Approximating Interval Scheduling Problems with Bounded Profits

Israel Beniaminy[*] Zeev Nutov[†] Meir Ovadia[‡]

Abstract

We consider the *Generalized Scheduling Within Intervals* (GSWI) problem: given a set J of jobs and a set $\mathcal{I}$ of intervals, where each job $j \in J$ has in interval $I \in \mathcal{I}$ length (processing time) $\ell_{j,I}$ and profit $p_{j,I}$, find the highest-profit feasible schedule. The best approximation ratio known for GSWI is $(1/2 - \varepsilon)$. We give a $(1 - 1/e - \varepsilon)$-approximation scheme for GSWI with bounded profits, based on the work by Chuzhoy, Rabani, and Ostrovsky [5], for the $\{0,1\}$-profit case. We also consider the *Scheduling Within Intervals* (SWI) problem, which is a particular case of GSWI where for every $j \in J$ there is a unique interval $I = I_j \in \mathcal{I}$ with $p_{j,I} > 0$. We prove that SWI is (weakly) NP-hard even if the *stretch factor* (the maximum ratio of job's interval size to its processing time) is arbitrarily small, and give a polynomial-time algorithm for bounded profits and stretch factor < 2.

Key-words. Interval scheduling, Approximation algorithm.

1 Introduction

We consider the following problem:

Generalized Scheduling Within Intervals (GSWI):
Instance: A set J of jobs and a set $\mathcal{I}$ of intervals, where each job $j \in J$ has in interval $I \in \mathcal{I}$ length $\ell_{j,I}$ and profit $p_{j,I}$, and each interval $I \in \mathcal{I}$ is given by $[r_I, d_I)$.
Objective: Find a maximum profit feasible schedule.

[*]ClickSoftware Technologies, israel@clicksoftware.com

[†]The Open University of Israel, nutov@openu.ac.il

[‡]The Open University of Israel, meiro@cadance.com

More precisely, a schedule S consists of a subset $J' \subseteq J$ of jobs, and for every $j \in J'$: an assignment to an interval $I(j) \in \mathcal{I}$ and a start time s_j, so that $[s_j, s_j + \ell_{j,I(j)}) \subseteq I(j)$. A schedule is feasible if the intervals $[s_j, s_j + \ell_{j,I(j)})$, $j \in J'$, are pairwise disjoint. The profit of such schedule is $p(S) = \sum_{j \in J'} p_{j,I(j)}$. We sometimes use S to denote only the corresponding set J' of jobs, if this does not cause ambiguity. Throughout this paper we assume that all the problem parameters are integers. Let $n = |J|$, $m = |\mathcal{I}|$, and let $P = \max_{j,I} p_{j,I}$.

GSWI and related problems have been used to model many applications of scheduling problems, including service and delivery, project planning, communication scheduling, space-mission-planning and optimization of production lines, c.f., [2, 1, 4, 7]. The following three particular cases of GSWI were studied extensively.

- *Max-Profit Generalized Assignment* (MAX-GAP):
  the intervals in $\mathcal{I}$ are pairwise disjoint.
- *Scheduling Within Intervals* (SWI):
  for every $j \in J$ exactly one interval $I \in \mathcal{I}$ has $p_{j,I} > 0$.
- *Job Interval Selection Problem* (JISP):
  for every pair $j \in J, I \in \mathcal{I}$, either $p_{j,I} = 0$ or $\ell_{j,I} = |I|$.

Each one of these particular cases is strongly NP-hard [8]. Bar-Noy et al. [2] gave an approximation scheme for SWI (in fact, also for GSWI) with ratio of $1/2 - \varepsilon$, see also [1], and a faster algorithm due to Berman and DasGupta [4]. MAX-GAP admits a $(1 - 1/e)$-approximation algorithm [7], see also [9]. Recently, Feige and Vondrak [6] showed that MAX-GAP admits a ratio better than $(1 - 1/e)$. However, MAX-GAP is much "easier" than GSWI, and the algorithms in [7, 9, 6] do not extend even to very special instances (e.g., $\{0, 1\}$ profits and unit lengths) of SWI/JISP. Chuzhoy, Rabani, and Ostrovsky [5] considered JISP with $\{0, 1\}$-profits, and gave for this case a randomized approximation scheme with ratio $(1 - 1/e - \varepsilon)$. We extend this result to arbitrary *bounded* profits. Our algorithm for GSWI uses as a subroutine an arbitrary constant ratio approximation algorithm SC for GSWI; let $1/\alpha$ be its approximation ratio and $Q = Q(n, m)$ its running time (e.g., the algorithm of [4] with $\alpha = 3$ has running time $Q(n, m) = O(n^2 m)$). A packing-type linear program is of the form $\max\{p \cdot y : yA \leq b, y \geq 0\}$, where A is an $n' \times m'$ $\{0, 1\}$-matrix, and b is an m' $\{0, 1\}$ vector. We assume that the time required to solve such program is $L(n', m')$.

