J Math Model Algor (2014) 13:597–614
DOI 10.1007/s10852-013-9248-2

A Graphical Approach to Solve an Investment Optimization Problem

Evgeny R. Gafarov · Alexandre Dolgui · Alexander A. Lazarev · Frank Werner

Received: 16 April 2013 / Accepted: 12 December 2013 / Published online: 8 February 2014
© Springer Science+Business Media Dordrecht 2014

Abstract We consider a project investment problem, where a set of projects and an overall budget are given. For each project, a piecewise linear profit function is known which describes the profit obtained if a specific amount is invested into this project. The objective is to determine the amount invested into each project such that the overall budget is not exceeded and the total profit is maximized. For this problem, a graphical algorithm (GrA) is presented which is based on the same Bellman equations as the best known dynamic programming algorithm (DPA) but the GrA has several advantages in comparison with the DPA. Based on this GrA, a fully-polynomial time approximation scheme is proposed having the best known running time. The idea of the GrA presented can also be used to solve some

E. R. Gafarov · A. Dolgui (✉)
Ecole Nationale Superieure des Mines, FAYOL-EMSE, CNRS:UMR6158, LIMOS, 42023 Saint-Etienne, France
e-mail: dolgui@emse.fr

E. R. Gafarov
e-mail: axel73@mail.ru

E. R. Gafarov · A. A. Lazarev
Institute of Control Sciences of the Russian Academy of Sciences, Profsoyuznaya st. 65, 117997 Moscow, Russia
e-mail: jobmath@mail.ru

A. A. Lazarev
Higher School of Economics (National Research University), Moscow, Russia

A. A. Lazarev
Lomonosov Moscow State University, Moscow, Russia

A. A. Lazarev
Moscow Institute of Physics and Technology (State University), Moscow, Russia

F. Werner
Fakultät für Mathematik, Otto-von-Guericke-Universität Magdeburg, PSF 4120, 39016 Magdeburg, Germany
e-mail: frank.werner@mathematik.uni-magdeburg.de

similar scheduling or lot-sizing problems in a more effective way, e.g., the related problem of finding lot-sizes and sequencing several products on a single imperfect machine.

Keywords Project investment · Graphical algorithm · Pseudo-polynomial time complexity · FPTAS

1 Introduction

The investment optimization problem can be formulated as follows. A set N of n potential projects and an investment budget (amount) $A > 0$ are given. For each project j, $j = 1, \ldots, n$, a profit function $f_j(t)$, $t \in [0, A]$, is defined in such a way that the value $f_j\left(t'\right)$ denotes the profit received if we invest the amount t' into project j. The goal is to define an amount $\tau_j \in [0, A] \bigcap Z$ for each project $j \in N$ such that $\sum_{j:=1}^{n} \tau_j \leq A$ and the total profit $\sum_{j:=1}^{n} f_j(\tau_j)$ is maximized.

Moreover, for a real-world generalization, it is often necessary to find such an optimal solution (investment strategy) for a flexible amount A, i.e., for all $A \in \left[A', A''\right]$. So, one looks for an algorithm which is able to solve this real-world generalization effectively. In the following, we assume that all functions $f_j(t)$, $j = 1, \ldots, n$, are continuous piecewise linear non-decreasing functions with k_j, $j = 1, 2, \ldots, n$, linear fragments (see Fig. 1). If the function $f_j(t)$ is defined only for some particular points t, e.g., for $t_1, t_2, \ldots, t_{k_j}$ with $t_1 < t_2 < \cdots < t_{k_j}$, then we can assume that $f_j(t) = f_j(t_i)$ holds on the interval $[t_i, t_{i+1})$, $i = 1, \ldots, k_j$, where $t_{k_j+1} = A$. In this paper, we propose an effective

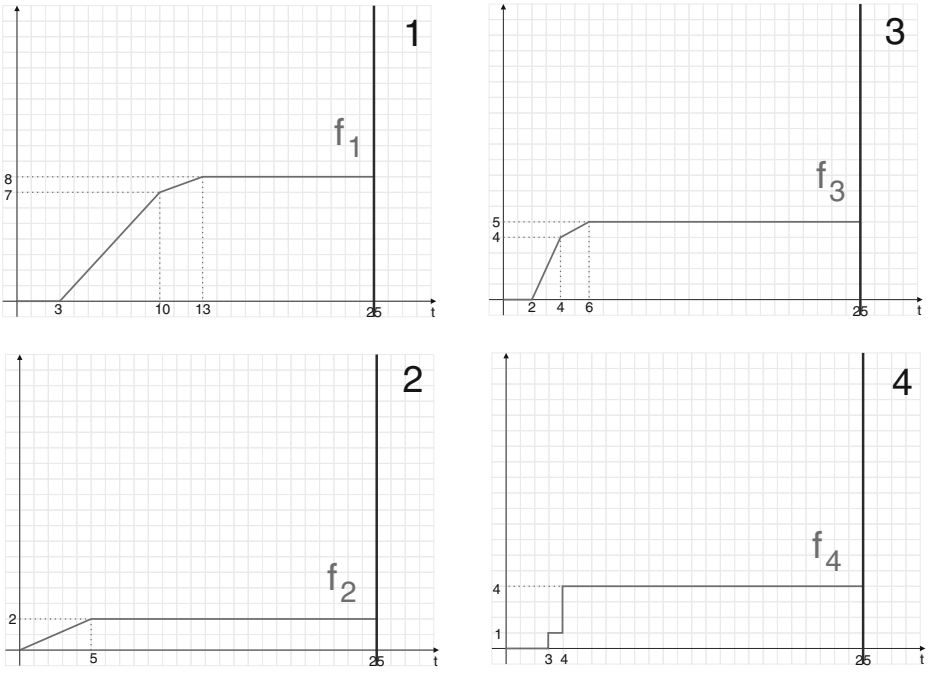

Fig. 1 Functions $f_j(t)$

pseudo-polynomial algorithm and derive a fully polynomial time approximation scheme (FPTAS) based on it.

To the best of our knowledge, there are no publications on the problem under consideration. However, there are some close special cases of knapsack-like problems which are as follows.

A special case of the problem is similar to the well-known bounded knapsack problem [3]:

$$\begin{array}{ll} \text{maximize} & \sum_{j:=1}^{n} p_j x_j \\ s.t. & \sum_{j:=1}^{n} w_j x_j \le A, \\ & x_j \in [0, b_j],\ x_j \in Z,\ j = 1, 2, \ldots, n, \end{array} \tag{1}$$

for which a dynamic programming algorithm with a time complexity of $O(nA)$ is known [3].

The following problem [4] is also similar to the problem under consideration:

$$\begin{array}{ll} \text{minimize} & \sum_{j:=1}^{n} f_j(x_j) \\ s.t. & \sum_{j:=1}^{n} x_j \ge A, \\ & x_j \in [0, A],\ x_j \in Z,\ j = 1, 2, \ldots, n, \end{array} \tag{2}$$

where $f_j(x_j)$ are piecewise linear as well. For the problem (2), a dynamic programming algorithm with a running time of $O\left(\sum k_j A\right)$ [4] and an FPTAS with a running time of $O\left(\left(\sum k_j\right)^3/\varepsilon\right)$ [5] are known. A short survey on this lot-sizing problem can be found in [5] as well.

It is easy to show that the investment optimization problem is NP-hard, since the classical 0-1 knapsack problem is a special case of it.

