

RESEARCH ARTICLE

A New Graphical Approach for Solving Single Machine Scheduling Problems Approximately

Evgeny R. Gafarov ^{a,b*}, Alexandre Dolgui^b and Frank Werner^c

^a*Ecole Nationale Supérieure des Mines, FAYOL-EMSE, CNRS:UMR6158, LIMOS, F-42023 Saint-Etienne, France;*

^b*Institute of Control Sciences of the Russian Academy of Sciences, Profsoyuznaya st. 65, 117997 Moscow, Russia;*

^c*Fakultät für Mathematik, Otto-von-Guericke-Universität Magdeburg, PSF 4120, 39016 Magdeburg, Germany*

(v1.0 released January 2013)

Often the problem of determining an optimal or approximate production schedule in a company can be reduced to the problem of solving a scheduling problem on a bottleneck machine. However, even the majority of the resulting single machine scheduling problems are NP-hard from a computational point of view and therefore, it is difficult to solve large instances of such problems exactly. In this paper, we consider five such single machine problems, where a dynamic programming algorithm of pseudo-polynomial complexity exists. The running time of such an algorithm can often be improved by so-called graphical algorithms which do not need to consider all states in a dynamic programming algorithm separately. Based on such graphical algorithms, we present fully polynomial-time approximation schemes (FPTASes) for five single machine total tardiness problems. The new FPTASes have the best running time among the known approximation schemes for these problems. The presented approach is rather general and can be applied to many other scheduling and combinatorial optimization problems as well.

Keywords: Single machine scheduling, Total weighted tardiness, Graphical algorithm, Fully polynomial-time approximation scheme

1. Introduction

Single machine scheduling problems often appear as sub-problems in real-world optimization, e.g., in more realistic shop scheduling problems with several identical or non-identical

*Corresponding author. Email: axel73@mail.ru

machines, in assembly line balancing, in project scheduling, in routing, etc. On the other hand, they have their own applications as well. We can consider a single machine as a whole plant, where each job can be interpreted as a particular customer order. The majority of single machine scheduling problems are hard to solve (i.e., they are NP-hard), and their instances appear in an instance of a real-world optimization problem for an exponential number of times. So, the use of exact algorithms to solve such practical scheduling problems seems to be often ineffective, and fast approximation algorithms with a guaranteed relative or absolute error can be preferable. One of the first classical reviews on single machine scheduling was presented in Maxwell (1964). Some single machine scheduling problems and solution algorithms to solve them were considered, e.g., in Bechara et al. (1981), Tanaev et al. (1994), Wanga et al. (2013). The gap between classical and real-world scheduling problems was discussed in MacCarthy et al. (1993).

In this paper, several total tardiness scheduling problems on a single machine are considered. These problems can be formulated as follows. We are given a set $N = \{1, 2, \dots, n\}$ of n independent jobs that must be processed on a single machine. Job preemption is not allowed. The machine can handle only one job at a time. All jobs are assumed to be available for processing at time 0. For each job $j \in N$, a processing time $p_j > 0$, a weight $w_j > 0$ and a due date $d_j > 0$ are given.

A feasible solution is described by a permutation $\pi = (j_1, j_2, \dots, j_n)$ of the jobs of the set N from which the resulting schedule can be uniquely determined by starting each job as early as possible. Let $C_{j_k}(\pi) = \sum_{l=1}^k p_{j_l}$ be the completion time of job j_k in the schedule resulting from the sequence π . If $C_j(\pi) > d_j$, then job j is tardy. If $C_j(\pi) \leq d_j$, then job j is on-time. Moreover, let $T_j(\pi) = \max\{0, C_j(\pi) - d_j\}$ be the tardiness of job j in the schedule resulting from the sequence π and let $GT_j(\pi) = \min\{T_j(\pi), p_j\}$ be the late work (i.e., the amount of processing of this job that is performed after its due date).

In the total tardiness minimization problem the objective is to find an optimal job sequence π^* that minimizes total tardiness, i.e., the objective function is $F(\pi) = \sum_{j=1}^n T_j(\pi)$. We denote this problem by $1||\sum T_j$ according to the traditional three-field notation $\alpha|\beta|\gamma$ for scheduling problems, where α describes the machine environment, β gives the job characteristics and further constraints and γ describes the objective function.

Similarly, for the total weighted tardiness problem $1||\sum w_j T_j$ the objective function $F(\pi) = \sum_{j=1}^n w_j T_j(\pi)$ and for the $1||\sum GT_j$ tardiness problem the objective function $F(\pi) = \sum_{j=1}^n GT_j(\pi)$ have to be minimized. The following special cases of the problems are considered in addition:

- the special case denoted $B-1$ (Lazarev et al. (2007)) of the total tardiness problem $1||\sum T_j$, where $p_1 \leq p_2 \leq \dots \leq p_n$, $d_1 \geq d_2 \geq \dots \geq d_n$ and $d_1 - d_n \leq p_1$;
- the special case $B-1G$ (Lazarev et al. (2007)) of the problem $1||\sum T_j$ with $d_{max} - d_{min} \leq p_{min}$, where $d_{max} = \max_{j \in N}\{d_j\}$, $d_{min} = \min_{j \in N}\{d_j\}$ and $p_{min} = \min_{j \in N}\{p_j\}$;
- minimizing total weighted tardiness when all due dates are equal, i.e., $d_j = d$, $j = 1, 2, \dots, n$. This problem is denoted by $1|d_j = d|\sum w_j T_j$.

We also consider the NP-hard problem of maximizing total weighted tardiness $1(no-idle)||\max \sum w_j T_j$ (Gafarov et al. (2010, 2013a)). Here, the objective is to find an optimal job sequence that maximizes total weighted tardiness, where each feasible schedule starts at time 0 and there is no idle time between the processing of the jobs (for practical interpretations and applications, see e.g. Gafarov et al. (2010, 2013a)).

A survey on the problem $1||\sum T_j$ was presented in Koulamas (2010). The problem $1||\sum T_j$ is NP-hard in the ordinary sense (Du et al. (1990), Lazarev et al. (2006)). A

pseudo-polynomial dynamic programming algorithm of time complexity $O(n^4 \sum p_j)$ was proposed in Lawler (1977). The algorithms in Szwarc et al. (1999) can solve special instances (Potts et al. (1982)) of this problem for $n \leq 500$ jobs. For the NP-hard special case $B-1$ (Lazarev et al. (2006)), a pseudo-polynomial algorithm is known (Lazarev et al. (2007)). The special case $B-1$ and its generalization $B-1G$ are known to belong to the most difficult instances for Lawler's algorithms as computational experiments in (Lazarev et al. (2007)) have shown. For the NP-hard problem $1||\sum GT_j$, pseudo-polynomial algorithms are known as well (Lawler et al. (1969), Gafarov et al. (2012b)). A single machine common due date problem was considered in Benmansourab et al. (2012).

Since the main topic of this paper is that of an analysis of approximation algorithms, we recall some relevant definitions. For the scheduling problem of minimizing a function $F(\pi)$, a polynomial-time algorithm that finds a feasible solution π' such that $F(\pi')$ is at most $\rho \geq 1$ times the optimal value $F(\pi^*)$ is called a ρ -approximation algorithm; the value of ρ is called a worst-case ratio bound. If a problem admits a ρ -approximation algorithm, it is said to be approximable within a factor ρ . A family of ρ -approximation algorithms is called a fully polynomial-time approximation scheme (FPTAS), if $\rho = 1 + \varepsilon$ for any $\varepsilon > 0$ and the running time is polynomial with respect to both the length of the problem input and $1/\varepsilon$. Notice that a problem, which is NP-hard in the strong sense, admits no FPTAS unless $P = NP$.

For the problem $1||\min\sum T_j$, Lawler (1982) converted his dynamic programming algorithm into an FPTAS that requires $O(n^7/\varepsilon)$ time. For its special case $B-1$, an FPTAS with a running time of $O(n^3 \log n + n^3/\varepsilon)$ was mentioned in Koulamas (2010). For the problem $1|d_j = d|\sum w_j T_j$, which is NP-hard in the ordinary sense (Yuan (1992)), Fathi et al. (1990) provided a 2-approximation algorithm that requires $O(n^2)$ time. An FPTAS for the latter problem with a running time of $O((n^6 \log \sum w_j)/\varepsilon^3)$ was presented in Kellerer et al. (2006). In Kacem (2010), a heuristic algorithm was presented, which was incorrectly called an FPTAS with a running time of $O(n^2/\varepsilon)$. This algorithm is based on the assumption that there is an optimal schedule, where the tardy jobs are scheduled in non-decreasing order of the values p_j/w_j . This does not hold for the first tardy job, so it is an heuristics. Nevertheless, we see that this algorithm can be easily transformed into an FPTAS. The corrected FPTAS will have the same time complexity of $O(n^3/\varepsilon)$ as our FPTAS presented in this paper. However, as we discuss later, our approach has some other advantages. Moreover, we present a general idea how some single machine scheduling problems can be solved approximately in an effective way. In addition, we show how our graphical algorithm and the FPTAS can be modified to solve other scheduling problems.

For a practical realization of some pseudo-polynomial algorithms, one can use the idea of a *graphical algorithm* (see e.g. Gafarov et al. (2012a)). In this paper, we present such a graphical modification of a pseudo-polynomial algorithm and an FPTAS based on this graphical algorithm with the best known running time for five single machine scheduling problems, namely, for the cases $B-1$, $B-1G$, $1|d_j = d|\sum w_j T_j$ and the problems $1||\sum GT_j$ and $1(no-idle)||\max\sum w_j T_j$. A brief overview on the detailed results presented in this paper was also given in Gafarov et al. (2013b).

Although any of the problems mentioned above can be used to explain and illustrate the approach presented, we selected the problem $1|d_j = d|\sum w_j T_j$ as one of the most popular problems of this type in the literature, which is treated in Sections 2 - 4. The FPTAS presented for this problem has the same running time as the FPTAS of Kacem (2010) after corrections, but some obvious advantages of our scheme are presented in Section 4, e.g., our FPTAS can be modified to solve scheduling problems with a variable

starting time of the machine (see Hoogeveen et al. (2010)), which is hard to do with other known algorithms.

The remainder of this paper is organized as follows. In Section 2, an exact dynamic programming pseudo-polynomial algorithm is presented. A graphical algorithm resulting from this dynamic programming algorithm is given in Section 3. Its advantages comparing with different known dynamic programming algorithms are discussed in Section 4. In Section 5, an FPTAS based on the graphical algorithm is derived. Modifications of the graphical algorithm and the FPTASes for the other considered in this paper problems are described in Section 6. Some further remarks are given in Section 7.

