A new effective dynamic program for an investment optimization problem

Evgeny A Gafarov, Alexandre Dolgui, Alexander A Lazarev, Frank Werner

A New Effective Dynamic Program for an Investment Optimization Problem

Evgeny R. Gafarov[a,b], Alexandre Dolgui[a], Alexander A. Lazarev[b,c,d,e,*], Frank Werner[f]

[a] *Ecole Nationale Superieure des Mines, LIMOS-UMR CNRS 6158, F-42023 Saint-Etienne, France*

[b] *V.A. Trapeznikov Institute of Control Sciences of the Russian Academy of Sciences, Profsoyuznaya st. 65, 117997 Moscow, Russian Federation*

[c] *Lomonosov Moscow State University, GSP-1, Leninskie Gory, 119991 Moscow, Russian Federation*

[d] *Moscow Institute of Physiscs and Technology, 9 Institutskiy per., Dolgoprudny, 141700 Moscow Region, Russian Federation*

[e] *International Laboratory of Decision Choice and Analysis, National Research University, Higher School of Economics, 20 Myasnsnitskaya street, 101000 Moscow Region, Russian Federation*

[f] *Fakultät für Mathematik, Otto-von-Guericke-Universität Magdeburg, PSF 4120, 39016 Magdeburg, Germany*

Abstract

After a series of publications of T.E. O'Neil et al. (e.g. in 2010), dynamic programming seems to be the most promising way to solve knapsack problems. Some techniques are known to make dynamic programming algorithms (DPA) faster. One of them is the graphical method that deals with piecewise linear Bellman functions. For some problems, it was previously shown that the graphical algorithm has a smaller running time in comparison with the classical DPA and also some other advantages. In this paper, an exact graphical algorithm (GrA) and a fully polynomial-time approximation scheme based on it are presented for an investment optimization problem having the best known running time. The algorithms are based on new Bellman functional equations and a new way of implementing the GrA.

Keywords: Multi-choice knapsack problem, Investment problem, Dynamic programming, FPTAS
MSC: 90B30, 90C39, 90-08.

*Corresponding author

Email addresses: axel73@mail.ru (Evgeny R. Gafarov), dolgui@emse.fr (Alexandre Dolgui), jobmath@mail.ru (Alexander A. Lazarev), frank.werner@ovgu.de (Frank Werner)

1. Introduction

The selection of projects is an important problem in science and engineering. In general, project portfolio selection problems are hard combinatorial problems, often formulated as knapsack or in a more general form as multi-dimensional multi-choice knapsack problems, possibly also with multiple objectives. They have a wide range of business applications in capital budgeting and production planning.

Different techniques have been used to solve the knapsack problem or its generalizations. Since the project data are often uncertain or imprecise, sometimes fuzzy techniques are used. For instance, in [1], a fuzzy multi-dimensional multiple-choice knapsack problem is formulated for the project selection problem, and then an efficient epsilon-constraint method and an multi-objective evolutionary algorithm are applied. In [2], a data envelope analysis, knapsack formulation and fuzzy set theory integrated model was suggested. In [3], the problem of selecting projects to be included in an R&D portfolio has also been formulated as a multi-dimensional knapsack problem. If partial funding and implementation is allowed, linear programming can be used and the sensitivity of the project selection decisions is examined. The selection among ranked projects under segmentation, policy and logical constraints was discussed in [4]. After ranking the projects by a multi-criteria approach, integer programming was applied to get a final solution satisfying the constraints. At the integer programming phase, a knapsack formulation was applied. Dynamic order acceptance and capacity planning on a single bottleneck resource has been considered in [5]. Stochastic dynamic programming was applied to determine a profit threshold for the accept/reject decision and to allocate a single bottleneck resource to the accepted projects with the objective to maximize expected revenue.

Often dynamic programming algorithms (DPA) are used for the exact solution and fully-polynomial time approximation schemes (FPTAS) based on these algorithms are derived for the approximate solution of such problems. In [6], a vector merging problem was considered in a dynamic programming context which can be incorporated into an FPTAS for the knapsack problem. Approximation algorithms for knapsack problems with cardinality constraints, where an upper bound is imposed on the number of items that can be selected, were given in [7]. Improved algorithms for this problem were presented in [8], where hybrid rounding techniques were applied. An efficient FPTAS for the multi-objective knapsack problem was given in [9]. It uses general techniques such as e.g. dominance relations in dynamic programming. Approximation algorithms for knapsack problems with sigmoid utilities were derived in [10]. The authors combined algorithms from discrete optimization with algorithms from continuous optimization. In [11], greedy algorithms for the knapsack problem were considered and improved approximation ratios for different variants of the problem were derived. Some applications of AI techniques to generation planning and investment were described in [12].

This paper deals with a project selection (or allocation) problem with the

single criterion of maximizing the total profit under one budget constraint, which is also a generalization of the knapsack problem. More precisely, a set $N = \{1, 2, \ldots, n\}$ of n potential projects and an investment budget (amount) $A > 0,\ A \in Z$, are given. For each project $j \in N$, a profit function $f_j(x),\ x \in [0, A]$, is also given, where the value $f_j(x')$ denotes the profit received if the amount x' is invested into the project j. The objective is to determine an amount $x_j \in [0, A],\ x_j \in Z$, for each project $j \in N$ such that $\sum_{j:=1}^{n} x_j \leq A$ and the total profit $\sum_{j:=1}^{n} f_j(x_j)$ is maximized. Closely related problems do not only exist in the area of project investment [13], but also in warehousing, economic lot sizing, etc. [14, 15].

In the following, we work with piecewise linear functions $f_j(x)$. The interval $[0, A]$ can be written as a union of intervals in the form

$$[0, A] = [t_j^0, t_j^1] \bigcup (t_j^1, t_j^2] \bigcup \ldots \bigcup (t_j^{k-1}, t_j^k] \bigcup \ldots \bigcup (t_j^{k_j - 1}, t_j^{k_j}]$$

such that the profit function has the form $f_j(x) = b_j^k + u_j^k(x - t_j^{k-1})$ for $x \in (t_j^{k-1}, t_j^k]$, where k is the number of the interval, b_j^k is the value of the function at the beginning of the interval, and u_j^k is the slope of the function. Without loss of generality, assume that $b_j^1 \leq b_j^2 \leq \ldots \leq b_j^{k_j},\ t_j^k \in Z,\ j \in N,\ k = 1, 2, \ldots, k_j$, and that $t_j^{k_j} = A,\ j = 1, 2, \ldots, n$.