This problem can be solved by a dynamic programming algorithm (DPA), which is based on the following well-known recursive Bellman equations:

$$F_j(T) = \max_{t=0,1,\ldots,T} \{f_j(t) + F_{j-1}(T-t)\},\ T = A, A-1, \ldots, 1,$$

with the initial conditions

$$\begin{array}{ll} F_0(T) = 0, & \text{for } T \ge 0, \\ F_0(T) = -\infty, & \text{for } T < 0. \end{array} \tag{3}$$

In each stage j, $j = 1, \ldots, n$, a function $F_j(T)$, $j = 1, \ldots, n$, is calculated and used in the next stage. Here the value $F_j(T)$ gives the maximal profit received from the realization of the first j projects if a total amount $T \le A$ is invested. Finally, the value $F_n(A)$ denotes the optimal objective function value. Here A and thus all values of T are integer. Note that, if A would be rational, we can assign $A := \lfloor A \rfloor$.

Usually in a DPA, one works like a calculator, i.e., to calculate $F_j\left(T'\right)$, we consider $T' + 1$ points t and for each point, we calculate the value $f_j(t) + F_{j-1}(T-t)$. Then we choose the maximal value among these $T' + 1$ values. So, the time complexity of such a DPA is $O\left(nA^2\right)$. Here we do not take into account the analytical form of the functions f_j. However, if we work with the analytical representation (not with particular values $f_j(t)$ calculated for $t \in Z$), we will have some advantages.

The remainder of the paper is as follows. In Section 2, we present the graphical algorithm (GrA) and discuss its advantages in comparison with the DPA. In Section 3, we illustrate the algorithm on a numerical example. Some advantages of the GrA in comparison with the DPA are discussed in Section 4. In Section 5, we give a modification of the GrA and derive an FPTAS. In Section 6 a GrA and an FPTAS for a related problem denoted as P-Cost are presented.

2 Graphical Algorithm

Next, we present the graphical algorithm (GrA) which uses the same Bellman equations as the DPA, but considers analytical expressions of the functions. We note that the foundations of such a type of algorithm have been explained e.g. in [1]. The functions $f_j(t)$, $j = 1, 2, \ldots, n$, can be saved in the computer memory 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 k_j (where the number of intervals k_j is defined for each $j = 1, 2, \ldots, n$), $\left[t_j^K\ ,\ t_j^{K+1}\right)$ is the Kth interval, and b_j^K, u_j^K are the parameters of the linear function $f_j^K(t)$ defined on the Kth interval. This data means the following. For each above interval, we store the parameters b_j^K and u_j^K for describing the function $f_j(t) = f_j^K(t) = u_j^K \cdot \left(t - t_j^K\right) + b_j^K$ on this interval.

For the example given in Fig. 1, the function $f_1(t)$ can be presented as follows in Table 2.

The points $t_j^1, t_j^2, \ldots, t_j^{k_j}$ are called "break points" since the slope of the function changes at these points. The functions $F_j(T)$ can be presented in the same way.

First, we formally describe the graphical algorithm (GrA). In the following, we denote the starting points as SP and the counter of the current interval as CI.

Step I. $j := 1$. Copy the table for the function $f_1(t)$ into the table for the function $F_1(T)$.

Step II. a Let $j > 1$ and assume that the function $F_{j-1}(T)$ is known for all resulting intervals. Assign $SP := A$ and $CI := 1$.

Step II. b Calculate the intervals of the *reflected* function $f_j'(t) = f_j(A - t)$ (Table 3). Here $t_1 = SP$, and the other points t_K are calculated as follows: $t_K = t_{K-1} - \left(t_j^{K+1} - t_j^K\right)$, $K = 2, \ldots, k_j$. On the interval $\left[t_{K+1}, t_K\right)$, we have $f_j'(t) = b_j^{K+1} - u_j^K(t - t_{K+1})$, where $f_j'(t)$ is the reflected function.

Step II. c For each point T_K and each point t_K, calculate their equations. *An equation for a point* denotes the values which are used to calculate $F_j(T)$, where T belongs to the interval $[SP - \varepsilon, SP]$.
To calculate the equation for a point T_K, we have to find the interval $[t_{s+1}, t_s)$, where $T_k \in [t_{s+1}, t_s)$. The equation for this point is $f_j'(T_K) + F_j(T_K) - u_j^s \cdot \varepsilon$. In the same way, we calculate the equation for t_K. Let $t_K \in [T_r\ ,\ T_{r+1})$. Then we have the equation $f_j'(t_K) + F_j(t_K) - u^r \cdot \varepsilon$. *We consider only points* $T_K < SP$ *and points* $t_K > 0$.

Step II. d Among all equations, we have to find a *leading equation*, i.e., a *leading point*. Let us consider the two points x^1 and x^2 and their equations $B^{x_1} - U^{x_1}\varepsilon$ and

Table 1 Function $f_j(t)$

K	1	2	...	k_j
interval K	$\left[t_j^1, t_j^2\right)$	$\left[t_j^2, t_j^3\right)$	...	$\left[t_j^{k_j}, A\right)$
b_j^K	b_j^1	b_j^2	...	$b_j^{k_j}$
u_j^K	u_j^1	u_j^2	...	$u_j^{k_j}$

Table 2 Function $f_1(t)$

K	1	2	3	4
interval K	[0, 3)	[3, 10)	[10, 13)	[13, 25]
b_1^K	0	0	7	8
u_1^K	0	1	$\frac{1}{3}$	0

$B^{x_2} - U^{x_2}\varepsilon$. If $B^{x_1} > B^{x_2}$ or ($B^{x_1} = B^{x_2}$ and $U^{x_2} > U^{x_1}$), then the point x_1 dominates the point x_2. For the leading point, there is no other point which dominates it.

Step II. e Calculate the length of the minimal interval $LMI = \min\{t_s - T_r\}$, $s = 1, \ldots, k_j$, $r = 1, \ldots, BP_{j-1}$, where $t_s > T_r$. Now we check whether there is a point x' with an equation $B^{x'} - U^{x'}\varepsilon$, which is dominated by the leading point x^* and whose diagram has an intersection on the interval $[SP - LMI, SP]$. Let $SP - \varepsilon'$ be an intersection point. We have to find the minimal value ε' among all such points x'. Let ε_{min} be such a point. Then $LMI := \varepsilon_{min}$. Such points $SP - LMI$ will be called *intersection points*.

Step II. f In the table for the function $F_j(T)$, save a column with the interval $[t_{CI}, t_{CI+1}] = [SP - LMI, SP]$ and the values $b^{CI} = B^{x^*} - U^{x^*} \cdot LMI$, $u^{CI} = U^{x^*}$.

Step II. g If the slope of the function on the interval of $F_j(T)$ just saved before was the same, then we can join both intervals.

Step II. h $SP := SP - LMI$ and $CI := CI + 1$. If $SP > 0$, then GOTO **Step II. b**.

Step III. $BP_j := CI - 1$. Modify the form of the table for the function $F_j(T)$ into the form as in Table 4. Let $j := j + 1$. If $j \le n$, GOTO **Step II.a**.

Step IV. Use backtracking to find an optimal solution at the point A and the optimal objective function value $F_n(A)$.