2. Dynamic Programming Algorithm

In this section, we present a property of an optimal sequence and an exact algorithm for the problem $1|d_j = d|\sum w_j T_j$ which is the base for the modification in Section 3. The idea of this algorithm can be easily adapted to the other problems under consideration (see Section 6).

Lemma 2.1: *Lawler et al. (1969) There exists an optimal job sequence π for the problem $1|d_j = d|\sum w_j T_j$ that can be represented as a concatenation (G, x, H) , where all jobs $j \in H$ are tardy and $S_j \geq d$ for all $j \in H$, and all jobs $i \in G$ are on-time. All jobs from the set G are processed in non-increasing order of the values $\frac{p_i}{w_i}$ and all jobs from the set H are processed in non-decreasing order of the values $\frac{p_i}{w_i}$. The job x is the first job, for which $C_x(\pi) \geq d$.*

The job x is called the *straddling* job. Assume that the jobs are numbered as follows:

$$\frac{p_2}{w_2} \leq \frac{p_3}{w_3} \leq \dots \leq \frac{p_n}{w_n}, \quad (1)$$

where the job with number 1 is the straddling job. As a corollary from Lemma 2.1, there is a straddling job $x \in N$ to which the number 1 will be assigned such that for each $l \in \{1, 2, \dots, n\}$, there exists an optimal schedule in which all jobs $j \in \{1, 2, \dots, l\}$ are processed from time t one by one, and there is no job $i \in \{l+1, l+2, \dots, n\}$ which is processed between these jobs. Thus, we can present a dynamic programming algorithm (DPA) based on Lemma 2.1. For each $x \in N$, we number the jobs from the set $N \setminus \{x\}$ according to (1) and perform Algorithm 1. Then we choose a best schedule among the n constructed schedules. At each stage $l, 1 \leq l \leq n$, of Algorithm 1, we construct a best partial sequence $\pi_l(t)$ for the set of jobs $\{1, 2, \dots, l\}$ and for each possible starting time t of the first job (which represents a possible state in the dynamic programming algorithm). $F_l(t)$ denotes the total weighted tardiness value for the job sequence $\pi_l(t)$. $\Phi^1(t)$ and $\Phi^2(t)$ are temporary functions, which are used to compute $F_l(t)$.

Algorithm 1

1. FOR $t := 0$ TO $\sum_{j=2}^n p_j$ DO
 - $\pi_1(t) := (1), F_1(t) := w_1 \max\{0, p_1 + t - d\};$
2. FOR $l := 2$ TO n DO
 - FOR $t := 0$ TO $\sum_{j=l+1}^n p_j$ DO
 - $\pi^1 := (l, \pi_{l-1}(t + p_l)), \pi^2 := (\pi_{l-1}(t), l);$

$$\begin{aligned}\Phi^1(t) &:= w_l \max\{0, p_l + t - d\} + F_{l-1}(t + p_l); \\ \Phi^2(t) &:= F_{l-1}(t) + w_l \max\left\{0, \sum_{j=1}^l p_j + t - d\right\}; \\ \text{IF } \Phi^1(t) &< \Phi^2(t) \text{ THEN } F_l(t) := \Phi^1(t) \text{ and } \pi_l(t) := \pi^1, \\ \text{ELSE } F_l(t) &:= \Phi^2(t) \text{ and } \pi_l(t) := \pi^2;\end{aligned}$$

3. $\pi_n(0)$ is an optimal job sequence for the chosen job x with the objective function value $F_n(0)$.

Theorem 2.2: *By using Algorithm 1 for each $x \in N$, an optimal job sequence of the type described in Lemma 2.1 can be found in $O(n^2 \sum p_j)$ time.*

Proof. We prove the theorem indirectly. Assume that there exists an optimal job sequence of the form $\pi^* = (G, 1, H)$ of the type described in Lemma 2.1. Assume that $F(\pi^*) < F(\pi_n(0)) = F_n(0)$.

Let $\pi' := \pi^*$. For each $l = 1, 2, \dots, n$, we successively consider the part $\bar{\pi}_l \in \pi'$ of the schedule where $\{\bar{\pi}_l\} = \{1, 2, \dots, l\}$. Let $\pi' = (\pi_\alpha, \bar{\pi}_l, \pi_\beta)$ and $t^* = \sum_{i \in \pi_\alpha} p_i$. If $\bar{\pi}_l \neq \pi_l(t^*)$, then $\pi' := (\pi_\alpha, \pi_l(t^*), \pi_\beta)$. It is obvious that $F((\pi_\alpha, \bar{\pi}_l, \pi_\beta)) \geq F((\pi_\alpha, \pi_l(t^*), \pi_\beta))$. Applying this procedure to subsequent values l , we have $F(\pi^*) \geq F(\pi') = F_n(0)$ at the end. Thus, the schedule $\pi_n(0)$ is also optimal for the chosen job $x = 1$.

The assignments $\pi^1 := (l, \pi_{l-1}(t + p_l))$, $\pi^2 := (\pi_{l-1}(t), l)$ and $\pi_l(t) := \pi^1$ require $O(l)$ operations if we deal with sequences of length l . Let $x_l = 1$ mean that l is added as the first job and let $x_l = 0$ mean that l is added as the last job in a job sequence. Then $\pi_l(t)$ can be presented as a 0-1 sequence, i.e., like a number in binary format. So, a partial solution can be presented like a number. Let $\pi_{l-1}(t + p_l)$ be presented like a number X_1 and $\pi_{l-1}(t)$ be presented like a number X_2 . Then π^1 will be presented like $X_1 + 2^l$ and π^2 like X_2 . So, these assignments can be done in constant time.

Obviously, the time complexity of Algorithm 1 is equal to $O(n \sum p_j)$. So, an optimal job sequence can be found in $O(n^2 \sum p_j)$ time. \square

Here the following comments should be given. It is obvious that for some chosen job $x \in N$ in the job sequence $\pi_n(0)$, the job x cannot be straddling (i.e., either $S_x \geq d$ or $C_x < d$ holds). This means that there exists another straddling job $x' \in N$ for which the value $F_n(0)$ will be less. Note that for all problems considered in this paper, a partial solution can be presented like a number, i.e., the assignments $\pi^1 := (l, \pi_{l-1}(t + p_l))$, $\pi^2 := (\pi_{l-1}(t), l)$ and $\pi_l(t) := \pi^1$ can be performed in constant time.

Next we show how Algorithm 1 can be modified to improve the running time to $O(n^2d)$ or $O(n^2UB)$, where UB is an upper bound on the optimal function value for the problem. The explanations given for the time complexity $O(n^2UB)$ are later used in the description of the FPTAS. Algorithm 1 can be modified by considering for each $l = 1, 2, \dots, n$, only the interval $[0, d - p_l]$ instead of the interval $[0, \sum_{i=l+1}^n p_i]$ since for each $t > d - p_l$, job l is tardy in any partial sequence $\pi_l(t)$ and the partial sequence $\pi^2 := (\pi_{l-1}(t), l)$ is optimal. Thus, the time complexity of the modified Algorithm 1 is equal to $O(nd)$, and an optimal schedule can be found in $O(n^2d)$ time which is equal to the running time of the solution algorithm for the problem presented in Lawler et al. (1969).

The values $F_l(t) = \sum_{j=1}^l w_j \max\{C_j(\pi_l(t)) - d, 0\}$ can be calculated using the starting time t and the completion times $C_j(\pi_l(t))$ calculated according to the job sequence $\pi_l(t)$. Since all parameters p_j, w_j for all $j \in N$, and d are integer, we can state the following.

Proposition 2.3: *For all integer t , the values $F_l(t)$ are integer.*

Let UB be an upper bound on the optimal function value for the problem which is found by the 2-approximation algorithm in Fathi et al. (1990), i.e., $UB \leq 2F(\pi^*)$. If we have $F_l(t_l^{UB}) = UB$ for some $t_l^{UB} \in (-\infty, +\infty)$, then for each $t > t_l^{UB}$ we have $F_l(t) > UB$ (since t denotes the starting time of an optimal schedule obtained from the job sequence $\pi_l(t)$ for the jobs $1, 2, \dots, l$ and $F_l(t)$ denotes the corresponding value of the monotonic objective function). So, the states $t > t_l^{UB}$ seem to be unpromising, i.e., for any job sequence π constructed using these states, we will have $F(\pi) > UB$, i.e., π is not optimal. Thus, we need to consider different values $F_l(t)$ only for $t \in [0, t_l^{UB}]$ and assume that $F_l(t) = +\infty$ for $t \in (t_l^{UB}, +\infty)$. According to Proposition 2.3, there are at most $UB + 2$ different values $F_l(t)$.

In addition, we can note the following. If there is a point $t' \in (-\infty, +\infty)$ such that $F_l(t') = 0$ and $F_l(t' + 1) > 0$, then $F_l(t) = 0$ for all $t \leq t'$ and $F_l(t) < F_l(t + 1)$ for all $t \geq t' + 1$, since all the functions $F_l(t)$ are monotonic. Thus, we can modify Algorithm 1 as follows. If we will save at each stage l instead of all states $t \in [0, t']$ only one state t' , then the number of saved states will be restricted by UB , since for all saved states t we have $F_l(t) < F_l(t + 1)$. The running time of the modified algorithm is $O(n \min\{d, UB\})$. If we consider only the states $t \in [d - \sum_{j=1}^n p_j, d]$ instead of $[0, d]$ at each stage $l, l = 1, 2, \dots, n$, then we obtain an optimal solution for the chosen straddling job for each possible starting time $t \in (-\infty, t_n^{UB})$ in $O(nUB)$ time.

Let π' be a job sequence, where all n jobs are processed in non-decreasing order of the values $\frac{p_j}{w_j}$. Denote by $F(\pi', d)$ the total weighted tardiness for the job sequence π' , where the processing of the jobs starts at time d . It is obvious that for this starting time the schedule π' is optimal. Then, by using the modified Algorithm 1, we can obtain an optimal solution for each possible starting time $t \in (-\infty, +\infty)$ in $O(n^2 F(\pi', d))$ time.

We note that the inequality $F_l(t) < F_l(t + 1)$ does not necessarily hold for all $t > t'$ in the problem 1|| $\sum GT_j$, i.e., the running time of Algorithm 1 is not restricted by UB for the problem 1|| $\sum GT_j$.

