
- [7] G. Calinescu, A. Chakrabarti, H. J. Karloff, and Y. Rabani. Improved approximation algorithms for resource allocation. In *9th International Integer Programming and Combinatorial Optimization Conference*, volume 2337 of *LNCS*, pages 401–414, 2002.
- [8] B. Chen, R. Hassin, and M. Tzur. Allocation of bandwidth and storage. *IIE Transactions*, 34:501–507, 2002.
- [9] J. Gergov. Algorithms for compile-time memory optimization. In *10th Annual Symposium on Discrete Algorithms*, pages 907–908, 1999.
- [10] S. Leonardi, A. Marchetti-Spaccamela, and A. Vitaletti. Approximation algorithms for bandwidth and storage allocation problems under real time constraints. In *20th Conference on Foundations of Software Technology and Theoretical Computer Science (FSITCS)*, 20:409–420, 2000.
- [11] C. Phillips, R. N. Uma, and J. Wein. Off-line admission control for general scheduling problems. *Journal of Scheduling*, 3:365–381, 2000.