Lemma 1 *At each stage j, $j = 1, 2, \ldots, n$, of the algorithm, there are no more than $BP_j + k_j + k_j \cdot BP_j$ intersection points.*

Proof During step II, there are no more than $k_j \cdot BP_j$ different equations (with different points or different slopes). At each point $(SP - LMI)$ calculated according to the points T_K and t_K, only for two points the equations (slopes of their equations) are changed. In the initial point $T = A$, we have only $BP_j + k_j$ equations. All these equations correspond to linear functions. So, there are no more than $BP_j + k_j + k \cdot BP_j$ intersection points. □

Table 3 Intervals of the reflected function $f_j'(t)$

K	1	2	...	k_j
interval K	$[t_2, t_1)$	$[t_3, t_2)$	...	$\left[0, t_j^{k_j}\right)$

Table 4 Function $F_j(T)$

K	1	2	...	BP_j
interval K	$[T_{BP_j+1}, T_{BP_j})$	$[T_{BP_j}, T_{BP_j-1})$	...	$[T_2, T_1)$
b^K	b^1	b^2	...	b^{BP_j}
u^K	u^1	u^2	...	u^{BP_j}

Theorem 1 *The GrA constructs an optimal solution for all points* $T \in [0, A]$ *in* $O\left(n(BP_{max}k_{max})^2\right)$ *time, where* $BP_{max} = \max_{l=1,2,\ldots,n}\{BP_j\}$, $k_{max} = \max_{j\in N} k_j$.

Proof Since the GrA uses the same Bellman equations as the DPA, it founds an optimal solution for any $T \in [0, A]$.

Steps $II.b - II.e$ can be simultaneously performed in $O(BP_j + k_j)$ time for each SP. The number of starting points SP is less than $BP_j \cdot k_j + (BP_j + k_j + k_j \cdot BP_j)$, where the number of intersection points is $BP_j + k_j + k_j \cdot BP_j$. Since there are n stages, the time complexity of the algorithm is equal to $O\left(n(BP_{max}k_{max})^2\right)$. □

If we only need to find an optimal solution for $T \in [0, A]$, $T \in Z$, then the GrA can be modified as follows. If in the table for the function $F_j(T)$, $j = 1, 2, \ldots, n$, we have two columns with the intervals $[T_{K-1}, T_K)$ and $[T_K, T_{K+1})$, where $T_K \notin Z$, then we can change the intervals to $[T_{K-1}, \lfloor T_K \rfloor)$ and $[\lceil T_K \rceil, T_{K+1})$. If $t_s \in (\lfloor T_K \rfloor, \lceil T_K \rceil)$ at the next stage, then its equation is not taken into account. Thus, we have $BP_{max} \le A$ and obtain the following corollary. Denote this modification as GrA-I.

Corollary 1 *The GrA-I constructs an optimal solution for all points* $T \in [0, A]$, $T \in Z$, *in* $O\left(n(A \cdot k_{max})^2\right)$ *time, where* $k_{max} = \max_{j\in N} k_j$.

2.1 A Modification of the GrA-I with Time Complexity $O(nk_{max}A\log(k_{max}A))$

Steps $II.b - II.e$ can also be performed in another way. At each step II (if an intersection point is not considered), only two equations go out and two new equations go in. In step 1 (or at the first iteration of step II, but not I !) (when $SP = A$) of each stage j, $j = 2, \ldots, n$, we can sort all $BK_{j-1} + k_j$ first equations in non-decreasing order of the time points t, when they become leading equations. So, the first equation in this ordered list is the leading one. This ordering can be done in $O((BK_{j-1} + k_j)\log(BK_{j-1} + k_j))$ time. To find an intersection point ε_{min}, we only need to compare the first and second equations in this list. To add two new equations in a step of the GrA, we only need to perform $O(\log(BK_{j-1} + k_j))$ operations (to put them into the ordered list). Below we explain such a technique for Stage 3 of the numerical example considered.

In the same way, we can create a list of the lengths of the intervals which are used to compute LMI. In each step, the length of the list of the LMI values is less than or equal to k_j. So, to construct this list, we need to perform $O(k_j \log(k_j))$ operations. To recalculate it, we need $O(\log k_j)$ operations in each step. The first value in the list is LMI.

So, instead of $O(BK_j \cdot k_j)$ operations in each step $i > 1$, we need only to perform $O(\log(BK_{j-1} + k_j))$ operations and $O((BK_{j-1} + k_j)\log(BK_{j-1} + k_j))$ operations in the first step. Therefore, the time complexity of the modified GrA-I is $O(nk_{max}A\log(k_{max}A))$, if we only need to find a solution for any $T \in [0, A]$, $T \in Z$.

3 Numerical Example

In this section, the idea of the GrA is explained on the numerical example presented in Fig. 1, where $n = 4$.

Stage 1. **Determination of** $F_1(T)$.
It is obvious that $F_1(T) = f_1(t)$.

Stage 2. **Determination of** $F_2(T)$.

Step 1.

Let us analyze Fig. 2.1. To calculate the value $F_2(25)$ in the DPA, we can do the following. We draw $F_1(T)$ and $f_2(t)$ as shown in the figure, where $f_2(t)$ is drawn in a *reflected* way from the point t, i.e., the diagram of function $f_2'(t) = f_2(25-t)$ is drawn. Now, for each $t \in [0, 25]$, it is easy to calculate the values $f_2(t) + F_1(25-t)$. The diagram of this function is presented by a dotted line. We note that the diagram of the function $f_2(t) + F_1(25-t)$ is piecewise linear and continuous as well. So, its maximal values are reached at the break points, which correspond to the break points of the functions $F_1(T)$ and $f_2(t)$, i.e., $t_1, t_2, T_2, T_3, T_4, T_5$. We have the maximal value 10 at the point $t = 20$ which corresponds to the break point t_2.

To calculate the values $F_2(25-\varepsilon)$, we have to shift the diagram of the function $f_2(t)$ (and all break points t_1, t_2 which correspond to function $f_2(t)$) to the left by ε. It is easy to see that the value at the new break point t_2 will be $10 - u_1^4\varepsilon = 10$, if $\varepsilon \le t_2 - T_2 = 20 - 13 = 7$. Here u_1^4 is the slope of the function $F_1(T)$ on the interval $[T_2, A)$. At the same time, the values at the other break points will be changed as follows:

$$
\begin{aligned}
F'(T_5) &= F_1(T_5) - \varepsilon * u_2^2 = 2,\\
F'(T_4) &= F_1(T_4) - \varepsilon * u_2^2 = 2,\\
F'(T_3) &= F_1(T_3) - \varepsilon * u_2^2 = 9,\\
F'(T_2) &= F_1(T_2) - \varepsilon * u_2^2 = 10,\\
F'(t_1) &= F_1(t_1) - \varepsilon * u_1^4 = 8.
\end{aligned}
$$

Here $F'(T_x)$ denotes the equality corresponding to the point T_x if we shift the diagram of the function $f_2(t)$ to the left. So, on the interval $[25-7, 25] = [18, 25]$, we have $F_2(T) = 10 - u_1^4\varepsilon = 10$ (see Fig. 2.2).

Step 2.

At the point $T = 18$, the linear part of $F_1(T)$ corresponding to the break point t_2 changes. We have the equation: $F'(t_2) = F_1(t_2 = 18) - \varepsilon \cdot u_1^3 = 10 - \varepsilon \cdot \frac{1}{3}$ (see Fig. 2.3). The same change happens for the point T_2. For $T = 18 - \varepsilon$, we obtain the values $F'(T_2) = 10 - \varepsilon \cdot u_2^1 = 10 - \varepsilon\frac{2}{5}$. The slopes of all other functions $F'(T_3)$, $F'(T_4)$, $F'(T_5)$, $F'(t_1)$ do not change. So, the point t_2 remains the *leading point*.

Subsequently, we will use the following two terms: a *leading equation* and a *leading point*. Let us consider the two points x^1 and x^2 and their equations $B^{x_1} - U^{x_1}\varepsilon$ and $B^{x_2} - U^{x_2}\varepsilon$. If $B^{x_1} > B^{x_2}$ or ($B^{x_1} = B^{x_2}$ and $U^{x_2} > U^{x_1}$), then the point x_1 dominates the point x_2. For a leading point, there is no other point which dominates it.