Theorem 1.1 GSWI *admits a $(1 - 1/e - \varepsilon)$-approximation algorithm for any $\varepsilon > 0$ and constant $P = \max_{j,I} p_{j,I}$. The running time of the algorithm is $O(n \cdot Q(n, m) \cdot \ln(P/\varepsilon) + (nm)^{q+1} L((nm)^{q+1}, n))$, where $q = 6k^{k \ln k + 3}$ for $k = \lceil (P + 2\alpha + 1)/\varepsilon \rceil = O(P/\varepsilon)$, and*

$L(n', m')$ is the time required to solve a packing type linear program with n' variables and m' constraints.

For $P = 1$ we get the algorithm of [5]. The running time, although polynomial for any $\varepsilon > 0$, is not practical. We leave an open question whether the time complexity can be reduced to be polynomial in the input size and $1/\varepsilon$, or whether GSWI admits a $(1 - 1/e)$-approximation algorithm. Such a better algorithm is known for MAX-GAP, but is not known even for the $\{0, 1\}$-profit JISP/SWI.

Recall that in SWI every job $j \in J$ can be scheduled in a unique interval $I_j \in \mathcal{I}$ (as j has profit 0 in the other intervals); let j have length ℓ_j and profit p_j in I_j. Note that GSWI includes the "multiple machine" version of SWI. We consider SWI with bounded profits and small *stretch factor*, which is $\max_{j\in J} |I_j|/\ell_j$. Formally, let θ-SWI be the restriction of SWI to instances with $\max_{j\in J} |I_j|/\ell_j < \theta$, where $\theta > 1$. Berman and DasGupta [4] gave a pseudo-polynomial algorithm for θ-SWI with running time $O(\theta T n \log\log T)$ and with approximation ratio $1/2 + 1/(2^{a+2} - 4 - 2a)$, where $T = \max_j d_j$ and a is the largest integer strictly smaller than θ; this was used to derive an approximation scheme with ratio $(1/2-\varepsilon)+1/(2^{a+2}-4-2a)$ and running time $O(n^2/\varepsilon)$. For 2-SWI $a = 1$ and thus the approximation ratio is $1 - \varepsilon$ (in [2] and in [4] 2-SWI was mistakenly mentioned to be in P, but this is so only for bounded profits). We prove:

Theorem 1.2 *θ-SWI is NP-hard for any $\theta > 1$ even if $p_j = \ell_j$ for every job $j \in J$ and all the time windows have the same length. 2-SWI can be solved in $O(\min\{n\mathcal{P}, rn^r \log n\})$ time, where $\mathcal{P} = \sum_{j\in J} p_j$ and r is the number of distinct profits.*

Theorems 1.1 and 1.2 are proved in Sections 2 and 3, respectively.

A preliminary version of this paper is [3].

2 Proof of Theorem 1.1

Our algorithm extends the algorithm of [5] for $\{0, 1\}$-profit JISP/SWI, to GSWI with any profits bounded by a constant P. Here is a high-level description of the algorithm. Let $[0, T)$ be the *timeline*, namely, the smallest interval that contains all the intervals in $\mathcal{I}$. The algorithm has two phases. In Phase I, the algorithm computes a partition $\mathcal{B}$ of $[0, T)$ into blocks (intervals) and:

(i) A schedule S^I within a subset $\mathcal{B}^I \subseteq \mathcal{B}$ of blocks (no job intersects a boundary of a block);

(ii) Subsets $J^{II} \subseteq J \setminus S^I$ of jobs and $B^{II} \subseteq \mathcal{B} \setminus \mathcal{B}^I$ of blocks.

The computed partition $\mathcal{B}$ has the property that the restriction "no job intersects a boundary of a block in $\mathcal{B}$" decreases the optimum by a small amount (if ε is small). We also have an upper bound on the profit of jobs from J^{II} scheduled by any optimal solution within any block in $\mathcal{B}^{II}$. In Phase II, the algorithm schedules jobs from J^{II} within blocks of $\mathcal{B}^{II}$ using an LP-relaxation created by enumerating all feasible schedules in each such block. LP-rounding gives a feasible schedule with expected approximation ratio $(1 - 1/e)$, and the algorithm is derandomized using the method of conditional expectations. The main differences between our algorithm and that of [5] is that in Phase I our algorithm partitions the timeline according to the *profit* of jobs we can schedule in each block, and that at Phase II we use a more general linear program.

Formally, let $\mathrm{OPT}_{\mathcal{B}}(J)$ be any optimal schedule of jobs from J under the constraint that no job's schedule intersects the boundary of a block from $\mathcal{B}$, and let $\mathsf{opt}_{\mathcal{B}}(J) = p(\mathrm{OPT}_{\mathcal{B}}(J))$ be its profit; for brevity $\mathrm{OPT}(J) = \mathrm{OPT}_{\{[0,T]\}}(J)$ and $\mathsf{opt}(J) = p(\mathrm{OPT}(J))$. Given a set J' of jobs and a block B, let $\mathrm{SC}(J', B)$ be any $1/\alpha$-approximation algorithm for scheduling jobs from J' within the block B, and let us denote by $Q = Q(n, m)$ its running time. As was mentioned, we may substitute $\alpha = 3$ and $Q(n, m) = O(n^2 m)$. Given $\varepsilon > 0$ let $k = \lceil (P + 2\alpha + 1)/\varepsilon \rceil$. We prove:

