A faster FPTAS for makespan minimization with time-dependent agreeable V-shaped processing times

Nir Halman[1] and Helmut A. Sedding[2]

[1] Bar-Ilan University Ramat-Gan, Israel
nir.halman@biu.ac.il
[2] ZHAW, Institute of Data Analysis and Process Design, Switzerland
helmut.sedding@zhaw.ch

Keywords: time-dependent scheduling, FPTAS, K-approximation sets and functions, monotone dynamic programming.

1 Introduction

Scheduling jobs with time-dependent V-shaped processing times on a single machine minimizing makespan $C_{\max}$ is an NP-hard scheduling problem even if all jobs have the same slopes, but it permits a fully polynomial time approximation scheme (FPTAS), even for job-dependent slopes if they are agreeable (Sedding 2020*a*). We improve the FPTAS runtime by factor $1/\varepsilon$ (up to log terms) by giving an alternative dynamic programming (DP) formulation and employing the recent advances in Alon & Halman (2021) that let us apply the technique of K-approximation sets and functions.

Job deterioration is a common theme in time-dependent scheduling (Gawiejnowicz 2020, Sedding 2020*a*): The processing time is linearly dependent on the job's start time t, defined by function $p_j(t) = \ell_j + b_j t$ for basic processing time $\ell_j \geq 0$ and slope $b_j \geq 0$. Minimization of $C_{\max}$ requires to order the jobs by ℓ_j/b_j nondecreasingly (jobs with $b_j = 0$ last).

We study the extension to shortening as well as deteriorating processing times in a V-shape (Sedding 2020*a*), which model walking time for assembly operations, from a moving assembly line to a statically positioned supply at the line side. Constant times can only upper bound that, and are 51% higher in an extended case in Sedding (2023).

An instance specifies rational-valued $\ell_j \geq 0$, $b_j \geq 0$, a shortening slope $0 \leq a_j \leq 1$, and a common ideal start time τ. Then, a job's processing time is defined by

$$p_j(t) = \ell_j + \max\{-a_j\,(t-\tau),\, b_j\,(t-\tau)\}. \tag{1}$$

A solution is then structured into three parts (in that order): a first sequence S_1 that completes strictly before τ, a straddler job χ that starts no later than at τ and completes not earlier than at τ, and a second sequence S_2. Interestingly, the straddler job is not necessarily the shortest job. Within sequence S_2, jobs are sorted nondecreasingly by ℓ_j/b_j; while in S_1, they are sorted nonincreasingly by ℓ_j/a_j. Hence, solving an instance involves to select a straddler job, and a partition of the other jobs into the two sorted sequences. This problem is NP-hard already for common slopes ($a_j = a$ and $b_j = b$), which is shown in Sedding (2020*a*), as well as for the all-zero a_j case (Kononov 1997, Kubiak & van de Velde 1998), and the for all-zero b_j case (Cheng, Ding, Kovalyov, Bachman & Janiak 2003).

Let us only consider the special case where the basic processing times and slopes have *agreeable ratios*: $\ell_i/a_i \leq \ell_j/a_j \iff \ell_i/b_i \leq \ell_j/b_j$ for any pair of two jobs i, j. This case still permits an FPTAS, which is shown in Sedding (2020*a*). We give a faster FPTAS based on the technique of K-approximation sets and functions, introduced in Halman, Klabjan, Mostagir, Orlin & Simchi-Levi (2009), by reformulating the problem as a certain *monotone dynamic program* (DP) that falls into the FPTAS framework of Alon & Halman (2021).

Employing it allows us to omit an explicit algorithm statement, and directly conclude running time and approximation error. The resulting FPTAS's runtime dependency on n is in $\mathcal{O}(n^6)$, and is linear in $1/\varepsilon$ up to log terms, i.e., faster by a factor of $1/\varepsilon$ up to log terms than the tailor-made FPTAS in Sedding (2020*a*). As the considered problem includes special cases like all-zero a_j slopes, or all-zero b_j slopes, specialized FPTASes (Cai, Cai & Zhu 1998, Halman 2020, Kovalyov & Kubiak 1998, Kovalyov & Kubiak 2012, Ji & Cheng 2007, Sedding 2020*b*) can be substituted with our approach.