A special case of the problem under consideration is similar to the well-known bounded knapsack problem:

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

for which a dynamic programming algorithm (DPA) of time complexity $O(nA)$ is known [16]. Dynamic programming algorithms and an FPTAS for the bounded set-up knapsack problem, which is a generalization of the bounded knapsack problem in which each item type has a set-up weight and a set-up value included into the objective function, were suggested in [17]. These algorithms can also be applied to the bounded knapsack problem, and one of the dynamic programming algorithms has the same time complexity as the best known algorithm for the bounded knapsack problem.

It is known [18] that all branch and bound ($B\&B$) algorithms with a lower and an upper bound calculated in polynomial time have an exponential running time close to $2^{O(x)}$ operations, unlike $P = NP$, where x is the input length. For example, for the one-dimensional knapsack problem, the time complexity is equal to or greater than $\frac{3}{2}\frac{2^{n+3/2}}{\sqrt{\pi(n+1)}}$, where n is the number of items. This means that $B\&B$ algorithms are not far away from a complete enumeration.

In a series of publications by T.E. O'Neil et al. [19], it was shown that some of the well-known NP-hard problems can be solved by dynamic programming in sub-exponential time, i.e., in $2^{O(\sqrt{x})}$ operations. So, at the moment, dynamic

programming seems to be the most promising way to solve knapsack problems. Some techniques are known to make dynamic programming faster, e.g., in [20, 21, 22], functional equations and techniques were considered that are different from the ones considered in this paper.

The following problem is also similar to the problem under consideration:

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

where $f_j(x_j)$ are piecewise linear as well. For this problem, a DPA with a running time of $O(\sum k_j A)$ [23] and an FPTAS with a running time of $O((\sum k_j)^3/\varepsilon)$ [24] are known.

In this paper, we present an alternative DPA based on a so-called graphical approach. This algorithm has a running time of $O(\sum k_j \min\{A, F^*\})$, where F^* is the optimal objective function value. Thus, it outperforms an algorithm from [25] which has a worse running time close to $O(nk_{max}Alog(k_{max}A))$, where $k_{max} = \max_{j-1,\ldots,n} k_j$. The second contribution of this paper is an FPTAS derived by a scaling argument from this new DPA. Note that an FPTAS was already proposed for the treated problem in [25], but the new FPTAS has an improved running time of $O(\sum k_j n \log\log n/\varepsilon)$.

While the running time of a similar DPA from [23] is proportional to the sum of all linear profit pieces times the budget A, the new algorithm replaces A by the largest profit of a single project. The main idea is as follows. Instead of evaluating the dynamic programming functions for every budget value $t = 1, \ldots, A$, we keep the profit functions (depending on the budget value) as piecewise linear functions. These functions can be represented by a collection of linear pieces, instead of a full table of the values. Since all relevant data are integer and the profit functions must be non-decreasing in the budget value, one can easily bound the number of relevant linear pieces to obtain the improved complexity.

The remainder of the paper is as follows. In Section 2, we present the Bellman equations to solve the problem under consideration. In Section 3, an exact graphical algorithm (GrA) based on an idea from [26] is presented. In Section 4, an FPTAS based on this GrA is derived.

2. Dynamic programming algorithm

In this section, we present a DPA for the problem considered. For any project j and any state $t \in [0, A]$, we define $F_j(t)$ as the maximal profit incurred for the projects $1, 2, \ldots, j$, when the remaining budget available for the projects $j+1, j+2, \ldots, n$ is equal to t. Thus, we have:

$$\begin{array}{ll} F_j(t) = \max & \sum_{h:=1}^{j} f_h(x_h) \\ s.t. & \sum_{h:=1}^{j} x_h \leq A - t, \\ & x_h \geq 0,\ x_h \in Z,\ h = 1, 2, \ldots, j. \end{array} \tag{3}$$

We define $F_j(t) = 0$ for $t \notin [0, A]$, $j = 1, 2, \ldots, n$ and $F_0(t) = 0$ for any t. Then we have the following recursive equations:

$$\begin{aligned} F_j(t) &= \max_{x \in [0, A-t]} \{f_j(x) + F_{j-1}(t+x)\} \\ &= \max_{1 \le k \le k_j} \max_{x \in (t_j^{k-1}, t_j^k] \bigcap [0, A-t]} \{b_j^k - u_j^k t_j^{k-1} + u_j^k \cdot x + F_{j-1}(t+x)\}, \\ & \quad j = 1, 2, \ldots, n. \end{aligned} \tag{4}$$

Lemma 1. *All functions $F_j(t)$, $j = 1, 2, \ldots, n$, are non-increasing on the interval $[0, A]$.*

The proof of this lemma immediately follows from definition (3) of the functions $F_j(t)$.

The DPA based on the equations (4) can be organized as follows. For each stage $j = 1, \ldots, n$, for the state $t = 0$ we compute no more than $k_j A$ values $v_k^x = \{b_j^k - u_j^k t_j^{k-1} + u_j^k \cdot x + F_{j-1}(t+x)\}$, $1 \le k \le k_j$, $x \in (t_j^{k-1}, t_j^k]$ and put them into the corresponding lists L_k. If we have for the next value $v_k^x \le v_k^{x-1}$, we do not put it into the corresponding list. This means that the values in each list L_k are ordered in in non-decreasing order. For the next state $t = 1$, we only need to exclude the last element from the considered L_k, $k = 1, \ldots, n$, if it corresponds to an x which is not in the interval $(t_j^{k-1}, t_j^k] \bigcap [0, A - t]$ and compare the new k_j last elements of the lists. If we continue in the same way for $t = 2, \ldots, A$, we can calculate $F_j(t)$, $t = 1, 2, \ldots, A$, in $O(k_j A)$ time. As a consequence the running time of the DPA using such a type of Bellman equations is $O(\sum k_j A)$. A similar idea was presented in [23].

The algorithms presented in [25] for the problem under consideration are based on the functional equations (3) and another technique to implement the graphical method. In contrast, the GrA presented in this paper is based on the equations (4).

3. Graphical algorithm

In this section, we develop an algorithm which constructs the functions $F_j(t)$, $j = 1, 2, \ldots, n$, in a more effective way by using the idea of graphical approach proposed in [26]. We will use the name DPA for the algorithm presented in Section 2 and GrA for this new algorithm.

The underlying general idea of improving the DPA for piecewise linear functions is as follows: Instead of going through all capacity values up to A, we keep the Bellmans functions depending on the capacity. They are again piecewise linear functions. Keeping these function pieces well organized for all capacities involves a considerable amount of technicalities and allows for running time improvements if the data representation is done in a clever way.