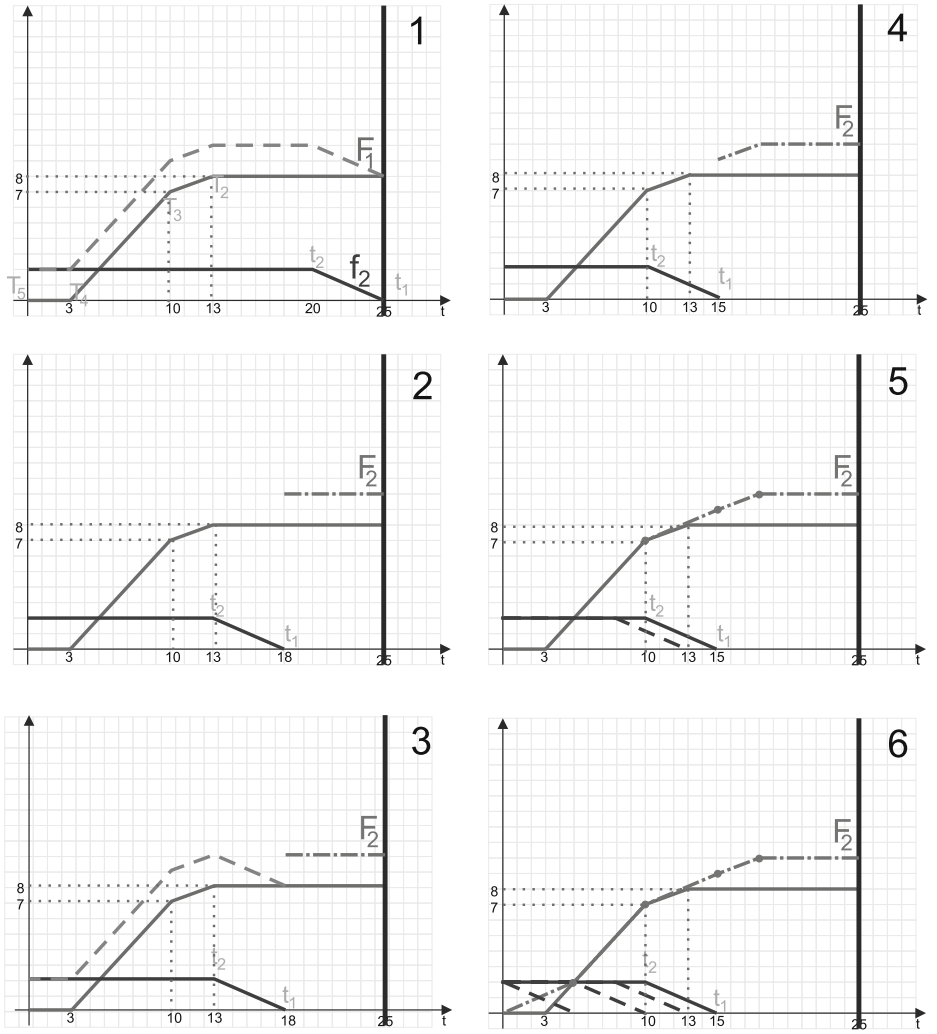

Fig. 2 Function $F_2(T)$

We continue with shifting the diagram of the function $f_2(t)$ to the left from the point $t = 18$. The values $F_2(18-\varepsilon)$ are calculated as $F_2(18-\varepsilon) = 10 - \varepsilon\frac{1}{3}$ corresponding to an equation for the point t_2. This holds till the next change of a slope which happens at the point $t = 15$, where t_2 and T_3 meet each other. So, on the interval $[15, 18]$, we have $F_2'(t) = 10 - (18-t)\frac{1}{3}$ (see Fig. 2.4).

Step 3.

From the point $T = 15$, we have the equation: $F'(t_2) = F(t_2) - \varepsilon \cdot u_1^2 = 10 - 3 \cdot \frac{1}{3} - \varepsilon \cdot 1 = 9 - \varepsilon$. The change of an equation happens for the point T_3 as well. For $T = 15 - \varepsilon$, we obtain the values $F'(T_3) = 9 - \varepsilon \cdot u_2^1 = 9 - \varepsilon\frac{2}{5}$. So, the point T_3 becomes a leading one, since its slope is less and it starts from the same value 9 as the point t_2.

Table 5 Function $F_2(T)$

K	1	2	3	4	5
interval K	[0, 5)	[5, 10)	[10, 15)	[15, 18)	[18, 25)
b^K	0	2	7	9	10
u^K	$\frac{2}{5}$	1	$\frac{2}{5}$	$\frac{1}{3}$	0

We continue with shifting the diagram of the function $f_2(t)$ to the left from the point $t = 15$. The values $F_2(15-\varepsilon)$ are calculated as $F_2(15-\varepsilon) = 9-\varepsilon\frac{2}{5}$ which corresponds to the equation for the point T_3. This holds till the next change of a slope (where the point t_1 and T_2 meet each other), namely at the point $t = 13$. So, on the interval [13, 15], we have $F_2(t) = 9-(15-t)\frac{2}{5}$ (see Fig. 2.5).

We continue our calculations in the same way. For each of the $k_2 + BP_1$ break points, where BP_1 is the number of break points of the function $F_1(T)$), we have an equation which characterizes how the value $F_2(T)$ will change if we shift the diagram of the function $f_2(t)$ to the left (i.e., those values of the function $F_2(T)$ we will have). Among these $k_2 + BP_1$ points, we choose the leading one and according to its equation we calculate $F_2(T)$. This calculation holds till the next point t, where a break point of $f_2(t)$ and a break point of $F_1(T)$ will meet each other.

In such a way, we obtain:

Step 4. For $t \in [10, 13]$: $F_2(t) = 8\frac{1}{5} - (13-t)\cdot\frac{2}{5}$ and the leading point T_3 the equation of which is $F'(T_3) = 8\frac{1}{5} - \varepsilon\cdot u_2^2 = 8\frac{1}{5} - \varepsilon\cdot\frac{2}{5}$.

Step 5. For $t \in [5, 10]$: $F_2(t) = 7-(10-t)\cdot 1$ and the leading point t_2 the equation of which is $F'(t_2) = 7-\varepsilon\cdot u_1^2 = 7-\varepsilon$.

Step 6. For $t \in [0, 5]$: $F_2(t) = 2-(5-t)\cdot\frac{2}{5}$ and the leading point T_4 the equation of which is $F'(t_2) = 2-\varepsilon\cdot u_2^2 = 2-\varepsilon\cdot\frac{2}{5}$.

The final diagram of the function $F_2(T)$ is presented in Fig. 2.6, and its linear parts are described in Table 5.

Stage 3. Determination of $F_3(T)$.

Next, we describe how the function $F_3(T)$ is calculated using the functions $F_2(T)$ and $f_3(t)$. All steps are performed in the same way as for stage 2. We present only their short description. In the following tables, a point (x, y) is described in the form $x|y$.

This data means the following. The point t_1 takes part in calculating the values $F_3(T)$ according to the equation $10-\varepsilon\cdot 0$. In the table, a leading point where only one equation influences $F_3(T)$ is marked by the symbol $*$. The length of the minimal interval (LMI) is obtained as $LMI = t_4 - T_2 = 1$, i.e., on this interval, the slopes of the equations do not change, which means that all the points hold their equations. So, on the interval $[25-LMI, 25] = [24, 25]$, the slope of the function $F_3(T)$ is equal to the slope of the equation for the point t_4, i.e., 0. In addition, we calculate $F_3(25) = 10+5 = 15$.