3. Graphical Algorithm

In this section, a new graphical algorithm (GrA) is derived which is based on an idea from Gafarov et al. (2012a). The idea of this algorithm can be easily adapted to the other problems under consideration (see Section 6). This GrA is a modification of Algorithm 1, in which the function $F_l(t)$ is defined for any $t \in (-\infty, +\infty)$ (not only for integer t). However, we need to compute these values only at the *break* points separating intervals in which the function $F_l(t)$ is a linear function of the form $F_l(t) = F_l^k(t) = u_l^k \cdot (t - t_l^{k-1}) + b_l^k$. In Proposition 3.2, we prove that $F_l(t)$ is a continuous piecewise linear function on the interval $[0, t_l^{UB}]$ whose parameters can be described in a tabular form (as in Table 1). We remind that in the modified Algorithm 1 (see the previous section), we have $F_l(t) = +\infty$ for $t \in (t_l^{UB}, +\infty)$. The same holds for the functions $F_l(t)$ considered in this section.

In each step of the GrA, we store the information on the function $F_l(t)$ for a number of intervals (characterized by the same best partial sequence and by the same total weight of tardy jobs) in a tabular form as given in Table 1.

In Table 1, k denotes the number of the current interval whose values range from 1 to $m_l + 1$ (where the number of intervals $m_l + 1$ is defined for each $l = 1, 2, \dots, n$), $(t_l^{k-1}, t_l^k]$ is the k th interval (where $t_l^0 = -\infty$ and $t_l^{m_l+1} = t_l^{UB}$), b_l^k, u_l^k are the parameters of the linear function $F_l^k(t)$ defined in the k th interval, and π_l^k is the best sequence of the first l jobs if they are processed from time $t \in (t_l^{k-1}, t_l^k]$.

These data mean the following. For each above interval, we store the parameters b_l^k and u_l^k for describing the function $F_l(t)$ and the resulting best partial sequence if the first job starts in this interval. For each starting time $t \in (t_l^{k-1}, t_l^k]$ ($t_l^0 = -\infty$) of the first job, we have a best partial sequence π_l^k of the jobs $1, 2, \dots, l$ with a total weight of the tardy jobs u_l^k and the function value $F_l(t) = u_l^k \cdot (t - t_l^{k-1}) + b_l^k$ (see Fig. 1(a)). We have $F_l(t) = 0$ for $t \in (t_l^0, t_l^1]$. Recall that the function $F_l(t)$ is defined not only for integers t , but also for real numbers t . For simplicity of the following description, we consider the whole t -axis, i.e., $t \in (-\infty, +\infty)$. The points $t_l^1, t_l^2, \dots, t_l^{m_l+1}$ are called *break points*, since there is a change from value u_l^{k-1} to u_l^k (which means that the slope of the piecewise-linear function changes). Note that some of the break points t_l^k can be non-integer. To describe each linear segment, we store its slope u_l^k and its function value $b_l^k = F_l(t)$ at the point $t = t_l^{k-1}$. So, in the table $b_l^1 = b_l^2 < b_l^3 < \dots < b_l^{m_l+1} < UB$ holds, since $t_l^1 < t_l^2 < \dots < t_l^{m_l+1}$.

In the GrA, the functions $F_l(t)$ reflect the same functional equations as in Algorithm 1, i.e., for each $t \in Z \cap [0, \sum_{j=2}^n p_j]$, the function $F_l(t)$ has the same value as in Algorithm 1 (see Fig. 1), but these functions are now defined for any $t \in (-\infty, +\infty)$. As a result, often a large number of integer states is combined into one interval (describing a new state in the resulting algorithm) with the same best partial sequence. In Fig. 1 (a), the function $F_l(t)$ from the GrA is presented and in Fig. 1 (b), the function $F_l(t)$ from Algorithm 1 is displayed.

Next, the GrA is described. The core is Step 3, where we explain how the states at stage $l, l > 1$, are obtained if the states at stage $l - 1$ are known.

Graphical algorithm (GrA)

Step 1. We number the jobs according to order (1);

Step 2. Set $l := 1$, $\pi_1(t) := (1)$, $F_1(t) := w_1 \max\{0, p_1 + t - d\}$ for all t . We represent the function $F_1(t)$ in a tabular form as given in Table 2. For all three intervals, there is the same best partial sequence (1). For $t \in (-\infty, d - p_1]$, there is no tardy job in the schedule corresponding to sequence (1) when this job is started at time t and for $t \in (d - p_1, +\infty)$, there is one tardy job. The value t_1^{UB} can be found from the equation $UB = (t_1^{UB} - (d - p_1))w_1 + 0$.

Step 3. Let $l > 1$ and assume that function $F_{l-1}(t)$ and the best partial sequences of the jobs $\{1, 2, \dots, l - 1\}$ for all resulting intervals given in Table 3 are known, where $t_{l-1}^{m_{l-1}+1} = t_{l-1}^{UB}$.

In the following, we describe how the function $F_l(t)$ is obtained by means of the function $F_{l-1}(t)$. Note that we can store the temporary functions $\Phi^1(t)$ and $\Phi^2(t)$ determined in Steps 3.1 and 3.2 also in a tabular form as in Table 1.

Step 3.1. The function $\Phi^1(t)$ is obtained from the function $F_{l-1}(t)$ by the following operations. We shift the diagram of the function $F_{l-1}(t)$ to the left by the value p_l and in the table for the function $F_{l-1}(t)$ and add a column which results from the new break point $t' = d - p_l$. If $t_{l-1}^s - p_l < t' < t_{l-1}^{s+1} - p_l$, $s \leq m_{l-1}$, then we have two new intervals of t in the table for $\Phi^1(t)$: $(t_{l-1}^s - p_l, t']$ and $(t', t_{l-1}^{s+1} - p_l]$. This means that we first replace each interval $(t_{l-1}^k, t_{l-1}^{k+1}]$ by $(t_{l-1}^k - p_l, t_{l-1}^{k+1} - p_l]$ in the table for $\Phi^1(t)$, and then replace the column with the interval $(t_{l-1}^s - p_l, t_{l-1}^{s+1} - p_l]$ by two new columns with the intervals $(t_{l-1}^s - p_l, t']$ and $(t', t_{l-1}^{s+1} - p_l]$.

Moreover, we increase the values $u_{l-1}^{s+1}, u_{l-1}^{s+2}, \dots, u_{l-1}^{m_{l-1}+1}$ by w_l , i.e., the total weight of tardy jobs (and thus the slope of the function) increases. The corresponding partial sequences π^1 are obtained by adding the job l as the first job to each previous partial sequence. In this way, we obtain the information on the function $\Phi^1(t)$ together with the corresponding partial sequences given in Table 4, which also includes the calculation of the corresponding b values.

Step 3.2. The function $\Phi^2(t)$ is obtained from the function $F_{l-1}(t)$ by the following operations. In the table for the function $F_{l-1}(t)$, we add a column which results from the new break point $t'' = d - \sum_{i=1}^l p_i$. If $t_{l-1}^h < t'' < t_{l-1}^{h+1}$, $h \leq m_{l-1}$, then there are two new intervals of t in the table for the functions $\Phi^2(t)$: $(t_{l-1}^h, t'']$ and $(t'', t_{l-1}^{h+1}]$.

This means that we replace the column with the interval $(t_{l-1}^h, t_{l-1}^{h+1}]$ by two new columns with the intervals $(t_{l-1}^h, t'']$ and $(t'', t_{l-1}^{h+1}]$.

Moreover, we increase the values $u_{l-1}^{h+1}, u_{l-1}^{h+2}, \dots, u_{l-1}^{m_{l-1}+1}$ by w_l , i.e., the total weight of the tardy jobs increases. The corresponding partial sequences π^2 are obtained by adding job l at the end to each previous partial sequence. In this way, we obtain the information on the function $\Phi^2(t)$ together with the corresponding partial sequences given in Table 6.

Step 3.3. Now we construct a table that corresponds to the function

$$F_l(t) = \min\{\Phi^1(t), \Phi^2(t)\}.$$

To construct the function $F_l(t)$ as the above minimum of two functions, we consider the functions $\Phi^1(t)$ and $\Phi^2(t)$ on all resulting intervals from both tables and search for intersection points of the diagrams of these functions.

To be more precise, let $T^1 = \{t_{l-1}^1 - p_l, t_{l-1}^2 - p_l, \dots\}$ be the list of all break points from the table $\Phi^1(t)$ and $T^2 = \{t_{l-1}^1, t_{l-1}^2, \dots\}$ be the list of all break points from the table $\Phi^2(t)$, then T^1 and T^2 are merged into the list T with $T = \{t_1, t_2, \dots, t_e\}$, $t_1 < t_2 < \dots, t_e$. Then we consider each interval $(t_i, t_{i+1}]$, $i = 1, 2, \dots, e-1$, and compare the two functions $\Phi^1(t)$ and $\Phi^2(t)$ over this interval. Let the interval $(t_i, t_{i+1}]$ be contained in the interval $(t_{z-1}^{\Phi^1}, t_z^{\Phi^1}]$ from the table for the function $\Phi^1(t)$ and in the interval $(t_{y-1}^{\Phi^2}, t_y^{\Phi^2}]$ from the table for the function $\Phi^2(t)$. Then $\Phi^1(t) = (t - t_{z-1}^{\Phi^1}) \cdot u_z^{\Phi^1} + b_z^{\Phi^1}$ and $\Phi^2(t) = (t - t_{y-1}^{\Phi^2}) \cdot u_y^{\Phi^2} + b_y^{\Phi^2}$. If there is no intersection point of the functions $\Phi^1(t)$ and $\Phi^2(t)$ in the interval $(t_i, t_{i+1}]$, then insert a new column into the table for $F_l(t)$ with this interval, where the value $b = \min\{b_z^{\Phi^1}, b_y^{\Phi^2}\}$ and the corresponding value of u ($u_z^{\Phi^1}$ or $u_y^{\Phi^2}$) are given. If there exists an intersection point t''' of the functions $\Phi^1(t)$ and $\Phi^2(t)$ in this interval, then insert two columns with the intervals $(t_i, t''']$ and $(t''', t_{i+1}]$.

This step requires $O(m_{l-1})$ operations.

Step 3.5. If in the table for the function $F_l(t)$, there are two columns with the intervals $(t_l^{k-1}, t_l^k]$ and $(t_l^k, t_l^{k+1}]$, $u_l^k = u_l^{k+1}$ and $b_l^k + u_l^k \cdot (t_l^k - t_l^{k-1}) = b_l^{k+1}$, then we assume $t_l^k := t_l^{k+1}$ and delete the column $k+1$. So, in this step we combine two adjoining linear fragments that are on the same line.

Step 3.6. Let m be the number of columns in the resulting table of $F_l(t)$ and for k , $1 \leq k < m_l$, the inequality $b_l^k < UB \leq b_l^{k+1}$ holds. Then compute the value t_l^{UB}

from the equality $UB = (t_l^{UB} - t_l^{k-1})u_l^k + b_l^k$. In the column with the interval $(t_l^{k-1}, t_l^k]$ (column k), assign $t_l^k = t_l^{UB}$ and delete all columns $k + 1, k + 2, \dots, m$.