Below we prove that the functions $F_j(t)$, $j = 1, 2, \ldots, n$, constructed in the GrA are piecewise linear. Any piecewise linear function $\varphi(x)$ can be defined by

three sets of numbers: a set of break points I (at each break point, a new linear segment of the piecewise linear function ends), a set of slopes U and a set of values of the function at the beginning of the intervals B. Let $I[k]$ denote the k-th element of the ordered set I. The same notations will be used for the sets U and B as well. The notation $\varphi.I[k]$ denotes the k-th element of the set I of the function $\varphi(x)$. Then, for example, for $x \in (t_j^{k-1}, t_j^k] = (f_j.I[k-1], f_j.I[k]]$, we have

$$f_j(x) = f_j.B[k] + f_j.U[k](x - f_j.I[k]).$$

Note that $\varphi.I[k] < \varphi.I[k+1],\ k = 1, 2, \ldots, |\varphi.I| - 1$ and $k_j = |f_j.I|$. In each step $j,\ j = 1, 2, \ldots, n$, of the subsequent algorithm, temporary piecewise linear functions Ψ_j^i and Φ_j^i are constructed. These functions are used to define functions $F_j(t),\ j = 1, 2, \ldots, n$, The functions $F_j(t)$ are piecewise linear as well. For $t \in Z$, their values are equal to the values of the functions $F_j(t)$ in the DPA.

Let $\varphi.I[0] = 0$ and $\varphi.I[|\varphi.I| + 1] = A$. The points $t \in \varphi.I$ and the other end points of the intervals with the piecewise linear functions considered in this article will be called *break points*. To *construct a function* in the GrA means to compute their sets I, U and B. Then the GrA is as follows. At each stage $j = 1, \ldots, n$, we compute the temporary functions $\Psi_j^k(t)$ and $\Phi_j^k(t)$ which are used to compute $F_j(t)$. The key features of the algorithm consist in the computation of $\Phi_j^k(t)$, modifying a piecewise linear function $F_{j-1}(t)$ by changing linear fragments.

Graphical algorithm

1. Let $F_0(t) = 0$, i.e., $F_0.I := \{A\},\ F_0.U := \{0\},\ F_0.B := \{0\}$;
2. FOR $j := 1$ TO n DO
   - 2.1. FOR $k := 1$ TO k_j DO
     - 2.1.1. Construct the temporary function

       $$\Psi_j^k(t) = f_j.B[k] - f_j.U[k] \cdot f_j.I[k-1] + f_j.U[k] \cdot t + F_{j-1}(t)$$

       according to Procedure 2.1.1.;
     - 2.1.2. Construct the temporary function

       $$\Phi_j^k(t) = \max_{x \in (f_j.I[k-1], f_j.I[k]] \bigcap [0, A-t]} \{\Psi_j^k(t+x) - f_j.U[k] \cdot t\}$$

       according to Procedure 2.1.2.;
     - 2.1.3. IF $k = 1$ THEN $F_j(t) := \Phi_j^k(t)$ ELSE $F_j(t) := \max\{F_j(t), \Phi_j^k(t)\}$.
   - 2.2. Modify the sets I, U, B of the function $F_j(t)$ according to Procedure 2.2.
3. The optimal objective function value is equal to $F_n(0)$.

The above algorithm uses Procedures 2.1.1. and 2.1.2. described below. In Procedure 2.1.1., we shift the function $F_{j-1}(t)$ up by the value $f_j.B[k] - f_j.U[k] \cdot f_j.I[k-1]$ and increase all slopes in its diagram by $f_j.U[k]$. If all values $t \in F_{j-1}.I$ are integer, then all values from the set $\Psi_j^i.I$ are integer as well. It is obvious that Procedure 2.1.1. can be performed in $O(|F_{j-1}.I|)$ time.

Procedure 2.1.1.

Given are k and j;

$\Psi_j^k.I = \emptyset$, $\Psi_j^k.U = \emptyset$ and $\Psi_j^k.B = \emptyset$.

FOR $i := 1$ TO $|F_{j-1}.I|$ DO

add the value $F_{j-1}.I[i]$ to the set $\Psi_j^k.I$;

add the value

$$f_j.B[k] - f_j.U[k] \cdot f_j.I[k-1] + f_j.U[k] \cdot F_{j-1}.I[i] + F_{j-1}.B[i]$$

to the set $\Psi_j^k.B$;

add the value $f_j.U[k] + F_{j-1}.U[i]$ to the set $\Psi_j^k.U$;

Before describing Procedure 2.1.2., we present Procedure FindMax in which the maximum function $\varphi(t)$ of two linear fragments $\varphi_1(t)$ and $\varphi_2(t)$ is constructed.

Procedure FindMax

1. Given are the functions $\varphi_1(t) = b_1 + u_1 \cdot t$ and $\varphi_2(t) = b_2 + u_2 \cdot t$ and an interval $(t', t'']$. Let $u_1 \leq u_2$;
2. IF $t'' - t' \leq 1$ THEN RETURN $\varphi(t) = \max\{\varphi_1(t''), \varphi_2(t'')\} + 0 \cdot t$ defined on the interval $(t', t'']$;
3. Find the intersection point t^* of $\varphi_1(t)$ and $\varphi_2(t)$;
4. IF t^* does not exist OR $t^* \notin (t', t'']$ THEN

   IF $b_1 + u_1 \cdot t' > b_2 + u_2 \cdot t'$ THEN RETURN $\varphi(t) = \varphi_1(t)$ defined on the interval $(t', t'']$;

   ELSE RETURN $\varphi(t) = \varphi_2(t)$ defined on the interval $(t', t'']$;
5. ELSE

   IF $t^* \in Z$ THEN

   $\varphi(t) := \varphi_1(t)$ on the interval $(t', t^*]$;

   $\varphi(t) := \varphi_2(t)$ on the interval $(t^*, t'']$;

   RETURN $\varphi(t)$;

   ELSE IF $t^* \notin Z$ THEN

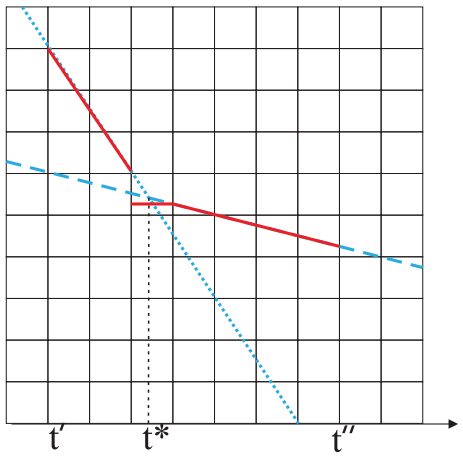

Figure 1: Procedure FindMax. Cutting of a non-integer point

$\varphi(t) := \varphi_1(t)$ on the interval $(t', \lfloor t^* \rfloor]$;
$\varphi(t) := b_2 + u_2 \cdot \lfloor t^* \rfloor + 0 \cdot t$ on the interval $(\lfloor t^* \rfloor - 1, \lfloor t^* \rfloor]$;
$\varphi(t) := \varphi_2(t)$ on the interval $(\lfloor t^* \rfloor, t'']$;
RETURN $\varphi(t)$;

The case when $t^* \notin Z$ is presented in Fig. 1. So, if both points t' and t'' are integer, then $\varphi.I$ contains only integer break points t. The running time of Procedure FindMax is constant.