Step 1. Starting point $T = 25$, see Fig. 3.1.

t_1 : 0|10	t_4 : 0|15*	T_4 : 0|12
t_2 : 0|10	T_2 : 0|15	T_5 : 0|7
t_3 : 0|14	T_3 : 0|14	T_6 : 0|5

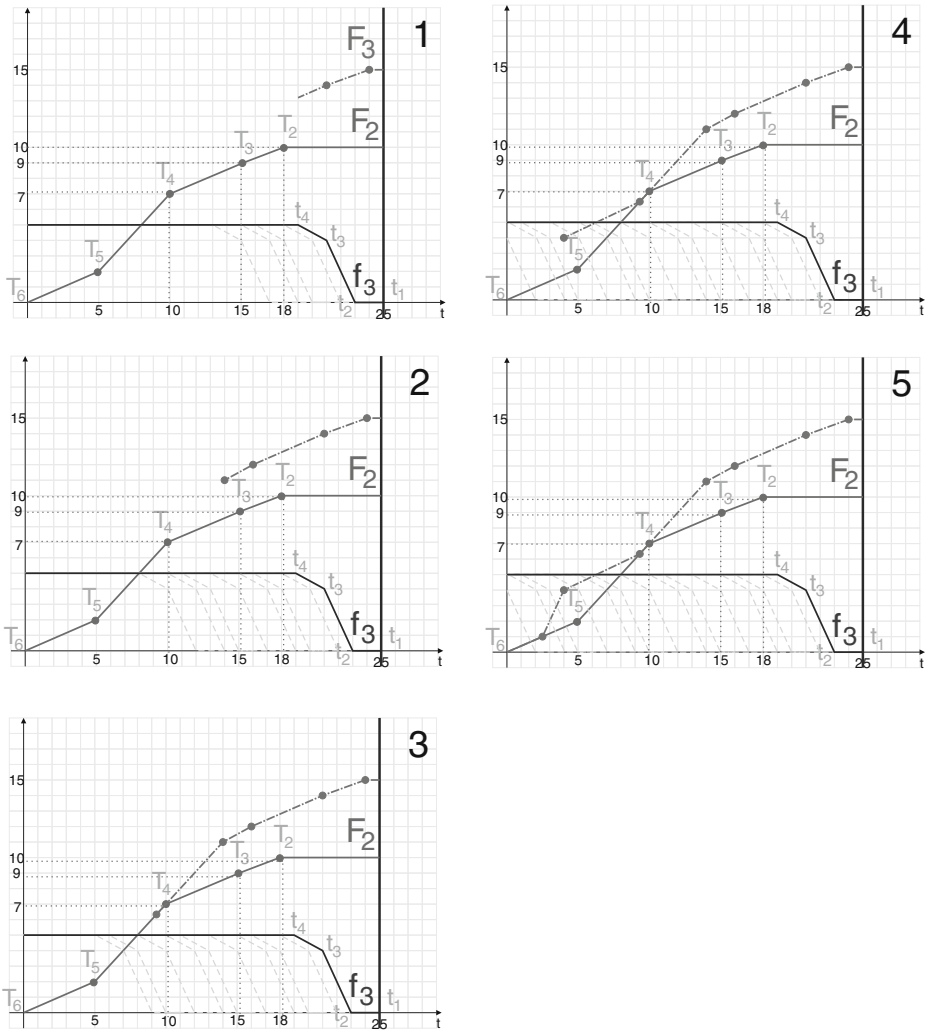

Fig. 3 Function $F_3(T)$

Step 2. Starting point $T = 24$, see Fig. 3.1.

–	$t_4: \ \frac{1}{3}\|15*$	–
–	$T_2: \ \frac{1}{2}\|15$	–
–	–	–

Next, we only present the calculations made in essential steps of the algorithm.

Step 7. Starting point $T = 16$, see Fig. 3.2.

–	$t_4: \ 1\|12$	$T_4: \ \frac{1}{2}\|12*$
$t_2 = \frac{2}{5}\|8\frac{3}{5}$		$T_5: \ 0\|7$
$t_3: \ \frac{2}{5}\|11\frac{4}{5}$	$T_3: \ 0\|9$	–

Table 6 Function $F_3(T)$

K	1	2	3	4	5	6	7	8
interval K	$\left[0, 2\frac{1}{2}\right)$	$\left[2\frac{1}{2}, 4\right)$	$\left[4, 9\frac{1}{3}\right)$	$\left[9\frac{1}{3}, 14\right)$	[14, 16)	[16, 21)	[21,24)	[21,25)
b^K	0	1	4	$6\frac{1}{3}$	11	12	14	15
u^K	$\frac{2}{5}$	2	$\frac{2}{5}$	1	$\frac{1}{2}$	$\frac{2}{5}$	$\frac{1}{3}$	0

We have $LMI = t_3 - T_4 = 12 - 10 = 2$. We check whether the equation for the point t_3 becomes leading on the interval [14, 16] (i.e., whether it overcomes the equation for T_4):

$$-\frac{1}{2}\varepsilon + 12 = -\frac{2}{5}\varepsilon + 11\frac{4}{5}; \qquad \frac{1}{5} = \frac{1}{10}\varepsilon;\ \varepsilon = 2.$$

This means that $LMI = 2 = \varepsilon$, and T_4 remains the leading point on the whole interval [14, 16]. On the interval [14, 16], the slope of the function $F_3(T)$ is equal to $\frac{1}{2}$.

Step 11. Starting point $T = 10$, see Fig. 3.3.

t_1 : 1|7	t_4 : $\frac{2}{5}\|6\frac{3}{5}$	T_4 is out of range
t_2 : 1|5		T_5 : $\frac{1}{2}\|6\frac{1}{2}$
t_3 : 1|7∗		–

We have $LMI = t_3 - T_5 = 1$. We check whether the equation for the point t_4 becomes leading on the interval [9, 10] (i.e., whether it overcomes the equation for t_3):

$$-\varepsilon + 7 = -\frac{2}{5}\varepsilon + 6\frac{3}{5}; \qquad \frac{2}{5} = \frac{3}{5}\varepsilon;\ \varepsilon = \frac{2}{3}.$$

On the interval $\left[9\frac{1}{3}, 10\right]$, the slope of the function $F_3(T)$ is equal to 1.

Step 17. Starting point $T = 4$, see Fig. 3.4.

t_1 : $\frac{2}{5}\|1\frac{3}{5}$	
t_2 : $\frac{2}{5}\|\frac{4}{5}$	
t_3 is out of range	T_6 : 2|4

We have $LMI = t_2 - T_6 = 2$. However, there is an intersection point of the equations of the points T_6 and t_1:

$$-2\varepsilon + 4 = -\frac{2}{5}\varepsilon + 1\frac{3}{5}; \qquad \frac{12}{5} = \frac{8}{5}\varepsilon;\ \varepsilon = \frac{3}{2}.$$

So, on the interval $\left[2\frac{1}{2}, 4\right]$, the slope of the function $F_3(T)$ is equal to 2.

Step 18. Starting point $T = 2\frac{1}{2}$, see Fig. 3.5.

In this step, the leading point is t_1. Thus, on the interval $\left[0, 2\frac{1}{2}\right]$, the slope of the function $F_3(T)$ is equal to $\frac{2}{5}$.

The final diagram of the function $F_3(T)$ is presented in Fig. 3.5, and its linear parts are described in Table 6.

Stage 4. Determination of $F_4(T)$.