Step 3.7. If $l = n$, then GOTO Step 4 else $l := l + 1$ and GOTO Step 3.

Step 4. In the table corresponding to the function $F_n(t)$, we determine the column $(t_n^k, t_n^{k+1}]$, where $t_n^k < 0 \leq t_n^{k+1}$. Then we have an optimal sequence $\pi^* = \pi_n^{k+1}$ for the chosen job x and the optimal function value $F(\pi^*) = b_n^{k+1} + (0 - t_n^k) \cdot u_n^{k+1}$.

Next, we analyze some properties of the GrA. According to the GrA, it is obvious that at each integer point $t \in (-\infty, t_l^{UB}]$, the value $F_l(t)$ is equal to that determined by Algorithm 1 (since in the GrA, the functions $F_l(t)$ represent the same equations as in Algorithm 1 for the integer values t considered). For some chosen straddling jobs x , it is possible that $F_n(0) = +\infty$. This means that the best job sequence with the chosen straddling job x is not better than a job sequence π^{UB} for which $F(\pi^{UB}) < F(\pi_n(0))$. Therefore, by using the GrA for each $x \in N$, an optimal job sequence of the type described in Lemma 2.1 for some x with the optimal value $F_n(0)$ will be found and we can state the following.

Proposition 3.1: *By using the GrA for each $x \in N$, an optimal job sequence of the type described in Lemma 2.1 will be found.*

We note that all functions $F_l(t)$, $l = 1, 2, \dots, n$, defined at the beginning of Section 3 are continuous and piecewise linear functions on the interval $(-\infty, t_l^{UB}]$. It is obvious that the function $F_1(t)$ is a continuous and piecewise linear function with two break points in the interval $(-\infty, t_1^{UB}]$. According to the operations described in Step 3, both functions $\Phi^1(t)$ and $\Phi^2(t)$ are also continuous and piecewise linear functions on the intervals $(-\infty, t_1^{UB} - p_2]$ and $(-\infty, t_1^{UB}]$. Thus, the function

$$F_2(t) = \min\{\Phi^1(t), \Phi^2(t)\}$$

and its modification obtained in Step 3.4. are continuous and piecewise linear functions on the interval $(-\infty, t_2^{UB}]$ as well (we can show that $t_l^{UB} \leq t_{l-1}^{UB}$). Continuing in this way, all functions $F_l(t)$, $l = 1, 2, \dots, n$, have the above properties. So, the following statement is true.

Proposition 3.2: *All functions $F_l(t)$, $l = 1, 2, \dots, n$, are continuous and piecewise linear functions on the interval $(-\infty, t_l^{UB}]$.*

Now assume that we have obtained Table 7 for some function $F_l(t)$ in Step 3.

We prove that $b_l^1 = b_l^2 < b_l^3 < \dots < b_l^s < b_l^{s+1} < \dots < b_l^{m_i+1} = UB$ holds. Assume that we have $b_l^s \geq b_l^{s+1}$, $s > 1$. Let $F(\pi, t)$ be the total weighted tardiness value of the sequence π when the first job starts at time t . It is obvious that the function $F(\pi, t)$ is monotonic. For each $t \in (t_l^{s-1}, t_l^s]$, we have $F(\pi_l^s, t) \geq F(\pi_l^{s+1}, t)$ since for $t \in (t_l^s, t_l^{s+1}]$, the inequality $F(\pi_l^s, t) \geq F(\pi_l^{s+1}, t)$ holds, which is a contradiction. Then according to Step 3.5. of the GrA, we can state the following.

Proposition 3.3: *The number of columns in the table for the function $F_l(t)$ is less than or equal to the number of different values $b_l^{m_i+1}$ in the table plus 1.*

The GrA can be modified as follows to avoid fractional break points. If a point t''' calculated in Step 3.3. is fractional, then we can include the following two columns into the table for the function $F_l(t)$: $(t_i, \lfloor t''' \rfloor]$ and $(\lceil t''' \rceil, t_{i+1}]$. Denote this algorithm by GrA-I.

Theorem 3.4: *By using the GrA-I for each $x \in N$, an optimal job sequence of the type described in Lemma 2.1 will be found in $O(n^2 \min\{F^*, d\})$ time, where F^* is the optimal objective function value.*

Proof.

It is obvious that the running time of the GrA depends polynomially on the numbers of columns $m_l + 1$. According to Proposition 2.3, all values b_l^s in the table are integer. Then, according to Proposition 3.3 in the resulting table for the function $F_l(t)$, there will be no more than $UB + 2$ columns, since $b_l^s \in [0, UB]$ and b_l^s is integer. The GrA-I can be modified to consider only intervals in $[0, d]$ (see the previous section). Since all t_l^s considered are integer, there will be no more than $d + 1$ columns.

Then Step 3 requires $O(\min\{UB, d\})$ operations for each $l = 1, 2, \dots, n$. The time complexity of such a search is $O(n^2 F^*)$, since $\frac{1}{2}UB \leq F^* \leq UB$.

□

The required memory for the algorithms GrA-I and DPA is $O(\min\{UB, d\})$.

4. Advantages of the Graphical Algorithm

In fact, in each step $l = 1, 2, \dots, n$ of the GrA, we do not consider all points $t \in [0, d]$, $t \in Z$, but only points from the interval in which the optimal partial solution changes or where the resulting functional equation of the objective function changes. So, the main difference is that we operate *not with independent values F at each of the points t , but with functions which are transformed in each step analytically (according to their analytical form)*, which can have obvious advantages. For example, let us minimize a function $\Psi(t) + F(\pi, t)$, where the function $F(\pi, t)$ corresponds to a function $F(\pi)$ when the jobs are processed not starting from time 0, but from time t . If the function $F(t)$ is presented analytically (not in a tabular form (t, F)) and the function $\Psi(t)$ is presented analytically as well, then the search for the minimum of $\Psi(t) + F(\pi, t)$ can be performed in shorter time for some cases.

Moreover, such an approach has the following advantages when compared with Algorithm 1 (DPA):

1. The GrA can solve instances, where (some of) the parameters p_j , w_j , $j = 1, 2, \dots, n$ or/and d are not in Z .
2. The running time of the GrA for two instances with the parameters $\{p_j, w_j, d\}$ and $\{p_j \cdot 10^{const} \pm 1, w_j \cdot 10^{const} \pm 1, d \cdot 10^{const} \pm 1\}$, $const > 1$, is the same while the running time of the DPA will be 10^{const} times larger in the second case. Thus, one can usually solve considerably larger instances with the GrA. Here, ± 1 means that we add or subtract 1 to each parameter to ensure that the reduction by a factor 10^{const} is impossible.
3. For some single machine scheduling problems, the GrA has a reduced running time in comparison with the DPA (see Table 11 in Section 6).
4. The experimental results provided in Gafarov et al. (2012b) show a rather polynomial evolution of the calculation time in experiments for the GrA, since the total number of columns considered does not exceed n^2 .
5. It is obvious that the running time (not the complexity) of Algorithm GrA-I is always less than the running time of the DPA. When the number of break points is less than $const \cdot d$, we perform GrA-I, and from step l when it does not hold, we use the DPA.

6. Unlike the DPA, it is possible to construct easily an FPTAS based on the GrA. The FPTAS is presented in the next section.

Let us consider another type of a DPA. This alternative DPA does not present a new result, but we give it here in order to compare our main result (GrA) with an alternative way how the DPA can be organized. So, here we present the recurrent expression and a sketch of this algorithm, which can be easily substantiated. This algorithm iteratively generates some sets of states. In every iteration l , $l = n, n - 1, \dots, 1$, a set of states is generated. Each state can be represented by a string of the form (t, F) , where t is the completion time of the last known job scheduled at the beginning of a schedule and F is the value of the function, provided that the early jobs start at time 0 and the last known late job completes exactly at time $\sum_{j=1}^n p_j$. This algorithm can be roughly described as follows:

Alternative DPA A straddling job x is chosen and numbered by 1. The other jobs are ordered according to (1).

1. Put the state $(0, 0)$ into the set of states V_{n+1} .
2. FOR $l := n$ TO 1 DO
 - FOR each state (t, F) from the set of states V_{l+1} DO
 - Put a state $[t + p_l, F + w_l \max\{0, t + p_l - d\}]$;
 - Put a state $[t, F + w_l \max\{0, \sum_{j=1}^n p_j - (\sum_{j=1}^{l-1} p_j - t) - d\}]$;
3. Find $F^* = \min\{F | (t, F) \in V_1\}$.

We need to consider only states, where $t \leq d$ and $F \leq UB$. If there are two states with the same objective function value (t_1, F') and (t_2, F') and $t_2 > t_1$ in a list V_l , then the state (t_2, F') can be removed from consideration. So, the running time of the Alternative DPA is $O(n \min\{d, F^*\})$ which corresponds to the running time of GrA-I. However, in GrA-I some of the possible but unpromising states are not considered and, in contrast to the alternative DPA, Algorithm GrA-I finds all optimal schedules for all integer starting times $t \in [-\infty, t_n^{UB}]$ in $O(nF^*)$ time. So, the alternative DPA is not effective for the problems of minimizing a complex function $\Psi(t) + F(\pi, t)$. In addition, with GrA it is possible to solve scheduling problems with a variable starting time of the machine (Hoogeveen et al. (2010)) in an effective way.

5. A Fully Polynomial-Time Approximation Scheme

To explain the idea of the FPTAS, we use the problem $1|d_j = d|\sum w_j T_j$ but this idea can be easily adapted to the other problems under consideration (see Section 6).

The idea of the FPTAS is as follows. Denote $LB = UB/2$ and let $\delta = \frac{\varepsilon LB}{n} = \frac{\varepsilon UB}{2n}$. To reduce the time complexity of the GrA, we have to diminish the number of columns considered, which is the number of different objective function values $0 = b_l^1 = b_l^2, b_l^3, \dots, b_l^{m_l+1} = UB$. If we consider not the original values b_l^k but the values $\overline{b_l^k}$ which are rounded up or down to the nearest multiple of δ values b_l^k , there are no more than $\frac{UB}{\delta} = \frac{2n}{\varepsilon}$ different values $\overline{b_l^k}$. Then we will be able to convert the table $F_l(t)$ into a similar table with no more than $4\frac{n}{\varepsilon}$ columns. Furthermore, for such a modified table (function) $F'(t)$, we will have $|F(t) - F'(t)| < \delta \leq \frac{\varepsilon F(\pi^*)}{n}$. If we do the rounding and modification after each Step 3.5. of the GrA, then the cumulative error will be no more than $n\delta \leq \varepsilon F(\pi^*)$, and the total running time of the n runs of the GrA will be $O(\frac{n^3}{\varepsilon})$, i.e., an FPTAS is obtained.