Lemma 2.1 GSWI *admits an algorithm that for any* $\varepsilon > 0$ *computes in time* $O(n \cdot Q(n, m) \cdot \ln(P/\varepsilon))$ *a partition* $\mathcal{B}$ *of* $[0, T)$ *into at most* $n\varepsilon$ *blocks, a schedule* S^I *within a subset* $\mathcal{B}^I \subseteq \mathcal{B}$ *of blocks, and subsets* $J^{II} \subseteq J \setminus S^I$ *of jobs and* $\mathcal{B}^{II} \subseteq \mathcal{B} \setminus \mathcal{B}^I$ *of blocks such that:*

(i) $p(S^I) + \mathsf{opt}_{\mathcal{B}^{II}}(J^{II}) \geq (1 - \varepsilon)\mathsf{opt}(J)$.

(ii) *In each block* $B \in \mathcal{B}^{II}$*, any optimal solution schedules at most* $2\alpha \cdot k^{k \ln k + 3}$ *jobs from* J^{II}.

An instance of GSWI is $(\mathcal{B}, q)$*-restricted*, where $\mathcal{B}$ is a set of pairwise disjoint blocks in $[0, T)$ and q is an integer, if there exists an optimal solution that schedules all its jobs within the blocks of $\mathcal{B}$ with at most q jobs per block.

Lemma 2.2 $(\mathcal{B}, q)$*-restricted* GSWI *admits a* $(1-1/e)$*-approximation algorithm with running time* $O(|\mathcal{B}|(nm)^q \cdot L(|\mathcal{B}|(nm)^q, n + |\mathcal{B}|))$.

Lemmas 2.1, 2.2 easily imply Theorem 1.1. Execute the algorithm as in Lemma 2.1 to compute $\mathcal{B}^I, \mathcal{B}^{II}, S^I, J^{II}$ as in the lemma. Then apply Lemma 2.2 on the $(\mathcal{B}^{II}, q)$-restricted GSWI instance with $q = 2\alpha \cdot k^{k \ln k + 3}$ and the set of jobs J^{II} to compute a schedule S^{II}. The running time is as claimed. Clearly, $S^I \cup S^{II}$ is a feasible solution, since $S^{II} \subseteq J \setminus S^I$ and since no block in $\mathcal{B}^I$ intersects a block in $\mathcal{B}^{II}$. The approximation ratio is as claimed since:

$$\begin{aligned} p(S^I) + p(S^{II}) &\geq p(S^I) + (1 - 1/e)\mathsf{opt}_{\mathcal{B}^{II}}(J^{II}) \\ &\geq (1 - 1/e)\left(p(S^I) + \mathsf{opt}_{\mathcal{B}^{II}}(J^{II})\right) \\ &\geq (1 - 1/e)(1 - \varepsilon)\mathsf{opt}(J) \geq (1 - 1/e - \varepsilon)\mathsf{opt}(J)\ . \end{aligned}$$

2.1 Proof of Lemma 2.1

In the following procedure PartitionTimeLine, in iteration i, $\mathcal{B}_i$ is the set of blocks partitioning the timeline $[0, T)$ and S_i is the schedule (or the set of jobs scheduled) within the blocks of $\mathcal{B}_i$; only these jobs are available for scheduling in the next iteration. Eventually, the algorithm returns a partition $\mathcal{B}$ of $[0, T)$ into blocks, a schedule S^I within a subset $\mathcal{B}^I \subseteq \mathcal{B}$ of blocks, and subsets $J^{II} \subseteq J \setminus S^I$ of jobs and $\mathcal{B}^{II} \subseteq \mathcal{B} \setminus \mathcal{B}^I$ of blocks.

Procedure PartitionTimeLine(J,ε)

```
Initialization: $i \leftarrow 1$; $\mathcal{B}_0 \leftarrow \{[0,T)\}$; $S_0 \leftarrow J$; $k \leftarrow \lceil (P + 2\alpha + 1)/\varepsilon \rceil$.
Loop
  $S_i \leftarrow \emptyset$, $\mathcal{B}_i \leftarrow \mathcal{B}_{i-1}$
  For every block $B \in \mathcal{B}_i$ in ascending time order do:
      If $p(\mathrm{SC}(S_{i-1} \setminus S_i, B)) \geq k^{i+2}$ then do:
        - $S_i \leftarrow S_i \cup \mathrm{SC}(S_{i-1} \setminus S_i, B)$;
        - In ascending time order, scan the jobs scheduled by SC in $B$ and partition $B$
          into blocks, each with largest possible profit $\leq k^{i+2}$, and add this partition to $\mathcal{B}_i$.
  EndFor
  Termination Condition 1: If $i = \lceil k \ln k \rceil$ then do:
                             $S^I \leftarrow \emptyset$; $B^I \leftarrow \emptyset$;
                             $J^{II} \leftarrow J \setminus S_i$; $\mathcal{B}^{II} \leftarrow \mathcal{B}_i$; STOP.
  Termination Condition 2: If $p(S_i) \geq (1 - 1/k)p(S_{i-1})$ then do:
                             $S^I \leftarrow S_i$;
                             $\mathcal{B}^I \leftarrow$ the blocks in $\mathcal{B}_i$ containing jobs from $S^I$;
                             $J^{II} \leftarrow J \setminus S_{i-1}$; $\mathcal{B}^{II} \leftarrow \mathcal{B}_{i-1} \setminus \mathcal{B}^I$; STOP.
  Else (Termination Conditions 1,2 do not apply) $i \leftarrow i + 1$.
EndLoop
```

The following statement uses only the fact that the number of iterations in Partition-

TimeLine is $\lceil k \ln k \rceil \le k^2$, and it is independent of the algorithm SC used.