Next, we describe how the function $F_4(T)$ is calculated using the functions $F_3(T)$ and $f_4(t)$. In the case of a step function $f_4(t)$, we can perform this stage in an easier way. We can construct the functions $\Phi_1(T) = F_3(T)$, $\Phi_2(T) = 1 + F_3(T - 3)$ and $\Phi_3(T) = 4 + F_3(T - 4)$. Then we have to construct the function $F_4(t) = \max\{\Phi_1(T), \Phi_2(T), \Phi_3(T)\}$. All these functions are presented in Fig. 4.

Of course, we can perform the same operations like at Stages 2 and 3, but this is more complicated and takes more time.

Backtracking

To find an optimal solution at the point $T = 25$, we can do backtracking. We have $\tau_4 = 4$ and $f_4(\tau_4) = 4$, $\tau_3 = 6$ and $f_3(\tau_3) = 5$, $\tau_2 = 5$ and $f_2(\tau_2) = 2$; $\tau_1 = 10$ and $f_1(\tau_1) = 7$. Moreover, $F^*(25) = 18$.

Let us analyze the running times of the graphical algorithm (GrA) and the DPA. In the DPA, we have to consider $(4 - 1) * \frac{25(25+1)}{2} + 25 = 1000$ points t.

In the first stage of the GrA, we consider 4 points of the function $f_1(t)$. At the second stage, we consider less than $4 * 2 = 8$ steps and recalculate up to $4 + 2 = 6$ equations in each step (practically, we have $6+6+6+5+4+3+2 = 32$). At the second stage, we consider less than the $5 * 4 = 20$ steps obtained from the break points and the two steps obtained due to the intersection of equations (practically, 17 steps) and recalculate up to $5 + 4 = 9$ equations in each step (practically, we have $9 + 9 + 9 + 8 + 8 + 8 + 3 * 7 + 4 * 6 + 5 + 4 + 3 + 2 = 119$). In the last step, we have to consider $3 * 8 = 24$ break points of the three functions

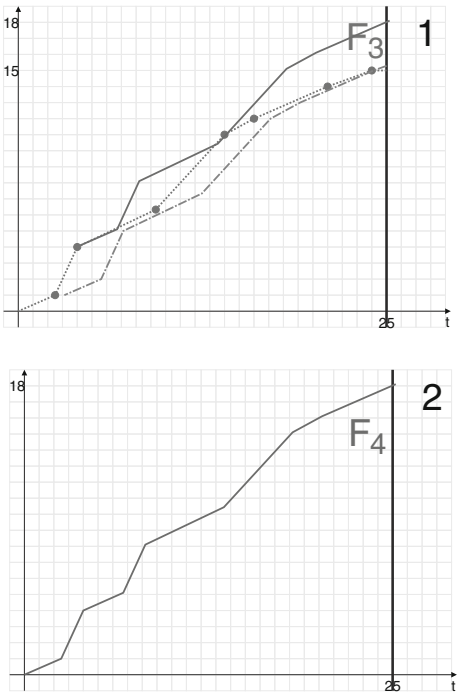

Fig. 4 Function $F_4(T)$

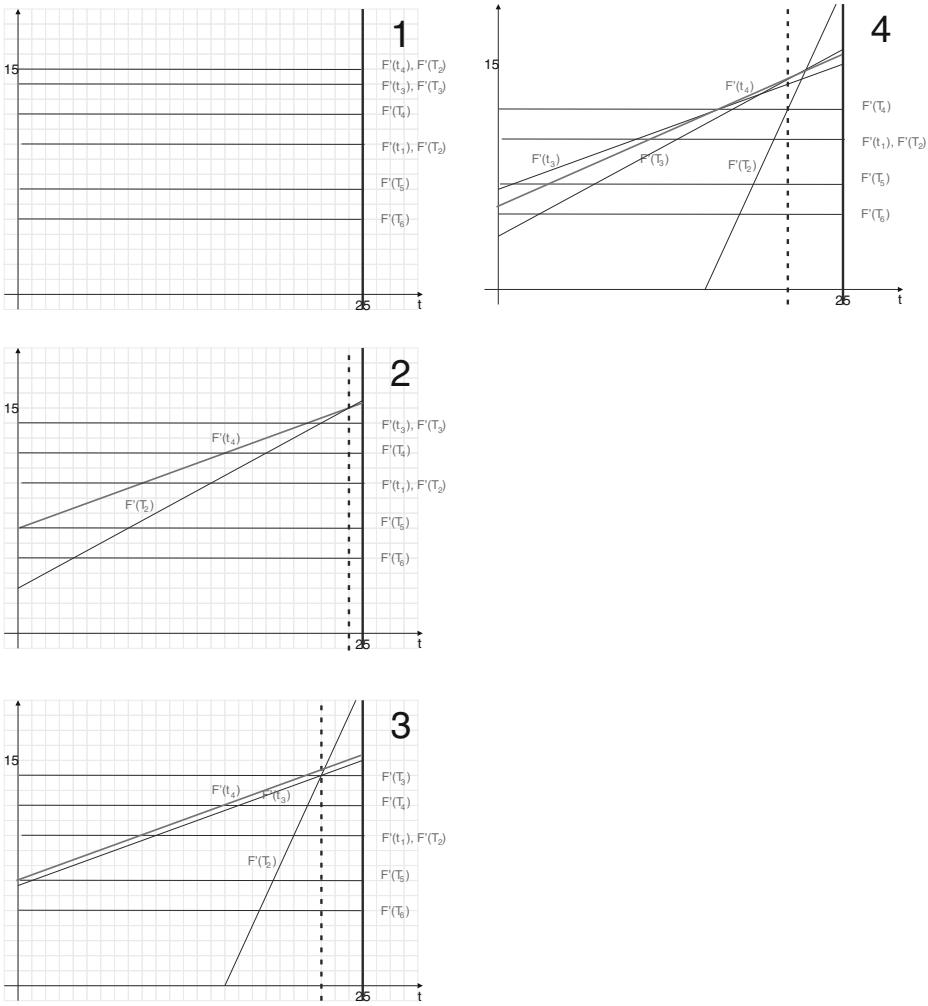

Fig. 5 List of equations in stage 3

$\Phi_1(T)$, $\Phi_2(T)$, $\Phi_3(T)$. Thus, in the GrA, we consider up to $4+32+119+24 = 179$ points or equations. If we scale our instance to a big number M (i.e., we multiply all input data by M), then the running time of the DPA increases M^2 times, but the running time of the GrA remains the same. Of course, for each equation, we have to do some simple calculations (operations). However, this number is constant: $O(1)$.

3.1 Stage 3 of the Numerical Example for the Modification of the GrA

In Fig. 5.1, the diagrams of $F'(t_1)$, $F'(t_2)$, $F'(t_3)$, $F'(t_4)$, $F'(T_1)$, $F'(T_2)$, $F'(T_3)$, $F'(T_4)$, $F'(T_5)$, $F'(T_6)$ are presented. So, the ordered list in step 1 of the stage is $(t_4, T_2, T_3, t_3, T_4, t_1, t_2, T_5, T_6)$.

In step 2, we delete the two equations corresponding to t_4 and T_2 from the list. To enter a new equation $F'(t_4)$, we do the following. Compare $F'(t_4)$ and $F'(t_1)$. Their intersection

point is (9, 10), i.e., $t = 9$, $F = 10$. Moreover, at $SP = 24$, the value of $F'(t_4)$ is larger. So, in the list, t_4 is before t_1. Then compare it with t_3. The intersection point is (21, 14) and the value of $F(t_4)$ is larger at SP. Finally, we compare it with T_3. We have t_4 before T_3 in the list. Analogously, we look for a position for T_2 in the list. The ordered list remains the same $(t_4, T_2, T_3, t_3, T_4, t_1, t_2, T_5, T_6)$. In Fig. 5.2, the diagrams of the corresponding equations in step 2 are shown.