By transforming the GrA, we store the approximated functions $F_l^A(t)$ in the same tabular form but without the last row which describes an optimal partial job sequence π_l^k . The data from the last row are stored in a tabular form as described in Step 3.6. (see below). These n stored tables, corresponding to the n last rows, are used to restore an approximate job sequence.

FPTAS (as a modification of the GrA).

In the modified GrA, Steps 1, 2, 3.1.–3.5. remain the same. Instead of Step 3.6, we use the following steps. Assume that we have obtained Table 7 for a function $F_l(t)$ in Step 3.5 with the objective function values $b_l^1, b_l^2, \dots, b_l^{m_l+1}$.

Step 3.6. Save Table 8.

We have $position_l^k := 0$ if in the partial sequence π_l^k , job l is the first job and $position_l^k = 1$, if job l is the last one (there are only these two possibilities).

Step 3.7. If $m_l \leq 4\frac{n}{\varepsilon}$, then GOTO 3.8. else do the following. Let $\overline{b_l^k}$ denote the result of rounding the value b_l^k from Table 7 to the nearest multiple of δ . Then we have $\overline{b_l^1} \leq \overline{b_l^2} \leq \dots \leq \overline{b_l^{m_l+1}}$.

We modify the table $F_l^A(t)$ as follows. Assume that, for $k_1 < k_2$, we have $\overline{b_l^{k_1}} < \overline{b_l^{k_1+1}} = \dots = \overline{b_l^{k_2}} < \overline{b_l^{k_2+1}}$. We substitute the columns corresponding to the values $\overline{b_l^{k_1}}, \dots, \overline{b_l^{k_2-1}}$ for the two columns presented in Table 9.

Step 3.8. If $l = n$, then GOTO 4 else $l := l + 1$ and GOTO 3.

End of the modification of the graphical algorithm.

Let us analyze the substitution proposed in Step 3.7. Let $\overline{F}_l(t)$ be the function obtained after Step 3.4 in the modified algorithm. In fact, the function $F_l(t)$ was modified in Step 3.7 in the way shown in Fig. 2. Let $F_l^A(t)$ describe the modified function.

Lemma 5.1: For all $t \in (t_l^{k_1-1}, t_l^{k_2-1}]$, we have $|F_l^A(t) - \overline{F}_l(t)| < \delta/2$.

Proof. It is only necessary to consider the values $|F_l^A(t) - \overline{F}_l(t)|$ at the break points $t_l^{k_1-1}, t_l^{k_1}, \dots, t_l^{k_2-1}$. At these points, the inequality holds and these points are end points of continuous and piecewise linear segments of the functions $\overline{F}_l(t)$ and $F_l^A(t)$. \square

Assume that the functions $F_l(t)$, $l = 1, 2, \dots, n$, are exact and constructed by the original GrA. Similarly, $F_l^A(t)$, $l = 1, 2, \dots, n$, are the approximated functions constructed by the modified algorithm (the functions after Step 3.7 of the modified algorithm).

Lemma 5.2: For each l , $l = 1, 2, \dots, n$, and all $t \in (-\infty, +\infty)$, we have $|F_l^A(t) - F_l(t)| \leq l \cdot \delta/2$.

Proof. The proof is accomplished by induction. The inequality holds for $l = 1$. Assume that it holds for $l - 1$, $1 < l < n$, i.e., for $t \in (-\infty, +\infty)$, we have $|F_{l-1}^A(t) - F_{l-1}(t)| \leq (l - 1)\delta/2$. Let the functions $\Phi^1(t)$ and $\Phi^2(t)$ be obtained from the function $F_{l-1}(t)$ and the functions $\phi^1(t)$ and $\phi^2(t)$ be obtained from the function $F_{l-1}^A(t)$. Then we have $|\phi^1(t) - \Phi^1(t)| \leq (l - 1)\delta/2$ and $|\phi^2(t) - \Phi^2(t)| \leq (l - 1)\delta/2$. Thus,

$$|F_l^A(t) - F_l(t)| \leq |(\min\{\phi^1(t), \phi^2(t)\} + \delta/2) - \min\{\Phi^1(t), \Phi^2(t)\}| \leq l \cdot \delta/2.$$

So, the lemma is true. \square

An approximate job sequence π' can be restored by backward recursion from the tables stored in Step 3.6. of the modified GrA as follows. In the table of the positions of job n , we determine the column $(t_n^{k_n}, t_n^{k_n+1}]$, where $t_n^{k_n} < 0 \leq t_n^{k_n+1}$. If we have an optimal sequence with $position_n^{k_n+1} = 0$, job n is the first job in the schedule and for the position of the job $n - 1$, we search in the column $(t_{n-1}^{k_{n-1}}, t_{n-1}^{k_{n-1}+1}]$ in the table of the positions of job $n - 1$, where $t_{n-1}^{k_{n-1}} < p_n \leq t_{n-1}^{k_{n-1}+1}$. Otherwise, job n is the last job and for the position of the job $n - 1$, we search in the column $(t_{n-1}^{k_{n-1}}, t_{n-1}^{k_{n-1}+1}]$, where $t_{n-1}^{k_{n-1}} < 0 \leq t_{n-1}^{k_{n-1}+1}$. We restore the job sequence π' . Continuing in such a way, we restore the job sequence π' in $O(n \log(n/\varepsilon))$ time. Let π^* be an optimal job sequence for the chosen straddling job x .

Lemma 5.3: *The inequality $F(\pi') - F(\pi^*) \leq n \cdot \delta \leq n \frac{2\varepsilon \cdot F(\pi^*)}{2n}$ holds.*

Proof. According to Lemma 5.1, we prove that $|F(\pi') - F_n^A(0)| \leq n \cdot \delta/2$. Furthermore, according to Lemma 5.2, we have $|F_n^A(0) - F_n(0)| \leq n \cdot \delta/2$, where $F_n(0) = F(\pi^*)$. From both propositions we establish that the lemma is true. \square

So, we can conclude the following.

Theorem 5.4: *By using the modified graphical algorithm for each $x \in N$, a job sequence π' of the type described in Lemma 2.1 will be found in $O(\frac{n^3}{\varepsilon})$ time, where $F(\pi') \leq (1 + \varepsilon)F(\pi^*)$.*

The running time $O(\frac{n^3}{\varepsilon})$ is obtained as follows. The modified GrA is used n times for each job $x \in N$. The running time of the modified GrA depends on n and the number of columns in the tables which describe the functions $F_l^A(t)$. The number of columns does not exceed $O(\frac{n}{\varepsilon})$.

We note that some of the break points can be fractional. However, since we deal with approximate solutions, these points can be approximated as well.

6. Algorithms for the Remaining Single Machine Problems

In this section, modifications of the algorithms DPA, GrA and an FPTAS for two special cases of the problem $1||\sum T_j$ and for the problems $1||\sum GT_j$ and $1(no-idle)||\max \sum w_j T_j$ are presented.

Lemma 6.1: *There exists an optimal job sequence π for the special case $B - 1G$ that can be represented as a concatenation (G, x, H) , where all jobs $j \in H$ are tardy and all jobs $i \in G$ are on-time. All jobs from the set G are processed in non-increasing order of the values p_j (longest processing time order, LPT) and all jobs from the set H are processed in non-decreasing order of the values p_j (shortest processing time order, SPT).*

Again, the job x is called the straddling job.

Proof. Let in an optimal job sequence $\pi' = (\pi_1, l, k, \pi_2)$, job k be the first tardy job. Then all jobs from the partial sequence π_2 are tardy and processed in SPT order, since $d_{max} - d_{min} < p_{min}$ (which can be easily proved by contradiction). If $S_k > d_{min}$, then all jobs in the partial sequence (k, π_2) have to be processed in SPT order and the jobs from π_1 can be processed in any order (e.g., in LPT order). In this case, l is the straddling job. If $S_k \leq d_{min}$, then all jobs in the partial sequence (π_1, l) can be reordered by the LPT rule without loss of optimality. In this case, k is the straddling job. \square

Lemma 6.2: Lazarev (2007) There exists an optimal job sequence π for the special case $B - 1$ that can be represented as a concatenation (G, H) . All jobs from the set G are processed in LPT order and all jobs from the set H are processed in SPT order.

Assume that for cases $B - 1$ and $B - 1G$, the jobs are numbered as follows: $p_1 \leq p_2 \leq \dots \leq p_n$.

Lemma 6.3: For the special cases $B - 1$ and $B - 1G$ and a job sequence $\pi_{SPT} = (1, 2, \dots, n)$, the following inequality holds: $F(\pi_{SPT}) \leq 3F(\pi^*)$.

Proof. Consider a modified instance, where all due dates are equal to d_{max} . Let $\overline{F}(\pi)$ be the total tardiness value for the modified instance and $F(\pi)$ be the value for the original one. Let in the job sequence $\pi_{SPT} = (1, 2, \dots, k, \dots, n)$, job k be the first tardy job for the original instance and $k < n$ (the case $k = n$ is trivial). It is obvious that the job sequence π_{SPT} is optimal for the modified instance. The following inequality holds: $\overline{F}(\pi_{SPT}) \leq F(\pi^*) \leq F(\pi_{SPT})$. Furthermore, $\overline{F}(\pi_{SPT}) \geq F(\pi_{SPT}) - (n - k + 1) \cdot p_{min}$ and $\overline{F}(\pi_{SPT}) \geq (n - k) \cdot p_{min}$ hold. Thus, $3\overline{F}(\pi_{SPT}) \geq F(\pi_{SPT})$, and the lemma is true. \square

Without loss of generality, we will consider only cases where in a job sequence π_{SPT} at least two jobs are tardy.

Lemma 6.4: Gafarov et al. (2012b) There exists an optimal job sequence π for the problem $1||\sum GT_j$ that can be represented as a concatenation (G, H) , where all jobs $j \in H$ are tardy and $GT_j(\pi) = p_j$. For all jobs $i \in G$, we have $0 \leq GT_i(\pi) < p_i$. All jobs from the set G are processed in EDD (earliest due date) order, and all jobs from the set H are processed in LDD (last due date) order.

