Quick Minimization of Tardy Processing Time on a Single Machine

Baruch Schieber[1] and Pranav Sitaraman[2]

[1] New Jersey Institute of Technology, Newark, NJ 07102, USA,
sbar@njit.edu
[2] Edison Academy Magnet School, Edison, NJ 08837, USA,
sitaraman.pranav@gmail.com

Abstract. We consider the problem of minimizing the total processing time of tardy jobs on a single machine. This is a classical scheduling problem, first considered by [Lawler and Moore 1969], that also generalizes the Subset Sum problem. Recently, it was shown that this problem can be solved efficiently by computing $(\max, \min)$-skewed-convolutions. The running time of the resulting algorithm is equivalent, up to logarithmic factors, to the time it takes to compute a $(\max, \min)$-skewed-convolution of two vectors of integers whose sum is $\mathrm{O}(P)$, where P is the sum of the jobs' processing times. We further improve the running time of the minimum tardy processing time computation by introducing a job "bundling" technique and achieve a $\tilde{\mathrm{O}}\left(P^{2-1/\alpha}\right)$ running time, where $\tilde{\mathrm{O}}(P^{\alpha})$ is the running time of a $(\max, \min)$-skewed-convolution of vectors of size P. This results in a $\tilde{\mathrm{O}}\left(P^{7/5}\right)$ time algorithm for tardy processing time minimization, an improvement over the previously known $\tilde{\mathrm{O}}\left(P^{5/3}\right)$ time algorithm.

1 Introduction

The input to the Minimum Tardy Processing Time (MTPT) Problem consists of n jobs each of which is associated with a due date and processing time $p_i \in \mathbb{N}$. Consider a (nonpreemptive) schedule of these jobs on a single machine that can execute only one job at a time. A job is *tardy* if it terminates after its due date. The MTPT Problem is to find a schedule of the jobs that minimizes the total processing time of the tardy jobs. In the standard scheduling notation the MTPT problem is denoted $1||\sum p_j U_j$.

Consider an instance of MTPT in which all the jobs have the same due date d. Let $P = \sum_{j=1}^{n} p_j > d$. The decision whether the total processing time of the tardy jobs is exactly $P - d$ (which is optimal in this case) is equivalent to finding whether there exists a subset of the jobs whose processing time sums to d. This is equivalent to the Subset Sum problem. It follows that MTPT is NP-hard. MTPT is weakly NP-hard and Lawler and Moore [6] gave an $\mathrm{O}(P \cdot n)$ time algorithm for this problem.

Bringmann *et al.* [2] introduced a new convolution variant called a $(\max, \min)$-skewed-convolution. They gave an algorithm for MTPT that uses $(\max, \min)$-skewed-convolutions, and proved that up to logarithmic factors, the running time of this algorithm is equivalent to the time it takes to compute a $(\max, \min)$-skewed-convolution of integers that sum to $\mathrm{O}(P)$. They also gave an $\tilde{\mathrm{O}}\big(P^{7/4}\big)$ time algorithm[3] for computing a $(\max, \min)$-skewed-convolution of integers that sum to $\mathrm{O}(P)$, which results in an $\tilde{\mathrm{O}}\big(P^{7/4}\big)$ time algorithm for the MTPT problem. Klein *et al.* [3] further improved the algorithm for computing a $(\max, \min)$-skewed-convolution and achieved an $\tilde{\mathrm{O}}\big(P^{5/3}\big)$ running time, and thus an $\tilde{\mathrm{O}}\big(P^{5/3}\big)$ time algorithm for the MTPT problem.