To be exact, we have to save not the points t_x and T_x in the list but the pairs $(Interval_x, t_x)$, where $Interval_x$ is the interval where the equation t_x is leading. In step 3, we delete the two old equations corresponding to t_4 and T_3 from the list and insert two new equations corresponding to these points. We have the following order: $(t_4, T_3, t_3, T_2, T_4, t_1, t_2, T_5, T_6)$ (see Fig. 5.3). We note that T_2 dominates T_4 only before the point $t = 21$, i.e., on the interval [21, 22). This information has to be saved in the list.

Analogously, in step 4, we have the same order $(t_4, T_3, t_3, T_2, T_4, t_1, t_2, T_5, T_6)$. We remind that T_4 dominates T_2 for $t < 21$.

We have to mention as well that after deleting an equation t_x from the list, we have to reinsert (reorder) the equations of its left and right neighbors as well, since it has common intersection points (edges of the interval $Interval_x$). This means that in each step, we have to delete up to 6 equations from the list and to add 6 new equations.

4 Some Benefits of the GrA (GrA-I) in Comparison with the DPA

1. The time complexity of the modified GrA-I is $O(nk_{max}A\log(k_{max}A))$ in contrast to the time complexity $O\left(nA^2\right)$ of the DPA.
2. In spite of the fact that the complexity of the non-modified GrA-I is $O\left(n(A\cdot k_{max})^2\right)$ and the complexity of the DPA is $O\left(nA^2\right)$, we suppose that in practice, the running time of the GrA-I is substantially less. This conjecture is made based on our numerical results with graphical algorithms for other problems.
3. The running time of the GrA for two instances I_1 and I_2, where all parameters of I_2 are equal to the parameters of I_1 multiplied by $M > 1$, $M \in Z$, will be the same, although the running time of the DPA increases M^2 times. Thus, using the GrA, we can solve large scale instances or instances with real numbers in a more effective way.
4. If at stage j, $j = 1, 2, \ldots, n$, we have $k_j BP_j > A$, then we can recalculate the table for $F_j(T)$ from the table used in the DPA and continue with all other stages according to the DPA. This means that the running time of such a modification is always less than the running time of the DPA.
5. The GrA-I is more effective for some sub-cases. As we saw in Section 2 for a numerical instance, for some functions at stage 4, the complexity of the GrA-I will be less, if all functions $f_j(t)$ are step functions. Then the time complexity of the GrA-I is $O(nk_{max}A)$, which is substantially less than the complexity of the DPA considered.
6. As it is shown in Section 5, it is easier to construct an FPTAS based on a GrA.

5 An FPTAS Based on a GrA

In this section, a fully polynomial-time approximation scheme (FPTAS) is presented based on the corresponding GrA.

First, we recall some relevant definitions. For the optimization problem of minimizing a function $F(x)$, a polynomial-time algorithm that finds a feasible solution x' such that

$F(x')$ is at most $\rho \geq 1$ times less than the optimal value $F(x^*)$ 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, or an 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.

Let $LB = \max\limits_{j=1,\ldots,n} f_j(A)$ be a lower bound and $UB = n \cdot LB$ be an upper bound on the optimal objective function value for the problem.

The idea of the FPTAS is as follows. Let $\delta = \frac{\varepsilon LB}{n}$. To reduce the time complexity of the GrA, we have to diminish the number of columns BK_j considered, which is the number of different objective function values $0 = b^1, b^2, b^3, \ldots, b^{BL_k} = UB$. If we do not consider the original values b^k but the values $\overline{b^k}$ which are rounded up or down to the nearest multiple of δ values b^k, there are no more than $\frac{UB}{\delta} = \frac{n^2}{\varepsilon}$ different values $\overline{b^k}$. Then we will be able to convert the table for the function $F_j(T)$ into a similar table with no more than $2\frac{n^2}{\varepsilon}$ columns (see Fig. 6). Furthermore, for such a modified table (function) $F'(T)$, we will have $|F(T) - F'(T)| < \delta \leq \frac{\varepsilon F^*(A)}{n}$. If we do the rounding and modification after each step III of the GrA, then the cumulative error will be no more than $n\delta \leq \varepsilon F^*(A)$, and the total running time of the n runs of the GrA will be

$$O\left(\frac{n^3 k_{max}}{\varepsilon} \log\left(k_{max}\frac{n^2}{\varepsilon}\right)\right),$$

i.e., an FPTAS is obtained.

In [6], a technique was proposed to improve the complexity of an approximation algorithm for optimization problems. This technique can be described as follows. If there is an FPTAS for a problem with a running time bounded by a polynomial $P\left(L, \frac{1}{\varepsilon}, \frac{UB}{LB}\right)$, where L is the length of the problem instance and UB, LB are known upper and lower bounds,

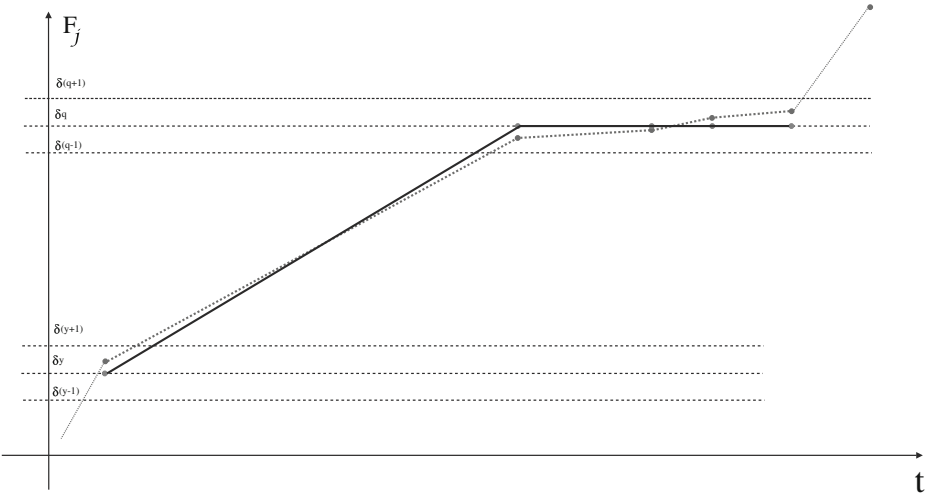

Fig. 6 Substitution of columns and modification of $F_j(T)$

and the value $\frac{UB}{LB}$ is not bounded by a constant, then the technique enables us to find in $P\left(L, \log\log\frac{UB}{LB}\right)$ 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 $P\left(L, \frac{1}{\varepsilon}\right)$, where P is the same polynomial. So, by using this technique, we can improve the FPTAS to have a running time of

$$O\left(\frac{n^2 k_{max}}{\varepsilon}\log\left(k_{max}\frac{n}{\varepsilon}\right)(1+\log\log n)\right),$$

We have to note that this time complexity is less than the time complexity of the FPTAS presented in [5] for a similar problem.

A detailed description of FPTASes based on a GrA for some single machine scheduling problems has been presented in [2].

6 A GrA for the Problem P-Cost

In this section, a GrA for the problem of finding lot-sizes and sequencing several products on a single imperfect machine is presented. For this problem, so-called P-cost and studied in [7], we have similar Bellman equations. Moreover, the problem P-Cost can be reduced to the project investment problem. Here, we consider a special case of problem P-Cost, for which the GrA improves the complexity of the DPA substantially.