Let for the problem $1||\sum GT_j$ the jobs be numbered as follows: $d_1 \geq d_2 \geq \dots \geq d_n$ and let $\pi_{EDD} = (n, n-1, \dots, 1)$. Denote by T^* the maximal late work of a job in the sequence π_{EDD} , i.e., $T^* = \max_{j \in N} \{GT_j(\pi_{EDD})\}$. We remind that $GT_j(\pi) = \min\{\max\{0, C_j(\pi) - d_j\}, p_j\}$ and $F(\pi) = \sum_{j=1}^n GT_j(\pi)$.

Lemma 6.5: For the problem $1||\sum GT_j$, the following inequality holds: $F(\pi_{EDD}) \leq nF(\pi^*)$.

Proof. It is easy to show that $F(\pi^*) \geq T^*$. Furthermore, it is obvious that $nT^* \geq F(\pi_{EDD})$. So, the lemma is true. \square

Lemma 6.6: Gafarov et al. (2010, 2013a) There exists an optimal job sequence π for the problem $1(no-idle)||\max \sum w_j T_j$ that can be represented as a concatenation (G, H) , where all jobs $j \in H$ are tardy and all jobs $i \in G$ are on-time. All jobs from the set G are processed in non-increasing order of the values $\frac{w_i}{p_i}$ and all jobs from the set H are processed in non-decreasing order of the values $\frac{w_i}{p_i}$.

For the problem $1(no-idle)||\max \sum w_j T_j$, the value $LB_{maxTT} = \max_{j \in N} (w_j (\sum_{i=1}^n p_i - d_j))$ is a lower bound. Then $UB_{maxTT} = nLB_{maxTT}$ is an upper bound on the optimal objective function value.

To solve these problems, Algorithms 1, GrA and FPTAS can be modified as follows.

For the special case $B - 1G$:

- **DPA.** Use the fact described in Lemma 6.1. In Algorithm 1, we number the jobs according to the order $p_2 \leq p_3 \leq \dots \leq p_n$. Assume that $F_1(t) := \max\{0, p_1 + t - d_1\}$, $\Phi^1(t) := \max\{0, p_l + t - d_l\} + F_{l-1}(t + p_l)$ and $\Phi^2(t) :=$

$F_{l-1}(t) + \max \left\{ 0, \sum_{j=1}^l p_j + t - d_l \right\}$. All other steps of the algorithm remain the same. Remember that $w_j = 1$ for all jobs $j \in N$ in the problem $1 || \sum T_j$. The running time of the modified Algorithm 1 is $O(nd_{max})$. Since it is necessary to consider n straddling jobs $x \in N$, an optimal job sequence can be found in $O(n^2 d_{max})$ time by using the modified Algorithm 1;

- **GrA**. The GrA remains the same. The parameters u_l^k denote the number of tardy jobs which is equal to the total weight of the tardy jobs, since $w_j = 1$ for all $j \in N$. In addition, assume that $UB = F(\pi_{SPT})$. By using the GrA-I, an optimal schedule can be found in $O(n^2 \min\{d_{max}, F^*\})$ time;
- **FPTAS**. In the FPTAS, assume that $\delta = \frac{\varepsilon F(\pi_{SPT})}{3n}$. Since the GrA is without changes, the time complexity of the FPTAS based on the GrA for the special case $B - 1G$ has a running time of $O(n^3/\varepsilon)$, which is less than the running time $O(n^7/\varepsilon)$ of the FPTAS for the general case presented in Lawler (1982).

For the special case $B - 1$:

- **DPA**. Use the fact described in Lemma 6.2. Algorithm 1 is modified as for the special case $B - 1G$. The running time of the modified Algorithm 1 is $O(nd_{max})$. Since there is no straddling job, an optimal job sequence can be found in $O(nd_{max})$ time by using the modified Algorithm 1 only once;
- **GrA**. The GrA remains the same as for the special case $B - 1G$. By the GrA-I, an optimal schedule can be found in $O(n \min\{d_{max}, F^*\})$ time;
- **FPTAS**. The FPTAS remains the same as for the special case $B - 1G$. Since there is no straddling job, the FPTAS for the special case $B - 1$ has a running time of $O(n^2/\varepsilon)$, which is less than the running time of $O(n^3 \log n + n^3/\varepsilon)$ of the FPTAS mentioned in Koullamas (2010).

For the problem $1 || \sum GT_j$:

- **DPA**. Use the fact described in Lemma 6.4. In Algorithm 1, we number the jobs according to the order $d_1 \geq d_2 \geq \dots \geq d_n$. Assume that $F_1(t) := \min\{p_1, \max\{0, p_1 + t - d_1\}\}$, $\Phi^1(t) := \min\{p_l, \max\{0, p_l + t - d_l\}\} + F_{l-1}(t + p_l)$ and $\Phi^2(t) := F_{l-1}(t) + \min \left\{ p_l, \max \left\{ 0, \sum_{j=1}^l p_j + t - d_l \right\} \right\}$. The running time of the modified Algorithm 1 is $O(nd_{max})$. Since there is no straddling job, an optimal job sequence can be found in $O(nd_{max})$ time by using the modified Algorithm 1 only once;
- **GrA**. The GrA remains almost the same. In addition to the break points t' and t'' in Steps 3.1 and 3.2, two new break points $\tau' = d_l$ and $\tau'' = d_l - \sum_{j=1}^{l-1} p_j$ are considered. The slope u_l^k of the function $F_l(t)$ is changed according to the function $\min\{p_l, \max\{0, p_l + t - d_l\}\}$. By the GrA-I, an optimal schedule can be found in $O(n \min\{d_{max}, nF^*\})$ time, since $UB = F(\pi_{EDD}) \leq nF(\pi^*)$ and there are at most $2UB + 2$ columns in each table $F_l(t)$ considered in the GrA;
- **FPTAS**. In the FPTAS, we assume that $\delta = \frac{\varepsilon F(\pi_{EDD})}{n^2}$. So, the FPTAS has a running time of $O(n^3/\varepsilon)$.

For the problem $1(no-idle) || \max \sum w_j T_j$:

- **DPA**. We use the fact described in Lemma 6.6. In Algorithm 1, we enumerate

- the jobs according to the order $\frac{w_1}{p_1} \leq \frac{w_2}{p_2} \leq \dots \leq \frac{w_n}{p_n}$. We assume that $F_1(t) := w_1 \max\{0, p_1 + t - d_1\}$, $\Phi^1(t) := w_l \max\{0, p_l + t - d_l\} + F_{l-1}(t + p_l)$ and $\Phi^2(t) := F_{l-1}(t) + w_l \max\left\{0, \sum_{i=1}^l p_i + t - d_l\right\}$. Since total tardiness is maximized, we have $F_l(t) := \max\{\Phi^1(t), \Phi^2(t)\}$. The running time of the modified Algorithm 1 is $O(nd_{max})$. Since there is no straddling job, an optimal job sequence can be found in $O(nd_{max})$ time by using the modified Algorithm 1 only once;
- **GrA.** The GrA remains the same as for the problem $1|d_j = d|\sum w_j T_j$. In Step 3.3., we have $F_l(t) = \max\{\Phi^1(t), \Phi^2(t)\}$. In Gafarov et al. (2010, 2013a), it is shown that the functions $F_l(t)$ represent continuous, piecewise-linear and convex functions. By the GrA-I, an optimal schedule can be found in $O(n \min\{d_{max}, nF^*, \sum w_j\})$ time.
 - **FPTAS.** In the FPTAS, assume that $\delta = \frac{\varepsilon UB_{maxTT}}{n^2}$. So, the FPTAS has a running time of $O(n^3/\varepsilon)$.

7. Some Further Remarks

Some further graphical algorithms for other single machine problems were presented in Gafarov et al. (2010). For some of these problems, similar FPTASes can be presented. If there exists a GrA with a running time of $O(n^\alpha UB)$ for a problem, then it is easy to construct a similar FPTAS with a time complexity of $O(n^\gamma + \frac{n^{\alpha+\beta}}{\varepsilon})$, where UB is an upper bound which is no more than $O(n^\beta)$ times greater than the optimal objective function value, computed in $O(n^\gamma)$ time, where α, β, γ are constants. Thus, the running time of such an FPTAS depends on the relative error of the upper bound found, i.e., on β . For the special cases $B - 1$, $B - 1G$ and for the problem $1|d_j = d|\sum w_j T_j$, we have $\beta = 0$ but for the remaining problems, we have $\beta = 1$, since $UB/LB = n$.

In Chubanov et al. (2006), Kovalyov (1995), a technique was proposed to improve the complexities of approximation algorithms for optimization problems. The technique can be described as follows. If there is an FPTAS for a problem with a running time bounded by a polynomial $P(L, \frac{1}{\varepsilon}, \frac{UB}{LB})$, where L is the problem instance length and UB, LB are known upper and lower bounds, and the value $\frac{UB}{LB}$ is not bounded by a constant, then the technique enables us to find in $P(L, \log \log \frac{UB}{LB})$ time values UB_0 and LB_0 such that $LB_0 \leq F^* \leq UB_0 < 3LB_0$, i.e., $\frac{UB_0}{LB_0}$ is bounded by the constant 3. By using such values UB_0 and LB_0 , the running time of the FPTAS will be reduced to $P(L, \frac{1}{\varepsilon})$, where P is the same polynomial.

If we use this technique for an FPTAS for the problems $1(no-idle)|\max \sum w_j T_j$ and $1|\sum GT_j$, we have a running time of $O(n^2 \log \log n + \frac{n^2}{\varepsilon})$ for the modified FPTAS.

We note that the modified Algorithm 1 (see the end of Section 2) can be considered as a base for an FPTAS as well. The running time of the modification is restricted by UB under the assumption that there is only one interval with the same objective function value $F_l(t)$ although in the proposed FPTAS, there are several intervals with the same objective function value. So, it seems to be more difficult to transform the modification into an FPTAS. The running time of such a modification will be increased by the factor $O(\log \frac{n}{\varepsilon})$ in comparison with the FPTAS based on the GrA.

In Table 10, a comparison of Algorithm 1 (classical DPA), GrA and the alternative DPA is presented. Denote $\Theta_l = \{x_1 p_1 + x_2 p_2 + \dots + x_l p_l | x_1, x_2, \dots, x_l \in \{0, 1\}\}$. Let $t_n^{LB} \in (-\infty, +\infty)$ be a value with $F_n(t_n^{LB}) = LB$. In Table 11, some results for the GrA

and the FPTASes are summarized.

In Gafarov et al. (2012b), an experimental analysis of the running time of the graphical algorithms for two NP-hard single machine problems was presented. According to the experimental results, for a significant part of the instances considered the number of columns (i.e., the running time) in the graphical algorithms does not exceed $O(n^2)$. We can conclude that for such instances the modified graphical algorithm (FPTAS) will find exact solutions. So, the modified graphical algorithms remain exact for a significant part of the instances.