A natural approach to further improve the MTPT algorithm is by improving the running time of a $(\max, \min)$-skewed computation. However, obtaining an $\tilde{o}\big(P^{3/2}\big)$ time algorithm for computing a $(\max, \min)$-skewed-convolution seems difficult as this would imply an improvement to the best known (and decades old) algorithm for computing a $(\max, \min)$-convolution [5]. We were able to "break" the $\tilde{\mathrm{O}}\big(P^{3/2}\big)$ barrier by introducing a job "bundling" technique. Applying this technique in conjunction with the best known algorithm for computing a $(\max, \min)$-skewed-convolution yields an $\tilde{\mathrm{O}}\big(P^{7/5}\big)$ time algorithm for the MTPT problem. This algorithm outperforms Lawler and Moore's algorithm [6] in instances where $n = \tilde{\omega}\big(P^{2/5}\big)$. In general, applying our technique in conjunction with an $\tilde{\mathrm{O}}(P^{\alpha})$ time algorithm for computing a $(\max, \min)$-skewed-convolution yields an $\tilde{\mathrm{O}}\big(P^{2-1/\alpha}\big)$ time for the MTPT problem.

The rest of the paper is organized as follows. In Section 2 we introduce our notations and describe the prior work that we apply in our algorithm. Section 3 specifies our algorithm, and Section 4 has some concluding remarks.

2 Preliminaries

Our notations follow the notations in [2]. The input to the MTPT problem is a set of n jobs $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$. Each job $J_j \in \mathcal{J}$ has due date $e_j \in \mathbb{N}$ and processing time $p_j \in \mathbb{N}$. Let $D_{\#}$ denote the number of distinct due dates, and denote the monotone sequence of distinct due dates by $d_1 < d_2 < d_3 < \cdots < d_{D_{\#}}$, with $d_0 = 0$. Let $\mathcal{J}_k \subseteq \mathcal{J}$ be the set of jobs with due date d_k. Let $D = \sum_{i=1}^{D_{\#}} d_i$ and $P = \sum_{i=1}^{n} p_i$. For any $I = \{i, \ldots, j\}$, where $1 \le i \le j \le D_{\#}$, let $\mathcal{J}_I = \bigcup_{i \in I} \mathcal{J}_i$ and $P_I = \sum_{J_i \in \mathcal{J}_I} p_i$.

Recall that the goal is to schedule the jobs in $\mathcal{J}$ so that the total processing time of tardy jobs is minimized. Since we only consider non-preemptive schedules, any schedule S corresponds to a permutation $\sigma_S : \{1, \ldots, n\} \to \{1, \ldots, n\}$ of the job indices. The completion time of job $J_j \in \mathcal{J}$ in schedule S is $C_j = \sum_{\sigma_S(i) \le \sigma_S(j)} p_i$, and j is tardy in S if $C_j > e_j$. Therefore, we can consider that our algorithm seeks to minimize $\sum_{J_j \in \mathcal{J}, C_j > e_j} p_j$.

Next, we recall the definition of convolutions and describe the techniques developed in [2] and used by our algorithm. Given two vectors A and B of dimension $n+1$ and two binary operations $\circ$ and $\bullet$, the $(\circ, \bullet)$-convolution applied on A and B results in a $2n+1$ dimensional vector C, defined as:

$$C[k] = \bigcirc_{i=\max\{0,k-n\}}^{\min\{k,n\}} A[i] \bullet B[k-i], \ \forall\ k \in \{0, \ldots, 2n\}\,.$$

[3] The notation $\tilde{\mathrm{O}}(\cdot)$ hides all logarithmic factors.

A $(\max, \min)$-skewed-convolution applied on A and B results in a $2n+1$ dimensional vector C, defined as:

$$C[k] = \max_{i=\max\{0,k-n\}}^{\min\{k,n\}} \min\{A[i], B[k-i]+k\}, \ \forall\, k \in \{0,\ldots,2n\}.$$

Bringmann *et al.* [2] apply an equivalent form of $(\max, \min)$-skewed-convolution defined as

$$C[k] = \max_{i=\max\{0,k-n\}}^{\min\{k,n\}} \min\{A[i], B[k-i]-i\}, \ \forall\, k \in \{0,\ldots,2n\}.$$

Below, we use this equivalent form as well.