The production line considered in this paper contains one facility (machine), which produces items of n different products in lots. A lot is a maximal set of items of the same product manufactured sequentially on the machine. The machine cannot process more than one item simultaneously. A sequence-dependent setup time is required between items of different products. The machine is imperfect in the following two senses: it can produce defective items which are not repairable, and it can break down. It is assumed that no item can be produced during a setup or breakdown and that setup and breakdown times cannot overlap. The following deterministic parameters are assumed to be given for each product j, $j = 1, \ldots, n$: b_j, $b_j > 0$, is the demand for good quality items; c_j, $c_j > 0$, is the per item cost of the unsatisfied demand; $t_j, t_j > 0$ is the processing time of an item; $s_{i,j}, s_{i,j} \geq 0$ is the setup time between items of products i and j; $f_j(x)$ is a non-decreasing integer-valued function representing the number of defective items, x is the total manufactured quantity of product j; $f_j(0) = 0$, and $f_j(x) < x$ for $x = 1, 2, \ldots, n$; $T(x_1, \ldots, x_n)$ is a non-negative non-decreasing real-valued function representing the cumulative machine breakdown time before the production of the last item, and x_j is the total number of manufactured items of product j.

The problem is to minimize the total cost

$$\sum_{j:=1}^{n} c_j \max\{0, b_j - (x_j - f_j(x_j))\},$$

of demand dissatisfaction subject to the constraint that the completion time C_{max} of the last item does not exceed a given upper bound A. As aforementioned, this problem was denoted as P-Cost.

In [7], this problem is solved in two steps. In the first step, a sequence of lots is computed by using an exact $B\&B$ algorithm for the Traveling Salesman Problem to minimize total

setup time. The time complexity of this algorithm is $O\left(n^2 2^n\right)$. Let TT be the total time given for production. Then $A = TT - TS^*$, where TS^* is the minimal total setup time.

In the second step, the function

$$F(x_1, \ldots, x_n) = \sum_{j:=1}^{n} c_j \max\{0, b_j - (x_j - f_j(x_j))\}$$

is minimized, where $\sum_{j:=1}^{n} x_j t_j \leq A$. It is obvious that this problem can be solved in $O\left(nA^2\right)$ time by a DPA with Bellman equations similar to the ones presented in Section 1.

In [7], the authors considered a special case of this problem, where $f_j(x_j) = \lfloor \alpha_j x_j \rfloor$, and they presented an FPTAS with a time complexity of

$$O\left(\frac{n^3}{\varepsilon^2} + n^3 \log\log\left(\sum_{j:=1}^{n} c_j b_j\right)\right).$$

According to the results of the numerical experiments provided in [8], this FPTAS works slower than an exact algorithm provided by CPLEX for a MIP formulation of the problem. However, we have to note the following. Since the traveling salesman problem (TSP) solved in the first step is many times more difficult, it seems to be necessary to do the following. First, using an effective algorithms we find UB_{TS} and LB_{TS}, which are an upper and a lower bound on the optimal objective function value for the TSP. Then we use the DPA to solve the problem P-Cost, where $A = TT - LB_{TS}$. As a result of the DPA, we will have optimal solutions *for all* $A' \in [0, A]$, in contrast to the fact that CPLEX provides only an optimal solution for $A' = A$. Then we return to the TSP and decide whether it makes sense to continue the $B\&B$ calculations, i.e., which result will be obtained for the problem P-Cost, if we improve the current value UB_{TS}. So, this approach seems to be more effective than a combination of the $B\&B$ algorithm for the TSP and CPLEX.

If we assume $f_j(x_j) = \alpha_j x_j$, then we can use the GrA to solve the problem, where $k_j = 2,\ j = 1, \ldots, n$. The analytical form $f_j(x_j) = \alpha_j x_j$ seems to be more adequate than $f_j(x_j) = \lfloor \alpha_j x_j \rfloor$ since, in practice, we deal with functions $f_j(x_j)$ statistically received.

Lemma 2 *The problem P-Cost with* $f_j(x_j) = \alpha_j x_j,\ j = 1, \ldots, n$, *is NP-hard.*

Proof We present a polynomial time reduction from the partition problem, which is as follows.

Partition problem A set $\overline{N} = \{a_1, a_2, \ldots, a_n\}$ of values $a_1 \geq a_2 \geq \cdots \geq a_n > 0$ with $a_i \in Z_+,\ i = 1, 2, \ldots, n$, and a value $\overline{A} \in Z$, $\overline{A} = \frac{1}{2}\sum_{j=1}^{n} a_j$ are given. Is there a subset $N' \subset \overline{N}$ such that $\sum_{j \in N'} a_j = \overline{A}$?

Each lot $j \in N$ corresponds to an item $j \in \overline{N}$. Moreover, we have $b_j = 1, \alpha_j = 0,\ c_j = t_j = a_j$ and $A = \overline{A}$. There exists an optimal solution of the problem P-Cost with $F^* = A$ if and only if there exists such a subset N' for the partition problem. □

Theorem 2 *The problem P-Cost with* $f_j(x_j) = \alpha_j x_j,\ j = 1, \ldots, n$, *can be solved by the GrA-I in* $O(nA \log A)$ *time.*

Theorem 3 *For the problem P-Cost with* $f_j(x_j) = \alpha_j x_j,\ j = 1, \ldots, n$, *there exists an FPTAS based on the GrA with a time complexity of*

$$O\left(\frac{n^2}{\varepsilon}\log(n/\varepsilon) + n^2 \log\log\left(\sum_{j:=1}^{n} c_j b_j\right)\right).$$

7 Concluding Remarks

The graphical approach can be applied to problems, where a pseudo-polynomial algorithm exists based on the Bellman equations. For the investment optimization and the P-Cost problems, the graphical algorithm improved the complexity of the corresponding pseudo-polynomial algorithm substantially. An FPTAS based on this GrA can be constructed with the best running time among the FPTASes known.

Acknowledgments This work was supported by the Russian Foundation of Basic Research (grant 13-01-12108), DAAD (Germany) and St-Etienne Metropolis government (France).

References

1. Lazarev, A.A., Werner, F.: A graphical realization of the dynamic programming method for solving NP-hard problems. Comput. Math. Appl. **58**(4), 619–631 (2009)
2. Gafarov, E.R., Dolgui, A., Werner, F.: Dynamic programming approach to design FPTAS for single machine scheduling problems. Research Report LIMOS UMR CNRS 6158 (2012)
3. Kellerer, H., Pferschy, U., Pisinger, D.: Knapsack Problems. Springer-Verlag, Berlin (2004)
4. Shaw, D.X., Wagelmans, A.P.M.: An algorithm for single-item capacitated economic lot sizing with piecewise linear production costs and general holding costs. Manag. Sci. **44**(6), 831–838 (1998)
5. Kameshwaran, S., Narahari, Y.: Nonconvex piecewise linear knapsack problems. Eur. J. Oper. Res. **192**, 56–68 (2009)
6. Chubanov, S., Kovalyov, M.Y., Pesch, E.: An FPTAS for a single-item capacitated economic lot-sizing problem with monotone cost structure. Math. Program. Ser. A **106**, 453–466 (2006)
7. Dolgui, A., Kovalyov, M.Y., Shchamialiova, K.: Multi-product lot-sizing and sequencing on a single imperfect machine. Comput. Optim. Appl. **50**, 465–482 (2011)
8. Schmeleva, K., Delorme, X., Dolgui, A., Grimaud, F., Kovalyov, M.Y.: Lot-sizing on a single machine, ILP models. Comput. Ind. Eng. **65**, 561–569 (2013)