In the subsequent Procedure 2.1.2., we do the following. When we shift s' to the right, we shift the interval $I' = [t_{left}, t_{right}]$ of the length $f_j.I[k] - f_j.I[k-1]$. We have to use the values $\Psi_j^k(x)$ for $x \in T'$ to calculate $\Phi_j^k(t)$ at the point $t = s'$. Since $\Psi_j^k(x)$ is piecewise linear, it is only necessary to consider the values $\Psi_j^k(x)$ at the break points belonging to T' and at the end points of the interval T'. So, if we shift s' to the right by a small value $x \in [0, \varepsilon]$ such that all the break points remains the same, then the value $\Phi_j^k(t)$ will be changed according to the value $\varphi_{max}(x)$.

Procedure 2.1.2.

2.1.2.1. Given are k, j and $\Psi_j^k(t)$;

2.1.2.2. $\Phi_j^k.I := \emptyset$, $\Phi_j^k.U := \emptyset$ and $\Phi_j^k.B := \emptyset$;

2.1.2.3. $s' := 0$, $t_{left} := s' + f_j.I[k-1]$, $t_{right} := \min\{s' + f_j.I[k], A\}$;

2.1.2.4. Let $T' = \{\Psi_j^k.I[v], \Psi_j^k.I[v+1], \ldots, \Psi_j^k.I[w]\}$ be the maximal subset of $\Psi_j^k.I$, where $t_{left} < \Psi_j^k.I[v] < \ldots < \Psi_j^k.I[w] < t_{right}$,

Let $T := \{t_{left}\} \bigcup T' \bigcup \{t_{right}\}$;

2.1.2.5. WHILE $s' \leq A$ DO

2.1.2.6. IF $T' = \emptyset$ THEN let
$w+1 = argmax_{i=1,2,\ldots,|\Psi_j^k.I|}\{\Psi_j^k.I[i] | \Psi_j^k.I[i] > t_{right}\}$
and $v = argmin_{i=1,2,\ldots,|\Psi_j^k.I|}\{\Psi_j^k.I[i] | \Psi_j^k.I[i] > t_{left}\}$;

2.1.2.7. IF $w+1$ is not defined THEN let $w+1 = |\Psi_j^k.I|$;

2.1.2.8. IF v is not defined THEN let $v = |\Psi_j^k.I|$;

2.1.2.9. IF $t_{left} < A$ THEN $\varepsilon_{left} := \Psi_j^k.I[v] - t_{left}$ ELSE $\varepsilon_{left} := A - s'$;

2.1.2.10. IF $t_{right} < A$ THEN $\varepsilon_{right} := \Psi_j^k.I[w+1] - t_{right}$ ELSE $\varepsilon_{right} := +\infty$;

2.1.2.11. $\varepsilon := \min\{\varepsilon_{left}, \varepsilon_{right}\}$;

2.1.2.12. IF $t_{left} < A$ THEN

$$b_{left} := \Psi_j^k.B[v] + \Psi_j^k.U[v] \cdot (t_{left} - \Psi_j^k.I[v-1]) - f_j.U[k] \cdot s'$$

ELSE $b_{left} := 0$;

2.1.2.13. IF $t_{right} < A$ THEN

$$b_{right} := \Psi_j^k.B[w+1] + \Psi_j^k.U[w+1] \cdot (t_{right} - \Psi_j^k.I[w]) - f_j.U[k] \cdot s'$$

ELSE $b_{right} := 0$;

2.1.2.14. IF $T' = \emptyset$ THEN $b_{inner} := 0$ ELSE

$$b_{inner} := \max_{s=v,v+1,\ldots,w}\{\Psi_j^k.B[s] + \Psi_j^k.U[s] \cdot (\Psi_j^k.I[s] - \Psi_j^k.I[s-1])\} - f_j.U[k] \cdot s';$$

2.1.2.15. Denote function

$$\varphi_{left}(x) := b_{left} - (f_j.U[k] - \Psi_j^k.U[v]) \cdot x.$$

IF $t_{left} = A$ THEN $\varphi_{left}(x) := 0$;

2.1.2.16. Denote function

$$\varphi_{right}(x) := b_{right} - (f_j.U[k] - \Psi_j^k.U[w+1]) \cdot x.$$

IF $t_{right} = A$ THEN $\varphi_{right}(x) := 0$;

2.1.2.17. Denote function

$$\varphi_{inner}(x) := b_{inner} - f_j.U[k] \cdot x.$$

IF $T' = \emptyset$ THEN $\varphi_{inner}(x) := 0$;

2.1.2.18. Construct the piecewise linear function

$$\varphi_{max}(x) := \max_{x \in [0,\varepsilon]}\{\varphi_{left}(x), \varphi_{right}(x), \varphi_{inner}(x)\}$$

according to Procedure *FindMax*;

2.1.2.19. add the values from $\varphi_{max}.I$ increased by s' to the set $\Phi_j^k.I$;

2.1.2.20 add the values from $\varphi_{max}.B$ to the set $\Phi_j^k.B$;

2.1.2.21. add the values from $\varphi_{max}.U$ to the set $\Phi_j^k.U$;

2.1.2.22. IF $\varepsilon = \varepsilon_{left}$ THEN exclude $\Psi_j^k.I[v]$ from the set T and $v := v+1$;

2.1.2.23. IF $\varepsilon = \varepsilon_{right}$ THEN include $\Psi_j^k.I[w+1]$ to the set T and $w := w+1$;

2.1.2.24. $s' := s' + \varepsilon$.

2.1.2.25. $t_{left} := s' + f_j.I[k-1]$, $t_{right} := \min\{s' + f_j.I[k], A\}$, recompute T' (for details, see the proof of Lemma 2);

2.1.2.26. Modify the function Φ_j^k according to Procedure 2.2. (described below).

Next, we present Procedure 2.2. used in step [2.1.26] of Procedure 2.1.2. In Procedure 2.2., we combine two adjoining linear fragments that are in the same line. That means that, if we have two adjacent linear fragments which are described by the values (slopes) $F_j.U[k]$, $F_j.U[k+1]$ and $F_j.B[k], F_j.B[k+1]$, where $F_j.U[k] \cdot (F_j.U[k] - F_j.U[k-1]) + F_j.B[k] = F_j.B[k+1]$, (i.e., these fragments are on the same line), then, to reduce the number of intervals $|F_j.I|$ and thus the running time of the algorithm, we can join these two intervals into one interval.