Lemma 2.3 *$|\mathcal{B}| \le |\mathrm{OPT}(J)|/k$ holds for the partition $\mathcal{B}$ produced by* PartitionTimeLine*. Thus $\mathsf{opt}_{\mathcal{B}}(J) \ge (1 - P/k)\mathsf{opt}(J)$.*

Proof: The number of new blocks created in iteration i is at most $p(S_i)/k^{i+2} \le p(S_i)/k^3$. Each new block eliminates at most one job from OPT. Since all jobs in S_i can be scheduled, and every job in $\mathrm{OPT}(J)$ has profit at least 1 we have $|\mathrm{OPT}(J)| \ge |S_i|$. The maximum number of iterations is $\lceil k \ln k \rceil$. Therefore, the number of jobs eliminated from the optimal solution by all iterations is at most

$$\sum_{i=1}^{\lceil k \ln k \rceil} \frac{|S_i|}{k^3} \le \frac{\lceil k \ln k \rceil}{k^3} |\mathrm{OPT}(J)| \le \frac{1}{k} |\mathrm{OPT}(J)| \ .$$

The second statement follows from the first since every job has profit at most P. □

In the rest of this section we prove the following lemma, that implies Lemma 2.1:

Lemma 2.4 *In each block $B \in \mathcal{B}_i$ computed by any iteration i of* PartitionTimeLine*, $\mathrm{OPT}(J)$ schedules jobs with at most total profit αk^{i+2} from $S_{i-1} \setminus S_i$.*

Proof: Consider two cases. In one case, block B may have been present in $\mathcal{B}_{i-1}$ and unmodified by iteration i. This could happen only if $\mathrm{SC}(S_{i-1} \setminus S_i, B)$ could not schedule jobs with total profits more than k^{i+2} in B. In the second case, block B was created by subdividing a block from $\mathcal{B}_{i-1}$ into blocks containing jobs with profit at most k^{i+2} by SC. In both cases, all the jobs from $S_{i-1} \setminus S_i$ were available for scheduling when SC started processing block B and SC gives $1/\alpha$-approximation. □

Lemma 2.5 *$p(S^I) + \mathsf{opt}_{\mathcal{B}^{II}}(J^{II}) \ge (1 - \varepsilon)\mathsf{opt}(J)$.*

Proof: Consider the two termination conditions of PartitionTimeLine.

Termination Condition 1: PartitionTimeLine terminated after $\lceil k \ln k \rceil$ iterations, and $J^{II} = J \setminus S_i = J \setminus S_{\lceil k \ln k \rceil}$. For all iterations $1 \le i < \lceil k \ln k \rceil$, the Termination Condition 2 was not satisfied, and thus $p(S_i) < (1 - 1/k)p(S_{i-1})$. Therefore

$$p(S_{\lceil k \ln k \rceil}) \le (1 - 1/k)^{\lceil k \ln k \rceil} p(S_1) \le p(S_1)/k \le \mathsf{opt}(J)/k \ .$$

In this case, $\mathcal{B}^{II} = \mathcal{B}$. By Lemma 2.3, $\mathsf{opt}_{\mathcal{B}^{II}}(J) \ge (1 - P/k)\mathsf{opt}(J)$. Thus

$$\mathsf{opt}_{\mathcal{B}^{II}}(J^{II}) \ge \mathsf{opt}_{\mathcal{B}^{II}}(J) - p(S_{\lceil k \ln k \rceil}) \ge (1 - P/k)\mathsf{opt}(J) - \mathsf{opt}(J)/k \ge (1 - \varepsilon)\mathsf{opt}(J) \ .$$

Termination Condition 2: PartitionTimeLine terminated at iteration $i < \lceil k \ln k \rceil$, because the condition $p(S_i) \geq (1 - 1/k)p(S_{i-1})$ was satisfied. In this case, $S^I = S_i$, $J^{II} = J \setminus S_{i-1}$, $\mathcal{B}^I$ includes all the blocks containing jobs from S^I, and $\mathcal{B}^{II} = \mathcal{B}_{i-1} \setminus \mathcal{B}^I$. Thus:

$$\begin{array}{rcl} \mathsf{opt}_{\mathcal{B}^{II}}(J^{II}) & \geq & \mathsf{opt}_{\mathcal{B}_{i-1}}(J^{II}) - \mathsf{opt}_{\mathcal{B}^I}(J^{II}) = \\ & = & \mathsf{opt}_{\mathcal{B}_{i-1}}(J \setminus S_{i-1}) - \mathsf{opt}_{\mathcal{B}^I}(J^{II}) \geq \\ & \geq & \mathsf{opt}_{\mathcal{B}_{i-1}}(J) - p(S_{i-1}) - \mathsf{opt}_{\mathcal{B}^I}(J^{II}). \end{array}$$