Let X and Y be two integral vectors. Define the *sumset* $X \oplus Y = \{x+y : x \in X, y \in Y\}$. It is not difficult to see that the sumset can be inferred from a $(+,\cdot)$-convolution of X_1 and X_2 which can be computed in $\tilde{O}(P)$ time for $X, Y \subseteq \{0,\ldots,P\}$ as in [1].

The set of all subset sums of entries of X, denoted $\mathcal{S}(X)$, is defined as $\mathcal{S}(X) = \{\sum_{x\in Z} x : Z \subseteq X\}$. These subset sums can be calculated in $\tilde{O}\left(\sum_{x\in X} x\right)$ time by successive computations of sumsets [4]. We note that we always have $0 \in \mathcal{S}(X)$. Define the $t-$*prefix* and $t-$*suffix* of $\mathcal{S}(X)$ as $\mathsf{pref}(\mathcal{S}, t) = \{x \in \mathcal{S}(X) \wedge x \le t\}$ and $\mathsf{suff}(\mathcal{S}, t) = \{x \in \mathcal{S}(X) \wedge x > t\}$.

We say that a subset of jobs $\mathcal{J}' \subseteq \mathcal{J}$ can be scheduled *feasibly* starting at time t if there exists a schedule of these jobs starting at time t such that all jobs are executed by their due date. Note that it is enough to check whether all jobs in $\mathcal{J}'$ are executed by their due date in the *earliest due date first* (EDD) schedule of these jobs starting at t.

For a consecutive subset of indices $I = \{i_0, i_0+1, \ldots, i_1\}$, with $1 \le i_0 \le i_1 \le D_\#$, define an integral vector $M(I)$ as follows. The entry $M(I)[x]$ equals $-\infty$ if none of the subsets of jobs in $\mathcal{J}_I$ with total processing time exactly x can be scheduled feasibly. Otherwise, $M(I)[x]$ equals the latest time t starting at which a subset of jobs in $\mathcal{J}_I$ with total processing time x can be scheduled feasibly. Applying the algorithm for $(\max, \min)$-skewed-convolutions given in [3], we get an $\tilde{O}\left({P_I}^{5/3}\right)$ time algorithm for computing $M(I)$, where $P_I = \sum_{J_i \in \mathcal{J}_I} p_i$. This implies an $\tilde{O}\left(P^{5/3}\right)$ time algorithm for the MTPT problem.

In addition to the algorithm that uses $(\max, \min)$-skewed-convolutions, Bringmann *et al.* [2] gave a second algorithm for the MTPT problem. The running time of this algorithm is $\tilde{O}(P \cdot D_\#)$. We use a version of this algorithm in our algorithm and for completeness we describe it in Algorithm 1.

3 The Algorithm

We define job bundles by coloring due dates in *red* and *blue*. The blue due dates are the bundled ones.

Choose some $\delta \in (0,1)$. For each $k = 1, 2, \ldots D_\#$, color the due date d_k red if $\sum_{J_i \in \mathcal{J}_k} p_i > P^{1-\delta}$. To determine the bundles we repeat the following procedure until all due dates are colored.

Algorithm 1 The $\tilde{\mathrm{O}}(P \cdot D_{\#})$ time algorithm

1: Let $d_1 < \ldots < d_{D_{\#}}$ denote the different due dates of jobs in $\mathcal{J}$.
2: **for** $i = 1, \ldots, D_{\#}$ **do**
3: Compute $X_i = \{p_j : J_j \in \mathcal{J}_i\}$
4: Compute $\mathcal{S}(X_i)$
5: Let $S_0 = \emptyset$.
6: **for** $i = 1, \ldots, D_{\#}$ **do** ▷ *compute the sumsets and exclude infeasible sums*
7: Compute $S_i = S_{i-1} \oplus \mathcal{S}(X_i)$.
8: Remove any $x \in S_i$ with $x > d_i$.
9: Return $P - x$, where x is the maximum value in $S_{D_{\#}}$.