Procedure 2.2.
Given is $F_j(t)$;
FOR $k := 1$ TO $|F_j.I| - 1$ DO

IF $F_j.U[k] = F_j.U[k+1]$ AND $F_j.U[k] \cdot (F_j.U[k] - F_j.U[k-1]) + F_j.B[k] = F_j.B[k+1]$ THEN

$F_j.B[k+1] := F_j.B[k]$;

Delete the kth elements from $F_j.B$, $F_j.U$ and $F_j.I$;

Lemma 2. *Procedure 2.1.2. has a running time of* $O(|F_{j-1}.I|)$.

Proof. Step [2.1.2.14] and the re-computation of T' in step [2.1.2.25] have to be performed with the use of a simple data structure. Let $\{q_1, q_2, \ldots, q_r\}$ be a maximal subset of T' having the following properties:

$q_1 < q_2 < \ldots < q_r$;

there is no $q \in T'$ such that $q_i \leq q < q_{i+1}$ and
$\Psi_j^k.B[q] + \Psi_j^k.U[q] \cdot (\Psi_j^k.I[q] - \Psi_j^k.I[q-1]) \geq \Psi_j^k.B[q_{i+1}] + \Psi_j^k.U[q_{i+1}] \cdot (\Psi_j^k.I[q_{i+1}] - \Psi_j^k.I[q_{i+1}-1])$,
$i = 1, \ldots, r-1$.

We can keep track of the set $\{q_1, q_2, \ldots, q_r\}$ by storing its elements in increasing order in a Queue Stack, i.e., a list with the property that elements at the beginning can only be deleted while at the end, elements can be deleted and added [27]. This data structure can easily be implemented such that each deletion and each addition requires a constant time. So, steps [2.1.2.14] and [2.1.2.25] can be performed in constant time.

Each of the steps [2.1.2.6]–[2.1.2.25] can be performed in constant time. The loop [2.1.2.5] can be performed in $O(|\Psi_j^k.I|)$ time, where $|\Psi_j^k.I| = |F_{j-1}(t).I|$, since each time a break point from $|\Psi_j^k.I|$ is added or deleted. So, the lemma is true.

We remind that in the DPA, the functional equations (4) are considered. In fact, in Procedure 2.1.1., we construct the function

$$b_j^k - u_j^k t_j^{k-1} + u_j^k \cdot (t + x) + F_{j-1}(t + x)$$

and in Procedure 2.1.2., we construct the function

$$\Phi_j^k(t) = \max_{x \in (t_j^{k-1}, t_j^k] \bigcap [0, A-t]} \{b_j^k - u_j^k t_j^{k-1} + u_j^k \cdot (t + x) - u_j^k \cdot t + F_{j-1}(t + x)\}.$$

Unlike the DPA, to construct $\Phi_j^k(t)$ in the GrA, we do not consider all integer points $x \in (t_j^{k-1}, t_j^k] \bigcap [0, A-t]$, but only the break points from the interval, since only they influence the values of $\Phi_j^k(t)$ (and in addition t_{left}, t_{right}). Step [2.1.3] can be performed according to Procedure FindMax as well, i.e., to construct the function $F_j(t) := \max\{F_j(t), \Phi_j^i(t)\}$, their linear fragments have to be compared in each interval, organized by their break points. It is easy to see that we do the same operations with the integer points t as in the DPA. So, the values $F_j(t),\ t \in Z$, are the same for the GrA and the DPA, and we can state the following:

Lemma 3. *The values $F_j(t),\ j = 1, 2, \ldots, n$, at the points $t \in [0, A] \bigcap Z$ are equal to the values of the functions $F_j(t)$ considered in the DPA.*

Lemma 4. *All functions $F_j(t),\ j = 1, 2, \ldots, n$, are piecewise linear on the interval $[0, A]$ with integer break points.*

Proof. For $F_0(t)$, the lemma is true. In Procedure 2.1.1., all break points from the set $\Psi_1^i.I$ are integer as well (see the comments after Procedure 2.1.1.). Since all points from $f_1.I$ are integer, we have $\varepsilon \in Z$ and as a consequence, $s' \in Z$. According to the Procedure FindMax, all points $\varphi_{max}.I$ considered in Procedure 2.1.2. are integer. So, all break points from $\Phi_j^i.I,\ i = 1, 2, \ldots, k_j$, are integer as well. Thus, the break points of the function $F_1(t) := \max\{F_1(t), \Phi_1^i(t)\}$ are integer, if we use Procedure FindMax to compute the function $\max\{F_1(t), \Phi_1^i(t)\}$. Analogously, we can prove that all break points of $F_2(t)$ are integer, etc.

It is obvious that all functions $F_j(t),\ j = 1, 2, \ldots, n$, constructed in the GrA are piecewise linear. Thus, the lemma is true.

$f_1.I = \{3, 10, 13, 25\}$	$f_2.I = \{5, 25\}$	$f_3.I = \{2, 4, 6, 25\}$	$f_4.I = \{3, 4, 25\}$
$f_1.U = \{0, 1, \frac{1}{3}, 0\}$	$f_2.U = \{\frac{2}{5}, 0\}$	$f_3.U = \{0, 2, \frac{1}{2}, 0\}$	$f_4.U = \{0, 0, 0\}$
$f_1.B = \{0, 0, 7, 8\}$	$f_2.B = \{0, 2\}$	$f_3.B = \{0, 0, 4, 5\}$	$f_4.I = \{0, 1, 4\}$

Table 1: Functions $f_j(t)$

Theorem 1. *The GrA finds an optimal solution of the problem in*

$$O\left(\sum k_j \min\left\{A, \max_{j=1,2,\ldots,n} \{|F_j.B|\}\right\}\right)$$

time.

Proof. Analogously to the proof of Lemma 4, after each step [2.1.3] of the GrA, the function $F_j(t),\ j = 1, 2, \ldots, n$, has only integer break points from the interval $[0, A]$. Each function $\Phi_j^i.I,\ j = 1, 2, \ldots, n, i = 1, 2, \ldots, k_j$, has only integer break points from $[0, A]$ as well. So, to perform step [2.1.3], we need to perform Procedure FindMax on no more than $A + 1$ intervals. Thus, the running time of step [2.1.3] is $O(A)$. According to Lemmas 1 and 2, the running time of steps [2.1.1] and [2.1.2] is $O(F_j.I)$, where $F_j.I \leq A$. The running time of step [2.2.] is $O(F_j.I)$ as well.