From which we get:

$$p(S^I) + \mathsf{opt}_{\mathcal{B}^{II}}(J^{II}) \geq \mathsf{opt}_{\mathcal{B}_{i-1}}(J) - (p(S_{i-1}) - p(S_i)) - \mathsf{opt}_{\mathcal{B}^I}(J^{II}). \quad (1)$$

We bound each term in (1) separately. By Lemma 2.3, $\mathsf{opt}_{\mathcal{B}_{i-1}}(J) \geq (1 - P/k)\mathsf{opt}(J)$. Since Termination Condition 2 applied, $p(S_i) \geq (1 - 1/k)p(S_{i-1})$, and $(p(S_{i-1}) - p(S_i)) \leq p(S_{i-1})/k \leq \mathsf{opt}(J)/k$.

To bound $\mathsf{opt}_{\mathcal{B}^I}(J^{II})$, note that $\mathcal{B}^I$ is non-empty only if PartitionTimeLine was terminated due to Termination Condition 2. In that case, $J^{II} = J \setminus S_{i-1} = \bigcup_{j=1}^{r-1}(S_{j-1} \setminus S_j)$, where r is the number of iterations performed by PartitionTimeLine. Since each block in $\mathcal{B}^I$ is contained within blocks produced in each iteration, it follows from Lemma 2.4 that the profit of jobs from J^{II} scheduled by OPT(J) in each block $B \in \mathcal{B}^I$ is at most $\sum_{i=1}^{r-1} \alpha k^{i+2} \leq 2\alpha k^{r+1}$. From the operation of PartitionTimeLine, jobs with total profit at least k^{r+2} from S^I were scheduled in each block $B \in \mathcal{B}^I$. Therefore, $\mathsf{opt}_{\mathcal{B}^I}(J^{II}) \leq \frac{2\alpha}{k} p(S^I) \leq \frac{2\alpha}{k}\mathsf{opt}(J)$.

Substituting these bounds into (1) gives:

$$\begin{array}{rcl} p(S^I) + \mathsf{opt}_{\mathcal{B}^{II}}(J^{II}) & \geq & (1 - P/k)\mathsf{opt}(J) - \mathsf{opt}(J)/k - 2\alpha \cdot \mathsf{opt}(J)/k \\ & = & (1 - \varepsilon)\mathsf{opt}(J) \; . \end{array}$$

□

Lemma 2.6 *In each block $B \in \mathcal{B}^{II}$, OPT(J) schedules at most $2\alpha \cdot k^{k \ln k + 3}$ jobs from J^{II}.*

Proof: Each job in J^{II} was removed from the schedule during one of the iterations. Thus, $J^{II} = \bigcup_{i=1}^{r}(S_{i-1} \setminus S_i)$, where r is the number of iterations performed by PartitionTimeLine. Since each block in $\mathcal{B}^I$ is contained within blocks produced in each iteration, it follows from Lemma 2.4 that the total profits of jobs from J^{II} scheduled by OPT(J) in each block $B \in \mathcal{B}^{II}$ is at most $\sum_{i=1}^{r} \alpha k^{i+2} \leq 2\alpha k^{r+2} \leq 2\alpha k^{klnk+3}$ (recall that the maximum number of iterations is $\lceil k \ln k \rceil$). □

To complete the proof of Lemma 2.1 it remains to show that PartitionTimeLine runs in time $O(n \cdot Q(n, m) \cdot \ln(P/\varepsilon))$. This is so since the loop has at most $\lceil k \ln k \rceil$ iterations, and in each iteration the dominating time is $O(Q(n,m)|\mathcal{B}|)$ for executing SC at most $|\mathcal{B}|$ times, and $|\mathcal{B}| \leq n\varepsilon$.

The proof of Lemma 2.1 is complete.

2.2 Proof of Lemma 2.2

The first step for proving Lemma 2.2 is defining a linear program whose integer solutions are feasible schedules. Then, we solve the program, and use randomized rounding to obtain a feasible schedule with expected approximation ratio $1 - 1/e$. Finally, the algorithm is derandomized using the method of conditional expectations.

Let $\mathcal{S} = \mathcal{S}(J, \mathcal{I})$ be the set of all sequences $(j_1, \dots, j_t, I_1, \dots, I_t)$ of t distinct jobs in J and t (not necessarily distinct) intervals in $\mathcal{I}$, $0 \leq t \leq q$. Clearly, $|\mathcal{S}| < \sum_{i=1}^{q} n^i m^i \leq 2(nm)^q$. Given $B \in \mathcal{B}$, any (feasible) schedule S of $t \leq q$ jobs within B defines a unique sequence $(j_1, \dots, j_t, I_1, \dots, I_t) \in \mathcal{S}$, where each j_i is processed in I_i and after j_{i-1}. Given such a sequence we can find a feasible schedule within B defining it or determine that such does not exist in time $O(t) = O(q)$ time, by attempting to schedule every job within the corresponding interval in the sequence. This attempt is done by taking each job from the sequence in turn, and placing it within B as early as possible given the job's interval and avoiding overlap with jobs already scheduled during this attempt. Clearly, any two schedules within B defining the same sequences have the same profits, thus we identify any feasible schedule S within B with the sequence it defines. The profit $p_{(S,B)}$ of $(S,B) \in \mathcal{S} \times \mathcal{B}$ is the profit of some schedule that S defines within B, if such schedule exists, and $p_{(S,B)} = 0$ otherwise (that is, if no schedule within B defining S exists). Let $\mathcal{R} = \{(S,B) \in \mathcal{S} \times \mathcal{B} : p_{(S,B)} > 0\}$ and let $\mathcal{S}_B = \{S \in \mathcal{S} : p_{(S,B)} > 0\}$. For $(S,B) \in \mathcal{R}$ and $j_i \in S$ let $p_{(S,B)}(j_i) = p_{j_i, I_i}$. For every $(S,B) \in \mathcal{R}$ introduce a variable $y_{(S,B)}$ which may be interpreted as the "amount of S selected in the block B". Then integer feasible solutions to the following linear program correspond to feasible schedules within the blocks of $\mathcal{B}$.