2 Dynamic Program

Based on the DP in Sedding (2020*a*), we introduce an alternative formulation as a certain monotone DP. First of all, reindex the jobs such that the straddler job χ is job $n+1$. Let state x of the n-stages dynamic program denote the (exact) completion time of the sequence of jobs that are executed before time τ, assuming that it can start being processed as early as time 0 (to be called first sequence S_1^j, containing jobs from set $\{1,\dots,j\}$). Because the straddler job starts by τ, the range of the state space for x is the rational valued interval $[0,\tau]$. For every stage $j=1,\dots,n$, we define two functions of the state x.

The first function, denoted by $z_j(x)$, refers to a second sequence $S_2^j(x)$ scheduled to start exactly at τ that minimizes the objective for all jobs $1,\dots,j$, and contains only the jobs $\{1,\dots,j\}\setminus S_1^j$. The value of $z_j(x)$ is explicitly not the objective value, but rather the makespan of $S_2^j(x)$ from τ to the completion time of the last job in the sequence. This makespan can be derived from Sedding (2020*a*, Eq. (6)) and, given state x and defining $F(i,S_2^j(x))\subset S_2^j(x)$ as the set of jobs that follow job i in $S_2^j(x)$, is equivalent to

$$z_j(x)=\textstyle\sum_{i\in S_2^j(x)}\Big(\ell_i\cdot\prod_{f\in F(i,S_2^j(x))}(1+b_f)\Big). \tag{2}$$

After stage n, the straddler job χ is appended to the end of the first sequence S_1^n, then the second sequence S_2^n follows, and the resulting completion time is the objective value. To calculate it, we use a second function, denoted by $y_j(x)$, that describes the proportional increase of sequence $S_2^j(x)$'s makespan $z_j(x)$ if increasing its start time, when having the jobs in $S_1^j(x)$ and $S_2^j(x)$ as determined by the state variable x (see below).

Algorithm 1 (DP). The alternative dynamic programming algorithm's steps are as follows.

1. Initialize functions $z_0(\cdot)\equiv 0$ and $y_0(\cdot)\equiv 1$ over their entire domain $[0,\tau]$.
2. For all j from 1 to n:
   (a) To append job j to the end of S_1, define

$$z'(x)=z_{j-1}\Big(\tfrac{x-\ell_j-a_j\tau}{1-a_j}\Big),\qquad \ell_j+a_j\tau\le x\le\tau. \tag{3a}$$

   To prepend job j to the beginning of S_2, define

$$z''(x)=\ell_j\cdot y_{j-1}(x)+z_{j-1}(x),\qquad 0\le x\le\tau. \tag{3b}$$

   (b) For S_2's makespan, define

$$z_j(x)=\begin{cases}z''(x), & \text{if } 0\le x<\ell_j+a_j\tau,\\ \min\{z'(x),z''(x)\}, & \text{if } \ell_j+a_j\tau\le x\le\tau,\end{cases}\qquad 0\le x\le\tau. \tag{3c}$$

   (c) For S_2's start-time-dependent makespan increase, define, for $0\le x\le\tau$,

$$y_j(x)=\begin{cases}(1+b_j)\cdot y_{j-1}(x), & \text{if } 0\le x<\ell_j+a_j\tau,\\ (1+b_j)\cdot y_{j-1}(x), & \text{if } \ell_j+a_j\tau\le x\le\tau \text{ and } z_j(x)=z''(x),\\ y_{j-1}\Big(\tfrac{x-\ell_j-a_j\tau}{1-a_j}\Big), & \text{if } \ell_j+a_j\tau\le x\le\tau \text{ and } z_j(x)=z'(x).\end{cases} \tag{3d}$$

3. Return completion time

$$C_{\max}^{\chi} = \inf_{0 \le x \le \tau} \{\tau + y_n(x) \cdot \max\{x + p_{\chi}(x) - \tau,\, 0\} + z_n(x)\}. \tag{3e}$$

Let us explain the DP recursion. Regarding function $z'(\cdot)$ in Step 2(a), to attain that job j finishes at time x, we need job $j-1$ to finish at time $\frac{x-\ell_j-a_j\tau}{1-a_j}$. If $\ell_j + a_j\tau > \tau$, then the domain of the function is empty, i.e., job j cannot be processed as a job of the first sequence because even if starting at time zero, it will finish after time τ. In this case, function z' is undefined, in Step 2(b) we set $z_j(x) \equiv z''(x)$, and in Step 2(c) we set $y_j(x) \equiv (1+b_j) \cdot y_{j-1}(x)$.