Analogously to Section 2, it is easy to show that $F_j(t),\ j = 1, 2, \ldots, n$, is a non-increasing function in t. Thus,

$$F_j.B[k] \geq F_j.B[k+1],\ j = 1, 2, \ldots, n,\ k = 1, 2, \ldots, |F_j.I| - 1.$$

Then, according to Procedure 2.2., there are no more than $2 \cdot |F_j.B|$ different values in the set $F_j.I$.

Thus, the running time of the GrA is

$$O\left(\sum k_j \min\left\{A, \max_{j=1,2,\ldots,n} \{|F_j.B|\}\right\}\right).$$

In fact, the running time is less than $O(\sum k_j \min\{A, F^*\})$, where F^* is the optimal objective function value, since $\max_{j=1,2,\ldots,n} |F_j.B| \leq F^*$.

4. Example

Next, we will illustrate the idea of the GrA using the numerical example presented in Fig. 2. A full description of all calculations can be found in [25]. Here, we only present a short sketch. In this instance, we consider four projects with the profit functions $f_j(t),\ j = 1, 2, 3, 4$ (see Table 1).

STEP $j = 1,\ k = 1$. According to Procedure 2.1.1, we have $\Psi_j^k(x) = 0$, $\Psi_j^k.I = \{0\},\ \Psi_j^k.U = \{0\}$ and $\Psi_j^k.B = \{0\}$.
Next, we consider each iteration of the cycle [2.1.2.5] in Procedure 2.1.2.
First consider, *s'=0*. Before the first iteration, we have $T' = \emptyset,\ t_{left} = 0$,

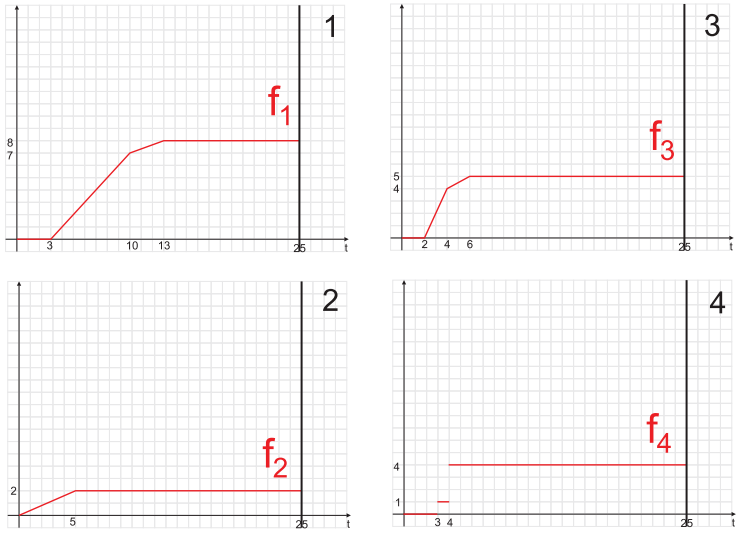

Figure 2: Functions $f_j(t)$

$t_{right} = 3$, and thus, in step [2.1.2.11], we have $\varepsilon = \min\{25-0, 25-3\} = 22$. In steps [2.1.2.12.-14], we obtain $b_{left} = 0$, $b_{right} = 0$ and $b_{inner} = 0$ and in steps [2.1.2.15.-17], we have $\varphi_{left}(x) = 0$, $\varphi_{right}(x) = 0$, $\varphi_{inner}(x) = 0$ and, as a consequence, $\varphi_{max}(x) = 0$. In step [2.1.2.24], we get $s' = s' + 22 = 22$;

So, we have $\Phi_1^1(x) = \varphi_{max}(x) = 0$ for $x = [0, 22]$ (from the previous to the current value of s').

Next, consider *s'=22*. After steps [2.1.2.22.-25] in the previous iteration, we have $T' = \{25\}$, $t_{left} = 22$ and $t_{right} = 25$. These values are used in this iteration. In step [2.1.2.11], we have $\varepsilon = 25 - 22 = 3$. Then we get $b_{left} = 0$, $b_{right} = 0$ and $b_{inner} = 0$. Moreover, $\varphi_{left}(x) = 0$, $\varphi_{right}(x) = 0$ and $\varphi_{inner}(x) = 0$. Thus, $\varphi_{max}(x) = 0$. We get $s' = 22 + 3 = 25$;
So, we have $\Phi_1^1(x) = \varphi_{max}(x) = 0$ for $x = [22, 25]$ as well and as a consequence, $\Phi_1^1(x) = 0$, $\Phi_1^1.I = \{0\}$, $\Phi_1^1.U = \{0\}$ and $\Phi_1^1.B = \{0\}$. Observe that instead of approximately 25 states t in the DPA, here we considered only two states s'. Next, we present the detailed computations for the functions Φ_1^2, Φ_1^3 and Φ_1^4.

STEP $j = 1$, $k = 2$. We have $\Psi_j^k(x) = x - 3$, $\Psi_j^k.I = \{25\}$, $\Psi_j^k.U = \{1\}$ and $\Psi_j^k.B = \{-3\}$.
First, consider *s'=0*. We get $T' = \emptyset$, $t_{left} = 3$, $t_{right} = 10$ and $\varepsilon = \min\{25 - 3, 25 - 10\} = 15$. Moreover, $b_{left} = 0$, $b_{right} = 7$ and $b_{inner} = 0$. Then $\varphi_{left}(x) = 0 + (1-1)x$, $\varphi_{right}(x) = 7 + (1-1)x$ and $\varphi_{inner}(x) = 0$. Thus, $\varphi_{max}(x) = 7$. We get $s' = s' + 15 = 15$;

Next, consider *s'=15*. We have $T' = \{25\}$, $t_{left} = 15+3 = 18$, $t_{right} = 15+10 = 25$ and $\varepsilon = 25 - 18 = 7$. Moreover, $b_{left} = -3 + 1 \cdot 18 - 1 \cdot 15 = 0$, $b_{right} = 0$

and $b_{inner} = -3 + 1 \cdot (25 - 0) - 1 \cdot 15 = 7$. We get $\varphi_{left}(x) = 0 + (1 - 1)x$, $\varphi_{right}(x) = 0$ and $\varphi_{inner}(x) = 7 - x$. Thus, we obtain $\varphi_{max}(x) = 7 - x$. We get $s' = s' + 7 = 22$;

Finally, consider *s'=22*. We have $T' = \emptyset$, $t_{left} = 25$, $t_{right} = 22 + 10 = 32$ and $\varepsilon = A - s' = 25 - 22 = 3$. Moreover, $\varphi_{left}(x) = \varphi_{right}(x) = \varphi_{inner}(x) = 0$. Then $\varphi_{max}(x) = 0$. We get $s' = s' + 3 = 25$;
We have $\Phi_1^2.I = \{15, 22, 25\}$, $\Phi_1^2.U = \{0, -1, 0\}$ and $\Phi_1^2.B = \{7, 7, 0\}$.