$$\begin{array}{lll} \max & \sum_{(S,B)\in\mathcal{R}} y_{(S,B)} \cdot p_{(S,B)} & \\ \text{s.t.} & \sum_{(S,B)\in\mathcal{R}, j\in S} y_{(S,B)} \leq 1 & \forall j \in J \\ & \sum_{S\in\mathcal{S}_B} y_{(S,B)} = 1 & \forall B \in \mathcal{B} \\ & y_{(S,B)} \geq 0 & \forall (S,B) \in \mathcal{R} \end{array} \tag{2}$$

Note that this LP has $|\mathcal{R}| \leq 2|\mathcal{B}|(nm)^q$ variables and $n + |\mathcal{B}|$ constraints (that are not just non-negativity constraints). We will apply a standard randomized rounding on y to obtain an integral feasible solution $\tilde{y}$ to (2). The rounding procedure is as follows.

1. For each block B choose randomly with distribution $y_{(S,B)}$ a unique schedule S_B assigned to block B at this stage (possibly $S_B = \emptyset$).

2. For every job j assigned to more than one block, remove j from all schedules containing it, except from one that has maximum $p_{(S,B)}(j)$.

Let $\tilde{y}$ be an integral solution derived from y by such randomized rounding, and let $\tilde{\nu} = \sum_{(S,B)\in\mathcal{R}} \tilde{y}_{(S,B)} \cdot p_{(S,B)}$ be the (random variable corresponding to the) profit of the schedule specified by $\tilde{y}$, and let ν be the optimal value of (2). We will prove that the expected value of $\tilde{\nu}$ is at least $(1 - 1/e)\nu$. The proof is similar to the proof of [7, Theorem 2.1] where MAX-GAP was considered and is presented here only for completeness of exposition. We use the following statement from [7]:

Lemma 2.7 ([7], Lemma 2.1) *Let $y_1, \ldots, y_\ell$ be a sequence of non-negative reals so that $\sum_{i=1}^{\ell} y_i \leq 1$ and let $p_1 \geq p_2 \geq \cdots \geq p_\ell \geq 0$. Then*

$$p_1 y_1 + p_2(1 - y_1)y_2 + \cdots + p_\ell \left[\prod_{i=1}^{t-1}(1 - y_i)\right] y_\ell \geq (1 - (1 - 1/\ell)^\ell) \sum_{i=1}^{\ell} p_i y_i \ .$$

Lemma 2.8 *The expected value of $\tilde{\nu}$ is at least $(1 - 1/e)\nu$.*

Proof: Let $j \in J$. Sort the profits $p_{(S,B)}(j)$, $(S, B) \in \mathcal{R}$, in a decreasing order

$$p_{(S_1,B_1)}(j) \geq p_{(S_2,B_2)}(j) \geq \cdots \geq p_{(S_\ell,B_\ell)}(j) \ .$$

For simplicity, denote $p_i = p_{(S_1,B_1)}(j)$ and $y_i = y_{(S_i,B_i)}$. Let $\nu(j) = \sum_{i=1}^{\ell} p_i y_i$ be the "profit of j" in LP (2), and note that $\nu = \sum_{j\in J} \nu(j)$. Let $\tilde{\nu}(j)$ be the (random variable corresponding to the) profit from job j in the schedule computed by the algorithm. By the linearity of expectation, it would be sufficient to prove that for every $j \in J$ the expected value of $\tilde{\nu}(j)$ is at least $(1 - 1/e)\nu(j)$. This follows from Lemma 2.7, since the expected value of $\tilde{\nu}(j)$ is:

$$p_1 y_1 + p_2(1 - y_1)y_2 + \cdots + p_\ell[\prod_{i=1}^{t-1}(1 - y_i)]y_\ell \geq (1 - (1 - 1/\ell)^\ell) \sum_{i=1}^{\ell} p_i y_i \geq (1 - 1/e)\nu(j) \ .$$