Let m be the largest index for which due date d_m is not yet colored. Find the smallest $k < m$ that satisfies the following conditions.
Condition 1: None of the due dates $d_k, \ldots, d_m$ are colored red.
Condition 2: $\sum_{i=k}^{m} \sum_{J_j \in \mathcal{J}_i} p_j \leq P^{1-\delta}$ Color all due dates $d_k, d_{k+1}, \ldots, d_m$ blue and "bundle" them into one group, denoted $B(k, m)$. We say that due date d_k is the *start* of the bundle and d_m is the *end* of the bundle.

Lemma 1. *The number of red due dates is $O(P^{\delta})$ and the number of bundles is $O(P^{\delta})$.*

Proof. Clearly, there can be at most P^{δ} due dates with $\sum_{J_i \in \mathcal{J}_k} p_i > P^{1-\delta}$. Consider a bundle $B(k, m)$. Since $k < m$ is the smallest index that satisfies the two conditions above, it is either true that d_{k-1} is red or $\sum_{i=k-1}^{m} \sum_{J_j \in \mathcal{J}_i} p_j > P^{1-\delta}$.

(i) Since there are at most P^{δ} red due dates, there can be at most only P^{δ} bundles $B(k, m)$ for which d_{k-1} is red.
(ii) Consider the sum $\sum_{i=k-1}^{m} \sum_{J_j \in \mathcal{J}_i} p_j$, for a bundle $B(k, m)$. Note that p_j of a job $J_j \in \mathcal{J}_{k-1} \cup \mathcal{J}_m$ may appear in at most one more sum that corresponds to a different bundle, while p_j of a job $J_j \in \bigcup_{i=k}^{m-1} \mathcal{J}_i$ cannot appear in any other such sum. Thus, the total of all sums cannot exceed $2P$. It follows that there are at most $2P^{\delta}$ bundles $B(k, m)$ for which $\sum_{i=k-1}^{m} \sum_{J_j \in \mathcal{J}_i} p_j > P^{1-\delta}$.

■

Algorithm 2 called Solve($\mathcal{J}$), given below, follows the structure of Algorithm 1 with additional processing of entire bundles that avoids processing each due date in the bundles individually. We prove later that processing a bundle takes $\tilde{\mathrm{O}}\big(P^{(1-\delta)\cdot\alpha} + P\big)$ time, where $\tilde{\mathrm{O}}(P^{\alpha})$ is the running time of the algorithm needed for computing a (max, min)-skewed-convolution. Processing each red due date takes $\tilde{\mathrm{O}}(P)$ time. Substituting $\delta = 1 - \frac{1}{\alpha}$ yields a total running time of $\tilde{\mathrm{O}}\big(P \cdot P^{1-1/\alpha}\big) = \tilde{\mathrm{O}}\big(P^{2-1/\alpha}\big)$.

Theorem 1. *Algorithm Solve($\mathcal{J}$) returns the longest feasible schedule that starts at d_0.*

Algorithm 2 $\textsc{Solve}(\mathcal{J})$

1: Let $T = \{0\}$
2: For each red due date d_i, compute $X_i = \{p_j : e_j = d_i\}$ and $\mathcal{S}(X_i)$
3: **for** $i = 1, \ldots, D_\#$ **do**
4:     **if** d_i is a red due date **then**
5:         Compute $T = T \oplus \mathcal{S}(X_i)$
6:         Remove any $x \in T$ with $x > d_i$
7:     **else if** d_i is the end of some bundle $B(k, i)$ **then**
8:         Let $I = \{k, \ldots, i\}$
9:         Compute the vector $M(I)$.
10:         Let $S_i = \{x \in \{0, \ldots, P_I\} : M(I)[x] \neq -\infty\}$.
11:         **if** $d_k - P_I \geq 0$ **then**
12:             Let $T = T \cup (\mathsf{pref}(T, d_k - P_I) \oplus S_i)$.
13:         Let M' be an integral vector of dimension d_k and initialize $M' = -\infty$.
14:         For each $x \in \mathsf{suff}(T, d_k - P_I)$, let $M'[x] = 0$
15:         **for** $y = 0, \ldots, d_k - 1 + P_I$ **do**
16:             Let $C[y] = \max_{x=0}^{y} \min\{M'[x], M(I)[y-x] - x\}$
17:         Let $T_i = \{y \in \{0, \ldots, d_k - 1 + P_I\} : C[y] = 0\}$
18:         Let $T = T \cup T_i$
19:         Remove any $x \in T$ with $x > d_i$
20: Return $P - x$, where x is the maximum value in T