STEP $j = 1$, $k = 3$. We have $\Psi_j^k(x) = x + 3\frac{2}{3}$, $\Psi_j^k.I = \{25\}$, $\Psi_j^k.U = \{\frac{1}{3}\}$ and $\Psi_j^k.B = \{3\frac{2}{3}\}$. This step is performed analogously. We have to consider $s' = 0, 12, 15$.
We obtain $\Phi_1^3.I = \{12, 15, 25\}$, $\Phi_1^3.U = \{0, -\frac{1}{3}, 0\}$ and $\Phi_1^3.B = \{8, 7, 0\}$.

STEP $j = 1$, $k = 4$. We have $\Psi_j^k(x) = 8$, $\Psi_j^k.I = \{25\}$, $\Psi_j^k.U = \{0\}$ and $\Psi_j^k.B = \{8\}$. This step is performed analogously. We have to consider $s' = 0, 12$.
We obtain $\Phi_1^4.I = \{12, 25\}$, $\Phi_1^4.U = \{0, 0\}$ and $\Phi_1^4.B = \{8, 0\}$.

So, after **STEP** $j = 1$, we have $F_1(t) = \max\{\Phi_1^1, \Phi_1^2, \Phi_1^3, \Phi_1^4\}$,
$F_1.I = \{12, 15, 22, 25\}$, $F_1.U = \{0, -\frac{1}{3}, -1, 0\}$ and $F_1.B = \{8, 8, 7, 0\}$, see Fig. 3.1. In fact, the function $F_1(t)$ is obtained only from the two functions Φ_1^2 and Φ_1^3, where Φ_1^2 is the maximum function on the interval $[0, 15]$ and Φ_1^3 is the maximum function on the interval $[15, 25]$.

Next, we only present the states considered and the functions calculated in the steps $j = 2, 3, 4$. As mentioned before, the full description of all calculations can be found in [25].

STEP $j = 2, k = 1$. The states considered are $s' = 0, 7, 10, 12, 15, 17, 20, 22$. We have $\Phi_2^1.I = \{7, 10, 15, 20, 25\}$, $\Phi_2^1.U = \{0, -\frac{1}{3}, -\frac{2}{5}, -1, -\frac{2}{5}\}$ and $\Phi_2^1.B = \{10, 10, 9, 7, 2\}$, see Fig. 3.2.

STEP $j = 2$, $k = 2$. Since $f_2.U[2] = 0$, this step can be done in an easier way. It is only necessary to shift the diagram of the function $F_1(t)$ to the left by the value 5 and up by the value 2. So, we have
$\Phi_2^2.I = \{12 - 5, 15 - 5, 22 - 5, 25 - 5\}$, $\Phi_2^2.U = \{0, -\frac{1}{3}, -1, 0\}$ and $\Phi_2^2.B = \{8 + 2, 8 + 2, 7 + 2, 0 + 2\}$.
In Fig. 3.3, the maximum function is presented. In fact, we have $F_2(t) = \Phi_2^1(t)$, i.e., $F_2.I = \{7, 10, 15, 20, 25\}$, $F_2.U = \{0, -\frac{1}{3}, -\frac{2}{5}, -1, -\frac{2}{5}\}$ and $F_2.B = \{10, 10, 9, 7, 2\}$.

STEP $j = 3$, $k = 1$. Since $f_3.U[1] = 0$, this step can be done in an easier way. To obtain the function $\Phi_3^1(t)$, it is only necessary to shift the diagram of the function $F_2(t)$ to the left by the value 0 and up by the value 0.

STEP $j = 3$, $k = 2$. The states considered are $s' = 0, 3, 5, 6, 8, 11, 13, 16, 18, 21, 23$. We have $\Phi_3^2.I = \{7 - 4, 10 - 4, 15 - 4, 20 - 4, 25 - 4, 23, 25\}$, $\Phi_3^2.U = \{0, -\frac{1}{3}, -\frac{2}{5}, -1, -\frac{2}{5}, -2, 0\}$ and $\Phi_3^2.B = \{14, 14, 13, 11, 6, 4, 0\}$.

STEP $j = 3$, $k = 3$. The states considered are $s' = 0, 1, 3, 4, 6, 9, 11, 14, 16, 19, 21$.
We have $\Phi_3^3.I = \{1, 4, 9, 11, 15\frac{2}{3}, 19, 21, 25\}$, $\Phi_3^3.U = \{0, -\frac{1}{3}, -\frac{2}{5}, -\frac{1}{2}, -1,$