Suppose the infimum in Step 3 is reached at a point x^*. The overall job sequence then is $S = S_1^n(x^*), (\chi), S_2^n(x^*)$ and can be found by backtracking. If in stage j the state's value when performing backtracking from $z_n(x^*)$ (i.e., the corresponding value of x_j^* in $z_j(x_j^*)$) was generated by assigning the value of $z_j'(x_j^*)$ to $z_j(x_j^*)$ (hence $x_j^* = x_{j-1}^* + \ell_j + a_j \cdot (\tau - x_{j-1}^*)$), then we append job j to the end of S_1^{j-1}. If it instead was generated by assigning the value of $z''(x^*)$ to $z_j(x^*)$ (hence $x_j^* = x_{j-1}^*$), we insert job j to the beginning of S_2^{j-1}. In Step 3, the straddler job χ is appended to the end of $S_1^n(x^*)$, and $S_2^n(x^*)$ is started at $\max\{x^* + p_{\chi}(x^*), \tau\}$ with p_{χ} as in (1). If χ completes strictly before τ, idle time is inserted before starting the second sequence such that it starts precisely at τ; in this case the result is dominated by a solution for another straddler job.

3 Fully Polynomial Time Approximation Scheme

Alon & Halman's (2021) framework is used to derive an FPTAS for the DP. For the ease of presentation, instead of referring directly to Alon & Halman (2021), we cite from the concise summary available in Gawiejnowicz, Halman & Kellerer (2023, Appendix A).

We convert the DP (Algorithm 1) to integer values by multiplying all input numbers $\tau = q_\tau/r_\tau$, $\ell_j = q_{\ell_j}/r_{\ell_j}$, $a_j = q_{a_j}/r_{a_j}$, $b_j = q_{b_j}/r_{b_j}$ by $M := r_\tau \prod_{j=1}^{n} (r_{\ell_j} \cdot r_{a_j} \cdot (r_{a_j} - q_{a_j}) \cdot r_{b_j})$. Note that the factor $(r_{a_j} - q_{a_j})$ turns $1/(1-a_j)$ in (3a) into an integer, since $1/(1-a_j) = 1/(1-q_{a_j}/r_{a_j}) = r_{a_j}/(r_{a_j} - q_{a_j})$. Doing so, the state space of x becomes the integer interval $[0, 1, \ldots, \tau M]$. Moreover, we divide the output value in (3e) by M to get back the unscaled objective value. Therefore, the DP is solved in $\mathcal{O}(n \cdot \tau M)$ time, which is to be repeated for each possible straddler job. Observing that M is in $\mathcal{O}(N^{4n+1})$ for

$$N := \max_{j=1,\ldots,n} \{q_\tau, r_\tau, q_{\ell_j}, r_{\ell_j}, q_{a_j}, r_{a_j}, q_{b_j}, r_{b_j}\}, \tag{4}$$

we conclude that the DP runtime is exponential in the number of input items (and is therefore not pseudo-polynomial).

Nevertheless, the DP (Algorithm 1) can be seen as a special case of Gawiejnowicz et al. (2023, Eq. (12)) in the following sense: (i) we set the level index t to be the index j and therefore the number of levels is $T = n$, i.e., the number of jobs to schedule; (ii) regarding DP equation (12) in Gawiejnowicz et al. (2023), we set $f_{t,1} = z_t$ and $f_{t,2} = y_t$, therefore the other index i is either 1 or 2, and $m = 2$; (iii) we set the state variable $I_{t,i}$ to be x, i.e., the completion time of the sequence of jobs that are executed before time τ; (iv) for every pair of levels t and i we set the additional information $A_{t,i}(x)$ to be the conditions stated in step 2 of the DP, which determine the values of $f_{t,i}$ in each case; (v) when considering level t in Gawiejnowicz et al. (2023, Eq. (12)), instead of using all previously calculated $\{z_{r,j}\}_{0 \le r < t, 1 \le j \le 2}$, we use only $\{z_{r,j}\}_{r=t-1, 1 \le j \le 2}$; (vi) we set the boundary functions to be $f_{0,1} \equiv 0$ and $f_{0,2} \equiv 1$. Thus, from (i)–(vi), we conclude that DP (Algorithm 1) is indeed a special case of Gawiejnowicz et al. (2023, Eq. (12)).

Next, we set a bound U_z on the ratio between the maximal value of functions $z_j(\cdot), y_j(\cdot)$ and their minimal non-zero value to be $U_z \leq (MN)^n$ (e.g., the product $\prod_{j=1}^{n}(1+b_j)$ after the scaling), and a bound U_S on the cardinality of the state space to be $U_S = MN$.