Proof. Consider iteration i of Algorithm $\textsc{Solve}(\mathcal{J})$, for an index i such that either d_i is a red due date or d_i is the end of some bundle $B(k, i)$. To prove the theorem it suffices to prove that at the end of any such iteration i the set T consists of the processing times of all feasible schedules of jobs in $\bigcup_{j=1}^{i} \mathcal{J}_j$ that start at d_0. The proof is by induction. The basis is trivial since T is initialized to $\{0\}$. Consider such an iteration i and suppose that the claim holds for all iterations $i' < i$ such that either $d_{i'}$ is a red due date or $d_{i'}$ is the end of some bundle $B(k', i')$. We distinguish two cases.

Case 1: d_i is a red due date. In this case it must be that either d_{i-1} is also a red due date or d_{i-1} is the end of some bundle $B(k', i-1)$. Thus, by our induction hypothesis, at the start of iteration i the set T consists of the processing times of all feasible schedules of subsets of jobs in $\bigcup_{j=1}^{i-1} \mathcal{J}_j$ that start at d_0. Since iteration i sets $T = T \oplus S(X_i)$ (Line 5), the claim follows.

Case 2: d_i is the end of some bundle $B(k, i)$. By our induction hypothesis, at the start of iteration i the set T consists of the processing times of all feasible schedules of subsets of jobs in $\bigcup_{j=1}^{k-1} \mathcal{J}_j$ that start at d_0. Let $I = \{k, \ldots, i\}$. The maximum length of any feasible schedule of subsets of jobs in $\mathcal{J}_I$ is P_I. Since the earliest due date of these jobs is d_k we are guaranteed that any such feasible schedule can start at any time up to (and including) $d_k - P_I$ (assuming that $d_k - P_I \geq 0$). By the definition of $M(I)$ the set $S_i = \{x \in \{0, \ldots, P_I\} : M(I)[x] \neq -\infty\}$ consists of the processing times of all feasible schedules of subsets of jobs in $\mathcal{J}_I$ (Line 10). $\mathsf{pref}(T, d_k - P_I)$ consists of the processing times of all feasible schedules of subsets of jobs in $\bigcup_{j=1}^{k-1} \mathcal{J}_j$ that start at d_0 and end at any time up to (and including)

$d_k - P_I$. Since iteration i sets $T = T \cup (\mathsf{pref}(T, d_k - P_I) \oplus S_i)$ (Line 12), after this line T consists of all the feasible schedules of subsets of jobs in $\bigcup_{j=1}^{i} \mathcal{J}_j$ that start at d_0 and also satisfy the condition that the sum of the lengths of the jobs in $\bigcup_{j=1}^{k-1} \mathcal{J}_j$ that are scheduled is at most $d_k - P_I$.