□

The algorithm can be derandomized using the method of conditional probabilities. We state the algorithm for the analysis of the time complexity, but omit the proof of its validity, as it is identical to the one in [9] where MAX-GAP was considered. Given $\mathcal{B}' \subseteq \mathcal{B}$ and $J' \subseteq J$ let $\nu(J', \mathcal{B}')$ denote the optimal value of (2) with J replaced by J' and $\mathcal{B}$ by $\mathcal{B}'$. The algorithm is as follows.

```
Initialization: $J' \leftarrow J$, $\mathcal{B}' \leftarrow \mathcal{B}$.
For every $B \in \mathcal{B}$ do:
    Schedule in $B$ a set $S_B \subseteq J'$ of jobs that maximizes $p_{(S,B)} + \nu(J' \setminus S, \mathcal{B}' \setminus \{B\})$;
    $J' \leftarrow J' \setminus S_B$, $\mathcal{B} \leftarrow \mathcal{B}' \setminus \{B\}$.
EndFor
```

To complete the proof of the Lemma 2.2 it remains to show the time complexity. The time required to compute the profits $p_{(S,B)}$ is $O(q|\mathcal{B}|(nm)^q)$, $O(q)$ time per variable. We assume that this time is dominated by the time required to solve the linear program (2), which is $O\left(L(|\mathcal{B}|(nm)^q, n + |\mathcal{B}|)\right)$. This is the time complexity of the randomized version. The time complexity of the deterministic version is $O(|\mathcal{B}|(nm)^q \cdot L(|\mathcal{B}|(nm)^q, n + |\mathcal{B}|))$, as claimed.

The proof of Lemma 2.2, and thus also of Theorem 1.1, is complete.

3 Proof of Theorem 1.2

We reduce the following NP-complete problem [8] to θ-SWI.

Subset-Sum
Instance: A set $A = \{a_1, \ldots, a_n\}$ of positive integers, and another integer B.
Question: Is there $A' \subseteq A$ such that the integers in A' sum to exactly B?

The reduction is as follows. Let $1 < \alpha < \min\{2, \theta\}$ be arbitrary. Let $K = B \cdot \lceil 1/(\alpha - 1) \rceil$, and let $T = nK + B$. For each $j = 1, \ldots, n$ there are two jobs j^- and j^+ each having the same single interval $I_j = [(j-1)K, jK + B)$, and processing time $\ell_{j^-} = K$ and $\ell_{j^+} = K + a_j$, respectively. The profit of each job equals to its processing time, and set $T = nK + B$.

Since $|I_j| = K + B$ and $\ell_{j^+}, \ell_{j^-} \geq K$ for each j, the stretch factor of the obtained instance of SWI is at most $(K+B)/K = 1 + \lceil 1/(\alpha - 1) \rceil^{-1} \leq \alpha < \theta$. Thus we obtain a θ-SWI instance.

Lemma 3.1 *The answer to* Subset-Sum *is* YES *if, and only if, the* SWI *instance can achieve a total profit of T.*

Proof:
The *if* part.
Let S be a feasible set of jobs that achieves a total profit of T. We claim that then for each j *exactly* one of j^-, j^+ is in S. To see this note that for each $1 \leq j \leq n$:

(i) at most one of j^-, j^+ is in S, since $|I_j| = K + B < 2K + a_j = \ell_{j^-} + \ell_{j^+}$;

(ii) at least one of j^-, j^+ is in S, since each time window I_j contains the non-empty sub-interval $[(j-1)K + B, jK)$ that is disjoint to all the other time windows.

For $j = 1, \ldots, n$ let b_j be the boolean variable with $b_j = 1$ if $j^+ \in S$ and $b_j = 0$ otherwise (that is, if $j^- \in S$). The total profit of the jobs in S is exactly $\sum_{j=1}^{n}(K + b_j a_j) = nK + \sum_{j=1}^{n} b_j a_j$. Comparing this to the total profit $T = nK + B$, we get $\sum_{j=1}^{n} b_j a_j = B$. This defines a solution to the Subset-Sum instance.

The *only if* part.
Let $A' \subseteq A$ such that A' sums to B. We will show a solution to the SWI instance that achieves a profit of T. Let $b_j = 1$ if $a_j \in A'$ and $b_j = 0$ otherwise. Create the following solution S to SWI. For each $j = 1, \ldots, n$ do: if $b_j = 1$ add the job j^+ to S, with the start time $(j-1)K + \sum_{i<j} b_i a_i$; otherwise, add the job j^- to S, with the same start time. It is easy to verify that this is a feasible solution to SWI with a total profit of T. □

The following dynamic-programming algorithm computes an optimal solution for 2-SWI. Define the *latest start* of job j as $t_j = d_j - \ell_j$. The jobs are sorted in order of non-decreasing t_j, breaking ties arbitrarily, so assume $t_1 \leq t_2 \leq \cdots \leq t_n$. For $1 \leq i \leq n$ and $0 \leq p \leq \sum_j p_j = \mathcal{P}$ define $D[i,p]$ as the minimal ending time of a feasible schedule $S \subseteq \{1, \ldots, i\}$ with total profit exactly p; $D[i,p] = \infty$ if no such feasible S exists. The optimal profit eventually equals to $\max\{p : D[n,p] < \infty\}$.