Functions $y_j(x)$ and $z_j(x)$, for $j = 1, \ldots, n$, are monotone non-increasing, since as x grows the problem becomes less constrained, i.e., there is more space available for scheduling jobs between 0 and x. Therefore, the DP (Algorithm 1) is monotone and Condition A.1 in Gawiejnowicz et al. (2023) is satisfied; as well, it can be shown that Conditions 2–4(i) are satisfied, which grant us an approximated DP. Its last step is polynomial since computing the infimum in (3e) corresponds to calculating the minimum in the pseudo-polynomial state space $[0, 1, \ldots, M\tau]$ while the approximated functions y_n, z_n are step functions with a polynomial number of steps. Finally, we use parameter value $\tau_f \in \mathcal{O}(n)$ to apply Theorem 4 in Gawiejnowicz et al. (2023) for each possible straddler job, and obtain an FPTAS.

Theorem 1. *There exists an FPTAS to minimize the makespan $C_{\max}$ on a single machine with time-dependent V-shaped processing times that runs in $\mathcal{O}\Big(\frac{n^6}{\varepsilon} \cdot \log^2 N \cdot \log \frac{n \log N}{\varepsilon}\Big)$ time, where N is the maximal value of the numbers in the input, as defined in equation* (4).

This runtime is by $1/\varepsilon$ (up to log terms) lower than Sedding's (2020*a*) FPTAS runtime, which is in $\mathcal{O}\Big(\frac{n^5}{\varepsilon^2} \cdot \log(1 + b_{\max}) \cdot (\log(1 + b_{\max}) + n \cdot \log(1 + b_{\max}))\Big)$.

References

Alon, T. & Halman, N. (2021), ‘Automatic generation of FPTASes for stochastic monotone dynamic programs made easier’, *SIAM Journal on Discrete Mathematics* **35**(4), 2679–2722.

Cai, J.-Y., Cai, P. & Zhu, Y. (1998), ‘On a scheduling problem of time deteriorating jobs’, *Journal of Complexity* **14**(2), 190–209.

Cheng, T. C. E., Ding, Q., Kovalyov, M. Y., Bachman, A. & Janiak, A. (2003), ‘Scheduling jobs with piecewise linear decreasing processing times’, *Naval Research Logistics* **50**(6), 531–554.

Gawiejnowicz, S. (2020), ‘A review of four decades of time-dependent scheduling: Main results, new topics, and open problems’, *Journal of Scheduling* **23**(1), 3–47.

Gawiejnowicz, S., Halman, N. & Kellerer, H. (2023), ‘Knapsack problems with position-dependent item weights or profits’, *Annals of Operations Research* **326**(1), 137–156.

Halman, N. (2020), ‘A technical note: Fully polynomial time approximation schemes for minimizing the makespan of deteriorating jobs with nonlinear processing times’, *Journal of Scheduling* **23**(6), 643–648.

Halman, N., Klabjan, D., Mostagir, M., Orlin, J. & Simchi-Levi, D. (2009), ‘A fully polynomial-time approximation scheme for single-item stochastic inventory control with discrete demand’, *Mathematics of Operations Research* **34**(3), 674–685.

Ji, M. & Cheng, T. C. E. (2007), ‘An FPTAS for scheduling jobs with piecewise linear decreasing processing times to minimize makespan’, *Information Processing Letters* **102**(2-3), 41–47.

Kononov, A. V. (1997), ‘On schedules of a single machine jobs with processing times nonlinear in time’, *Discrete Analysis and Operational Research* **391**, 109–122.

Kovalyov, M. Y. & Kubiak, W. (1998), ‘A fully polynomial approximation scheme for minimizing makespan of deteriorating jobs’, *Journal of Heuristics* **3**(4), 287–297.

Kovalyov, M. Y. & Kubiak, W. (2012), ‘A generic FPTAS for partition type optimisation problems’, *International Journal of Planning and Scheduling* **1**(3), 209–233.

Kubiak, W. & van de Velde, S. L. (1998), ‘Scheduling deteriorating jobs to minimize makespan’, *Naval Research Logistics* **45**(5), 511–523.

Sedding, H. A. (2020*a*), ‘Scheduling jobs with a V-shaped time-dependent processing time’, *Journal of Scheduling* **23**(6), 751–768.

Sedding, H. A. (2020*b*), *Time-Dependent Path Scheduling: Algorithmic Minimization of Walking Time at the Moving Assembly Line*, Springer Vieweg, Wiesbaden.

Sedding, H. A. (2023), ‘Mixed-model moving assembly line material placement optimization for a shorter time-dependent worker walking time’, *Journal of Scheduling*.