8. Conclusion

In this paper, FPTAS was presented, which can be used with some simple modifications for several single machine scheduling problems. This is based on an approach from Gafarov et al. (2012a, 2010). The idea of such a modification of graphical algorithms enables us to construct an FPTAS easily. The FPTAS presented has some advantages, e.g., in solving problems with a variable starting time.

The graphical approach can be applied to problems, where a pseudo-polynomial algorithm exists and Boolean variables are used in the sense that yes/no decisions have to be made (e.g., in the scheduling problems under consideration, a job may be completed on-time or not or for a knapsack problem, an item can be put into the knapsack or not). For the knapsack problem, the graphical algorithm often reduces substantially the number of points to be considered but the time complexity of the algorithm remains pseudo-polynomial. However, for the single machine problem of maximizing total tardiness, the graphical algorithm improved the complexity from $O(n \sum p_j)$ to $O(n^2)$ Gafarov et al. (2012a). Thus, the graphical approach has not only a practical but also a theoretical importance. For future research, it seems to be important to find other applications of the GrA and to compare it with the best known algorithms not based on dynamic programming.

Acknowledgement

This work was partially supported by RFBR (Russian Foundation for Basic Research): 11-08-01321, 11-08-13121. The authors are also grateful to Prof. V. Strusevich for his idea to use the GrA as a base for an FPTAS.

References

- Benmansourab, R., Allaouibc, H. and Artibaab, A., 2012. Stochastic single machine scheduling with random common due date. *International Journal of Production Research*, 50(13), 3560 – 3571.
- Bechara S. B. and Magazine, M. J., 1981. Myopic heuristics for single machine scheduling problems. *International Journal of Production Research*, 19(1), 85–95.
- Chubanov, S., Kovalyov, M.Y., and Pesch, E., 2006. An FPTAS for a Single-Item Capacitated Economic Lot-Sizing Problem with Monotone Cost Structure. *Math. Program.*, Ser. A 106, 453 – 466.
- Du, J., and Leung, Y.-T., 1990. Minimizing Total Tardiness on One Processor is NP-hard. *Math. Oper. Res.*, 15, 483 – 495.

- Fathi, Y., and Nuttle, H.W.L., 1990. Heuristics for the Common Due Date Weighted Tardiness Problem. *IEE Trans*, 22, 215 – 225.
- Gafarov, E.R., Lazarev, A.A., and Werner, F., 2012a. Transforming a Pseudo-Polynomial Algorithm for the Single Machine Total Tardiness Maximization Problem into a Polynomial One. *Annals of Operations Research*, 196(1), 247 – 261.
- Gafarov, E.R., Lazarev, A.A., and Werner, F., 2010. A Modification of Dynamic Programming Algorithms to Reduce the Time Complexity, Preprint 20/10, FMA, OvGU Magdeburg.
- Gafarov, E.R., Lazarev, A.A., and Werner, F., 2012b. A Note on a Single Machine Scheduling Problem with Generalized Total Tardiness Objective Function. *Information Processing Letters*, 112 (3), 72 – 76.
- Gafarov, E.R., Lazarev, A.A., and Werner, F., 2013. Single Machine Total Tardiness Maximization Problems: Complexity and Algorithms, *Annals of Operations Research*, Vol. 207 (1), 121 – 136.
- Hoogeveen, H., and T'Kindt, V., 2010. Minimizing the Number of Late Jobs When the Starting Time of the Machine is Variable. *Proceedings PMS*, 235 – 238.
- Kacem, I., 2010. Fully Polynomial Time Approximation Scheme for the Total Weighted Tardiness Minimization with a Common Due Date. *Discrete Applied Mathematics*, 158, 1030 – 1040.
- Kellerer, H., and Strusevich, V.A., 2006. A Fully Polynomial Approximation Scheme for the Single Machine Weighted Total Tardiness Problem with a Common Due Date. *Theoretical Computer Science*, 369, 230 – 238.
- Koulamas, C., 2010. The Single-Machine Total Tardiness Scheduling Problem: Review and Extensions. *European Journal of Operational Research*, 202, 1 – 7.
- Kovalyov, M.Y., 1995. Improving the Complexities of Approximation Algorithms for Optimization Problems. *Operations Research Letters*, 17, 85 – 87.
- Lawler, E.L., and Moore, J.M., 1969. A Functional Equation and its Application to Resource Allocation and Sequencing Problems. *Management Science*, 16(1), 77 – 84.
- Lawler, E.L., 1977. A Pseudopolynomial Algorithm for Sequencing Jobs to Minimize Total Tardiness. *Ann. Discrete Math.*, 1, 331 – 342.
- Lawler, E.L., 1982. A Fully Polynomial Approximation Scheme for the Total Tardiness Problem, *Operations Research Letters*, 1, 207 – 208.
- Lazarev, A.A., and Gafarov, E.R., 2006. Special Case of the Single-Machine Total Tardiness Problem is NP-hard. *Journal of Computer and Systems Sciences International*, 45(3), 450 – 458.
- Lazarev, A.A., Kvaratskheliya, A.G., and Gafarov, E.R., 2007. Algorithms for Solving the NP-Hard Problem of Minimizing Total Tardiness for a Single Machine. *Doklady Mathematics*, 75(1), 130 – 134.
- Lazarev, A.A., 2007. Solution of the NP-Hard Total Tardiness Minimization Problem in Scheduling Theory. *Computational Mathematics and Mathematical Physics*, 47, 1039 – 1049.
- Maccarthy, B. L., and Liu, J., 1993. Addressing the gap in scheduling research: a review of optimization and heuristic methods in production scheduling. *International Journal of Production Research*, 31(1), 59 –79.
- Maxwell, W.L., 1964. The Scheduling of Single Machine Systems: A Review. *International Journal of Production Research*, 3(3), 177 – 199.
- Potts, C.N., and Van Wassenhove, L.N., 1982. A Decomposition Algorithm for the Single Machine Total Tardiness Problem. *Operations Research Letters*, 1, 363 – 377.
- Szwarc, W., Della Croce, F., and Grosso, A., 1999. Solution of the Single Machine Total

- Tardiness Problem. *Journal of Scheduling*, 2, 55 – 71.
- Tanaev, V.S., Gordon, V.S., Shafranskii, I.M., 1994. Scheduling Theory: Single-Stage Systems, Springer, 372 p.
- Wanga, S., and Liua, M., 2013. A branch and bound algorithm for single-machine production scheduling integrated with preventive maintenance planning. *International Journal of Production Research*, 51(3), 847–868.
- Yuan, J., 1992. The NP-Hardness of the Single Machine Common Due Date Weighted Tardiness Problem. *System Sci. Math. Sci.*, 5, 328 – 333.
- Gafarov, E.R., Dolgui, A., Lazarev, A., and Werner, F., 2013. A Graphical Approach for Solving Single Machine Scheduling Problems Approximately. In: N. Bakhtadze, A. Dolgui, V. Lototsky (Eds.) *Proceedings of the IFAC Conference on Manufacturing Modelling, Management and Control (MIM2013) St Petersburg, Russia, June 19-21, 2013*, Elsevier Science, IFAC-PapersOnline.net (ISSN 1474-6670), 1340 – 1345.

Table 1.: Function $F_l(t)$

k	1	2	...	$m_l + 1$
interval k	$(-\infty, t_l^1]$	$(t_l^1, t_l^2]$...	$(t_l^{m_l}, t_l^{m_l+1}]$
b_l^k	0	b_l^2	...	$b_l^{m_l+1}$
u_l^k	0	u_l^2	...	$u_l^{m_l+1}$
π_l^k	π_l^1	π_l^2	...	$\pi_l^{m_l+1}$

Table 2.: Function $F_1(t)$

k	1	2
interval k	$(-\infty, d - p_1]$	$(d - p_1, t_1^{UB}]$
b_1^k	0	0
u_1^k	0	w_1
π_1^k	(1)	(1)

Table 3.: Function $F_{l-1}(t)$

k	1	2	...	$m_{l-1} + 1$
interval k	$(-\infty, t_{l-1}^1]$	$(t_{l-1}^1, t_{l-1}^2]$...	$(t_{l-1}^{m_{l-1}}, t_{l-1}^{UB}]$
b_{l-1}^k	0	b_{l-1}^2	...	$b_{l-1}^{m_{l-1}+1}$
u_{l-1}^k	0	u_{l-1}^2	...	$u_{l-1}^{m_{l-1}+1}$
π_{l-1}^k	π_{l-1}^1	π_{l-1}^2	...	$\pi_{l-1}^{m_{l-1}+1}$

-

Table 4.: Function $\Phi^1(t)$. Part 1

interval k	$(-\infty, t_{l-1}^1 - p_l]$	$(t_{l-1}^1 - p_l, t_{l-1}^2 - p_l]$...
b_k	0	b_{l-1}^2	...
u_k	0	u_{l-1}^2	...
best	(l, π_{l-1}^1)	(l, π_{l-1}^2)	...
partial sequence	π^1		

Table 5.: Function $\Phi^1(t)$. Part 2

interval k	...	$(t_{l-1}^s - p_l, t']$	$(t', t_{l-1}^{s+1} - p_l]$	$(t_{l-1}^{s+1} - p_l, t_{l-1}^{s+2} - p_l]$...	$(t_{l-1}^{m_{l-1}} - p_l, t_{l-1}^{UB} - p_l]$
b_k	...	b_{l-1}^{s+1}	b'	\tilde{b}_{l-1}^{s+2}	...	$\tilde{b}_{l-1}^{m_{l-1}+1}$
u_k	...	u_{l-1}^{s+1}	$u_{l-1}^{s+1} + w_l$	$u_{l-1}^{s+2} + w_l$...	$u_{l-1}^{m_{l-1}+1} + w_l$
best	...	(l, π_{l-1}^{s+1})	(l, π_{l-1}^{s+1})	(l, π_{l-1}^{s+2})	...	$(l, \pi_{l-1}^{m_{l-1}+1})$
partial sequence	π^1					

$$b' = b_{l-1}^{s+1} + (t' - (t_{l-1}^s - p_l)) \cdot u_{l-1}^{s+1}, \tilde{b}_{l-1}^{s+2} = b' + ((t_{l-1}^{s+1} - p_l) - t') \cdot (u_{l-1}^{s+1} + w_l), \dots, \tilde{b}_{l-1}^{m_{l-1}+1} = \tilde{b}_{l-1}^{m_{l-1}} + (t_{l-1}^{m_{l-1}} - t_{l-1}^{m_{l-1}-1}) \cdot (u_{l-1}^{m_{l-1}} + w_l)$$