The table D can be computed using the following recurrence. Set $D[1,p] = r_1 + \ell_1$ if $p = p_1$ and $r_1 + \ell_1 \leq d_1$, and $D[1,p] = \infty$ otherwise. Let $2 \leq i \leq n$. If $p_{i+1} > p$ set $D[i,p] = D[i-1,p]$. For $p_{i+1} \leq p$ set

$$s_i = \max\{D[i-1, p - p_i], r_i\}.$$

If $s_i + \ell_i > d_i$ set $D[i,p] = D[i-1,p]$. Otherwise,

$$D[i,p] = \min\{s_i + \ell_i, D[i-1,p]\}.$$

Lemma 3.2 *The table D is computed correctly in $O(\min\{n\mathcal{P}, rn^r \log n\})$ time, where $P = \sum_j p_j$.*

Proof: Note that if $s_j < s_i$ are feasible start times of jobs i, j, then $t_j < t_i$. Indeed, for θ-SWI, $t_j < s_j + (\theta - 1)\ell_j \leq s_i + (\theta - 1)\ell_j \leq t_i + (\theta - 1)\ell_j$. In particular, for $\theta = 2$ we have $t_j < t_i$. Clearly, every entry is computed using previous entries. Any feasible solution defines a sequence of jobs according to their starting times. By the above, any such sequence must be a subsequence of the latest start sequence. Therefore, the step of computing $D[i,p]$ from previous entries requires examining just two options: whether or not to append job i to the constructed schedule. The decision is made by choosing the earliest-finishing one from these two options. The initial sorting of jobs requires $O(n \log n)$ time. The number of entries in the table D is $O(n\mathcal{P})$, and each entry is computed in constant time. The time bound $O(n \log n + n\mathcal{P}) = O(n\mathcal{P})$ follows.

For the $O(rn^r \log n)$ time bound, we compute all possible profits $\Pi = \{p(S) : S \subseteq J\}$ of subsets of the items (assuming any set of items can be chosen) and sort these profits in an increasing order. This can be done in $O(rn^r \log n)$ time, as follows. Let $p_1, \ldots p_r$ be the possible distinct item profits, and let n_i be the number of items in J of profit p_i. Then $|\Pi| \leq (n_1+1)(n_2+1)\cdots(n_r+1) \leq (n/(r+1))^r = O(n^r)$, and Π can be computed in $O(rn^r)$ time. Thus the total time required including sorting is $O(rn^r + n^r \log(n^r)) = O(rn^r \log n)$. Then we use the same dynamic programming algorithm, but define and fill in the table only the relevant entries. The number of entries in the table is $O(n^r)$, and each entry is computed in constant time. $\square$

The proof of Theorem 1.2 is complete.

Remark: A standard profit-truncation algorithm may be utilized to give a FPTAS for 2-SWI with running time $O(n^2/\varepsilon)$, which is the same as the one of [4]. This FPTAS is based on algorithm pseudo-polynomial in $\mathcal{P}$, while the one of [4] is based on an algorithm pseudopolynomial in $T = \max_{j \in J} d_j$. It remains open whether θ-SWI is APX-hard for any $\theta > 2$.

4 Conclusions and open problems

In this paper we gave a $(1 - 1/e - \varepsilon)$-approximation scheme for GSWI with bounded profits, while showing that SWI with small stretch factor are NP-hard. One open problem is obtaining a ratio better than $1/2$ for arbitrary profits. Another open problem is whether SWI with $\{0,1\}$-profits (or JISP) admits a better ratio than $(1 - 1/e)$.

Acknowledgment: We thank anonymous referees for useful comments.

References

[1] A. Bar-Noy, R. Bar-Yehuda, A. Freund, J. Naor, and B. Shieber. A unified approach to approximating resource allocation and scheduling. *J. of the ACM*, 48(5):1069–1090, 2001.

[2] A. Bar-Noy, S. Guha, S. Naor, and B. Schieber. Approximating the throughput of multiple machines in real-time scheduling. *SIAM J.Comput.*, 31(2):331–352, 2001.

[3] I. Beniaminy, Z. Nutov, and M. Ovadia. Approximating interval scheduling problems with bounded profits. In Proc. *European Symposium on Algorithms (ESA)*, pages 487–497, 2007.

[4] P. Berman and B. DasGupta. Multi-phase algorithms for throughput maximization for real-time scheduling. *J. Comb. Optim.*, 4(3):307–323, 2000.

[5] J. Chuzhoy, R. Ostrovski, and Y. Rabani. Approximation algorithms for the job interval selection problem and related scheduling problems. In Proc. *Symposium on Foundations of Computer Science (FOCS)*, pages 348–356, 2001.

[6] U. Feige and J. Vondrák. Approximation algorithms for allocation problems: Improving the factor of $1-1/e$. In Proc. *Symposium on Foundations of Computer Science (FOCS)*, pages 667–676, 2006.

[7] L. Fleischer, M. X. Goemans, V. S. Mirrokni, and M. Sviridenko. Tight approximation algorithms for maximum general assignment problems. In Proc. *Symposium on Discrete Algorithms (SODA)*, pages 611–620, 2006.

[8] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman, San-Francisco, 1979.

[9] Z. Nutov, I. Beniaminy, and R. Yuster. A $(1-1/e)$-approximation algorithm for the generalized assignment problem. *Oper. Res. Lett.*, 34(3):283–288, 2006.