The set T is still missing the lengths of all the feasible schedules of subsets of jobs in $\bigcup_{j=1}^{i} \mathcal{J}_j$ that start at d_0 in which the sum of the lengths of the jobs in $\bigcup_{j=1}^{k-1} \mathcal{J}_j$ exceeds $d_k - P_I$. These schedules are added to T in Lines 13–18 of Solve($\mathcal{J}$). Consider such a feasible schedule of length y in which the length of the jobs in $\bigcup_{j=1}^{k-1} \mathcal{J}_j$ is some $x > d_k - P_I$, which implies that $M'[x] = 0$ (Line 14). To complement the prefix of this schedule by a feasible schedule of a subset of jobs in J_I that starts at x and is of length $y - x$ we must have $M(I)[y - x] \geq x$ or $\min\{M'[x], M(I)[y - x] - x\} = 0$. Lines 15–16 of Solve($\mathcal{J}$) check if such a feasible schedule exists. ∎

Lemma 2. *The running time of algorithm Solve($\mathcal{J}$) is $\tilde{O}\big(P^{(1-\delta)\cdot\alpha} + P\big) \cdot P^{\delta}$.*

Proof. By Lemma 1 the number of iterations that are not vacuous is P^{δ}. It is not difficult to see that all operations other than the computation of the vectors $M(I)$, C, and T_i take $\tilde{O}(P)$ time. (Note that the initialization of the first $d_k - P_I$ entries of vector M' can be done "implicitly".) The vector $M(I)$ is computed as in [2] in $\tilde{O}({P_I}^{\alpha})$ time. The vector C is also computed via a $(\max, \min)$-skewed-convolution and thus its computation time is proportional to the sum of lengths of the vectors $M(I)$ and M' (up to logarithmic factors). Naively, this sum of lengths is $d_k + P_I$. However, since $M'[x] = -\infty$ for all $x \leq d_k - P_I$, we can ignore these entries and implement the convolution in $\tilde{O}({P_I}^{\alpha})$ time. Since $M'[x] = -\infty$, for all $x \leq d_k - P_I$, we have also $C[x] = -\infty$, and thus T_i can be computed in $O(P_I)$ time (Line 17). Recall that by the definition of bundles $P_I \leq P^{1-\delta}$. Thus, the lemma is proved. ∎

4 Conclusions

We have shown a $\tilde{O}\big(P^{7/5}\big)$ time algorithm for tardy processing time minimization, an improvement over the previously known $\tilde{O}\big(P^{5/3}\big)$ time algorithm. Improving this bound further is an interesting open problem. In general, by applying our job "bundling" technique we can achieve a $\tilde{O}\big(P^{2-1/\alpha}\big)$ running time, where $\tilde{O}(P^{\alpha})$ is the running time of a $(\max, \min)$-skewed-convolution of vectors of size P. Since it is reasonable to assume that computing a $(\max, \min)$-skewed-convolution requires $\tilde{\Omega}\big(P^{3/2}\big)$ time, our technique is unlikely to yield a $\tilde{o}\big(P^{4/3}\big)$ running time. It will be interesting to see whether this running time barrier can be broken, and whether the MTPT problem can be solved without computing a $(\max, \min)$-skewed-convolution.

References

1. Alfred V. Aho, John E. Hopcroft, and Jeffrey D. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley, 1974.
2. Karl Bringmann, Nick Fischer, Danny Hermelin, Dvir Shabtay, and Philip Wellnitz. Faster minimization of tardy processing time on a single machine. *Algorithmica*, 84:1341–1356, 2022.
3. Kim-Manuel Klein, Adam Polak, and Lars Rohwedder. On minimizing tardy processing time, max-min skewed convolution, and triangular structured ILPs. In *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2947–2960, USA, 2023. Society for Industrial and Applied Mathematics.
4. Konstantinos Koiliaris and Chao Xu. A faster pseudopolynomial time algorithm for subset sum. In *Proceedings of the 2017 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1062–1072, USA, 2017. Society for Industrial and Applied Mathematics.
5. S. R. Kosaraju. Efficient tree pattern matching. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science*, pages 178–183, USA, 1989. IEEE Computer Society.
6. Eugene L. Lawler and J. M. Moore. A functional equation and its application to resource allocation and sequencing problems. *Management Science*, 16:77–84, 1969.