-

Table 6.: Function $\Phi^2(t)$

interval k	$(-\infty, t_{l-1}^1]$	$(t_{l-1}^1, t_{l-1}^2]$	\dots	$(t_{l-1}^h, t'']$	$(t'', t_{l-1}^{h+1}]$	$(t_{l-1}^{h+1}, t_{l-1}^{h+2}]$	\dots	$(t_{l-1}^{m_{l-1}}, t_{l-1}^{UB}]$
b_k	0	b_{l-1}^2	\dots	\widehat{b}_{l-1}^{h+1}	b''	\widehat{b}_{l-1}^{h+2}	\dots	$\widehat{b}_{l-1}^{m_{l-1}+1}$
u_k	0	u_{l-1}^2	\dots	u_{l-1}^{h+1}	$u_{l-1}^{h+1} + w_l$	$u_{l-1}^{h+2} + w_l$	\dots	$u_{l-1}^{m_{l-1}+1} + w_l$
best	(π_{l-1}^1, l)	(π_{l-1}^2, l)	\dots	(π_{l-1}^{h+1}, l)	(π_{l-1}^{h+1}, l)	(π_{l-1}^{h+2}, l)	\dots	$(\pi_{l-1}^{m_{l-1}+1}, l)$
partial sequence								
π^2								

$$b'' = b_{l-1}^{h+1} + (t'' - t_{l-1}^h) \cdot u_{l-1}^{h+1}, \widehat{b}_{l-1}^{h+2} = b'' + (t_{l-1}^{h+1} - t'') \cdot (u_{l-1}^{h+1} + w_l), \dots, \widehat{b}_{l-1}^{m_{l-1}+1} = \widehat{b}_{l-1}^{m_{l-1}} + (t_{l-1}^{m_{l-1}} - t_{l-1}^{m_{l-1}-1}) \cdot (u_{l-1}^{m_{l-1}-1} + w_l)$$

-

Table 7.: Function $F_l(t)$

k	1	2	...	s	$s+1$...	m_l+1
interval k	$(-\infty, t_l^1]$	$(t_l^1, t_l^2]$...	$(t_l^{s-1}, t_l^s]$	$(t_l^s, t_l^{s+1}]$...	$(t_l^{m_l}, t_l^{UB}]$
b_l^k	0	b_l^2	...	b_l^s	b_l^{s+1}	...	$b_l^{m_l+1}$
u_l^k	0	u_l^2	...	u_l^s	u_l^{s+1}	...	$u_l^{m_l+1}$
π_l^k	π_l^1	π_l^2	...	π_l^s	π_l^{s+1}	...	$\pi_l^{m_l+1}$

Table 8.: Positions of job l

k	1	2	...	$m_l + 1$	$m_l + 2$
interval k	$(-\infty, t_l^1]$	$(t_l^1, t_l^2]$...	$(t_l^{m_l}, t_l^{UB})$	$(t_l^{UB}, +\infty)$
$position_l^k$	$position_l^1$	$position_l^2$...	$position_l^{m_l+1}$	1

Table 9.: Substitution of columns

interval k	...	$(t_l^{k_1-1}, t_l^{k_1}]$	$(t_l^{k_1}, t_l^{k_2-1}]$...	$(t_l^{UB}, +\infty)$
b_l^k	...	$\frac{b_l^{k_1}}{b_l^{k_2}}$	$b_l^{k_2}$...	$+\infty$
u_l^k	...	$u = \frac{b_l^{k_2} - b_l^{k_1}}{t_l^{k_1} - t_l^{k_1-1}}$	0	...	0

Table 10.: Comparison of the classical DPA, the GrA and the alternative DPA (on the example of the problem $1|d_j = d|\sum w_j T_j$)

Note	Classical DPA	GrA (GrA-I)	Alternative DPA
Can it solve instances with $p_j \notin Z$ and instances with large values p_j	no	yes	yes
states t considered	all $t \in [0, d] \cap Z$	only t , where the slope of function $F_i(t)$ is changed	only t from the set Θ_i
The running time for the initial instance	$O(n \min\{d, UB\})$	$O(n \min\{d, UB\})$	$O(n \min\{d, UB\})$
- of the problem $1 \sum GT_j$ is	$O(nd_{max})$	$O(n \min\{d_{max}, UB\})$	$O(n \min\{d_{max}, UB\})$
- of the problem $1(no-idle) \max\sum w_j T_j$ is	$O(n \min\{d_{max}, UB\})$	$O(n \min\{d_{max}, UB, \sum w_j\})$	$O(n \min\{d_{max}, UB\})$
It finds all optimal schedules for all starting times $t \in [0, d]$ in time	$O(nd)$	$O(nd)$	-
It finds all optimal schedules for all starting times $t \in (-\infty, t_n^{UB}]$ in time	$O(nUB)$	$O(nUB)$	-
It finds all optimal schedules for all starting times $t \in (-\infty, +\infty)$ in time	$O(nF(\pi', d))$ (see Section 3 for the definition of $F(\pi', d)$)	$O(nF(\pi', d))$	-
The running time of the FPTAS is	$O(\frac{n^3}{\varepsilon} \log \frac{n}{\varepsilon})$	$O(n^3/\varepsilon)^*$	$O(n^3/\varepsilon)^{**}$

* In this time, for all $t \in (-\infty, t_n^{UB}]$ solutions can be found with an absolute error restricted by εLB . For all $t \in [t_n^{LB}, t_n^{UB}]$, $t_n^{LB} \leq 0 \leq t_n^{UB}$, solutions can be found with a relative error restricted by ε .

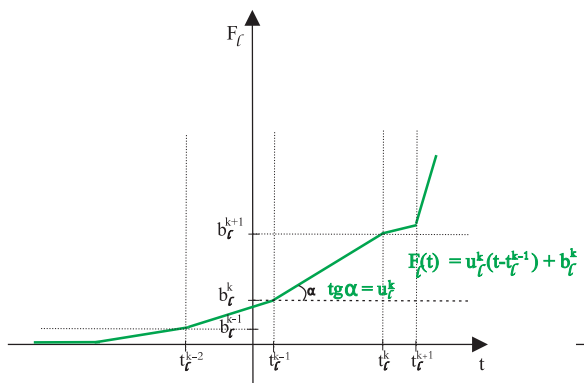
** An approximate solution is only found for the starting time $t = 0$.

Table 11.: Time complexity of the GrA-I and FPTAS based on the GrA-I

Problem	Time complexity of the GrA	Time complexity of the FPTAS	Time complexity of the classical DPA
$1 \sum w_j U_j$	$O(\min\{2^n, n \cdot \min\{d_{max}, F_{opt}\}\})$ Gafarov et al. (2010)	-	$O(nd_{max})$
$1 d_j = d'_j + A \sum U_j$	$O(n^2)$ Gafarov et al. (2010) (GrA)	-	$O(n \sum p_j)$
$1 \sum GT_j$	$O(\min\{2^n, n \cdot \{d_{max}, nF^*\}\})$	$O(n^2 \log \log n + \frac{n^2}{\varepsilon})$	$O(nd_{max})$
$1 \sum T_j$ special case $B-1$	$O(\min\{2^n, n \cdot \min\{d_{max}, F^*\}\})$	$O(n^2/\varepsilon)$	$O(nd_{max})$
$1 \sum T_j$ special case $B-1G$	$O(\min\{n^2 \cdot \min\{d_{max}, F^*\}\})$	$O(n^3/\varepsilon)$	$O(n^2 d_{max})$
$1 d_j = d \sum w_j T_j$	$O(\min\{n^2 \cdot \min\{d, F^*\}\})$	$O(n^3/\varepsilon)$	$O(n^2 d_{max})$
$1(n\alpha\text{-idle}) \max \sum w_j T_j$	$O(\min\{2^n, n \cdot \min\{d_{max}, nF^*, \sum w_j\}\})$ Gafarov et al. (2010)	$O(n^2 \log \log n + \frac{n^2}{\varepsilon})$	$O(nd_{max})$
$1(n\alpha\text{-idle}) \max \sum T_j$	$O(n^2)$ Gafarov et al. (2012a) (GrA)	-	$O(nd_{max})$

-

(a)



(b)

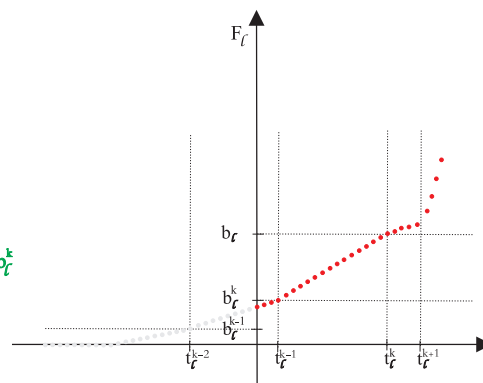


Figure 1.: Function $F_l(t)$ in the GrA and in Algorithm 1

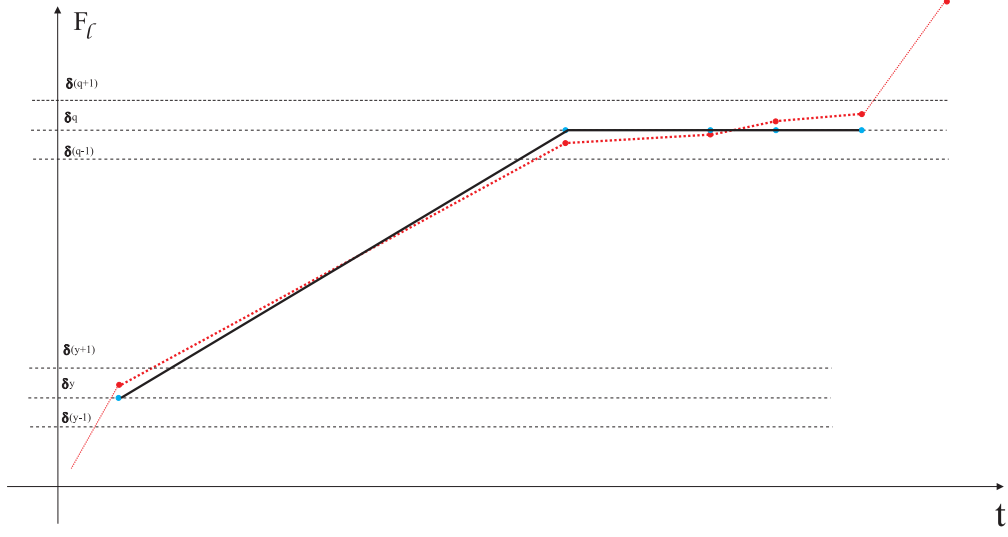


Figure 2.: Substitution of columns and modification of $F_l(t)$