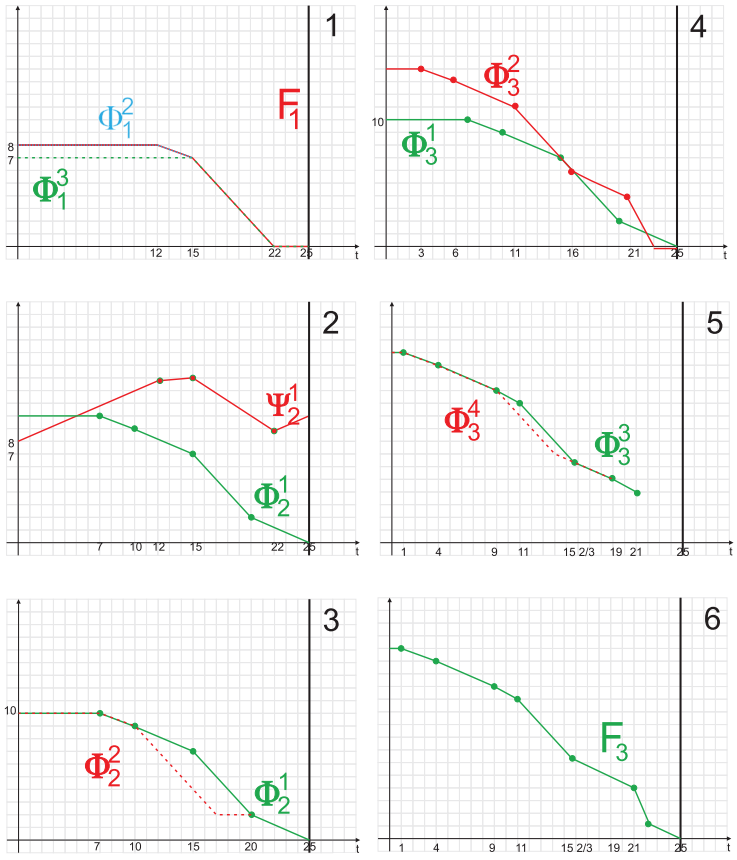

Figure 3: Calculations in the example

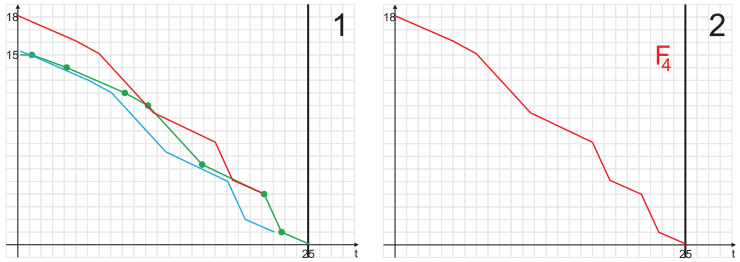

Figure 4: Function $F_4(t)$

$-\frac{2}{5}, -\frac{1}{2}, 0\}$ and $\Phi_3^3.B = \{15, 14, 12, 11, 6\frac{1}{3}, 5, 0\}$. In this example, we do not cut the point $15\frac{2}{3}$ as it is presented in Fig. 1. So, here we have two non-integer break points.

STEP $j = 3$, $k = 4$. Since $f_3.U[4] = 0$, this step can be done in an easier way. To obtain the function $\Phi_3^4(t)$, it is only necessary to shift the diagram of the function $F_2(t)$ to the left by the value 6 and up by the value 5.
The functions $\Phi_3^1(t)$ and $\Phi_3^2(t)$ are presented in Fig. 3.4, and the functions $\Phi_3^3(t)$ and $\Phi_3^4(t)$ are shown in Fig. 3.5. In Fig. 3.6, the maximum function

$$F_3(t) = \max\{\Phi_3^1(t), \Phi_3^2(t), \Phi_3^3(t), \Phi_3^4(t)\}$$

is presented. So, we have $F_3.I = \{1, 4, 9, 11, 15\frac{2}{3}, 21, 22\frac{1}{2}, 25\}$,
$F_3.U = \{0, -\frac{1}{3}, -\frac{2}{5}, -\frac{1}{2}, -1, -\frac{2}{5}, -\frac{1}{2}, -\frac{2}{5}\}$ and $F_3.B = \{15, 14, 12, 11, 6\frac{1}{3}, 4, 1\}$.

STEPS $j = 4$, $k = 1, 2, 3$ are performed in an easy way, i.e., to obtain the functions $\Phi_4^1(t), \Phi_4^2(t)$ and $\Phi_4^3(t)$, we have to shift the diagram of the function $F_3(t)$ to the left by the value $0, 3, 4$ and up by the value $0, 1, 4$, respectively. In Fig. 4, the maximum function $F_4(t)$ is presented.

To find an optimal solution at the point $s = 0$, we can do backtracking. We have $x_4 = 4$ and $f_4(x_4) = 4$, $x_3 = 6$ and $f_3(x_3) = 5$, $x_2 = 5$ and $f_2(x_2) = 2$ as well as $x_1 = 10$ and $f_1(x_1) = 7$. So, the optimal objective function value is $F^*(0) = 18$.

In the GrA, we considered the following number of states s' : $2+3+3+2 = 10$ (for $j = 1$), $8 + 4 = 12$ (for $j = 2$, where 4 states were considered for $k = 2$), $5 + 10 + 11 + 5 = 31$ (for $j = 3$, where 5 states were considered for $k = 1$ and $k = 4$), $7 + 7 + 7 = 21$ (for $j = 4$, i.e., during the shift of the diagram). So, in total we considered $10+12+31+21 = 74$ states s'. In the DPA, approximately $25(3 + 2 + 4 + 3) = 300$ states will be considered. If we scale our instance to a large number M (i.e., we multiply all input data by M), the running time of the DPA increases by the factor M, but the running time of the GrA remains the same. Of course, for each state in the GrA, we need more calculations than in the DPA. However, this number is constant and the GrA has a better running time.

5. An FPTAS based on the GrA

In this section, a fully polynomial-time approximation scheme (FPTAS) is derived based on the GrA presented in Section 3.

First, we recall some relevant definitions. For the optimization 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 less than 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 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 $1/\varepsilon$.

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.

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 $|F_j.B|$ considered, which corresponds to the number of different objective function values $b \in F_j.B, b \leq UB$. If we do not consider the original values $b \in F_j.B$ but the values $\overline{b}$ which are rounded up or down to the nearest multiple of δ values b, there are no more than $\frac{UB}{\delta} = \frac{n^2}{\varepsilon}$ different values $\overline{b}$. Then we will be able to approximate the function $F_j(t)$ into a similar function with no more than $2\frac{n^2}{\varepsilon}$ break points (see Fig. 5). Furthermore, for such a modified table representing a function $\overline{F}_j(t)$, we will have

$$|F_j(t) - \overline{F_j}(t)| < \delta \leq \frac{\varepsilon F(\pi^*)}{n}.$$

If we do the rounding and modification after each step [2.2], 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 step [2.2] will be

$$O\left(\frac{n^2 \sum k_j}{\varepsilon}\right),$$

i.e., an FPTAS is obtained.

In [28], a technique was proposed to improve the complexity of an approximation algorithm for optimization problems. This technique can be described as follows. Let there exist 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 input length of the problem instance and UB, LB are known upper and lower bounds, respectively. Let the value $\frac{UB}{LB}$ be not bounded by a constant. Then this 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

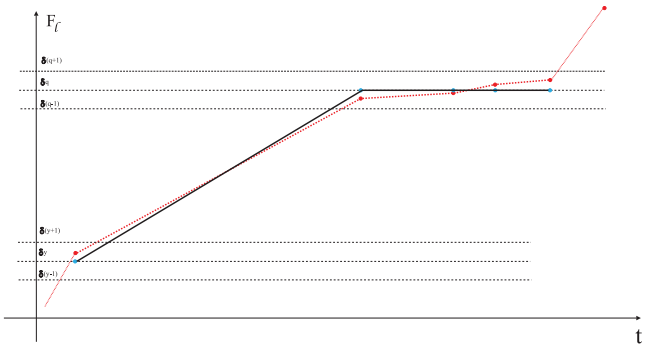

Figure 5: Substitution of columns and modification of $F_l(t)$

polynomial. So, by using this technique, we can improve the FPTAS to have a running time of

$$O\left(\frac{n \cdot \sum k_j}{\varepsilon}(1 + \log\log n)\right),$$

Finally, we only note that an FPTAS based on a GrA was presented in [29] for some single machine scheduling problems.

6. Concluding Remarks

In this paper, we used a graphical approach to improve a known pseudo-polynomial algorithm for a project investment problem and to derive an FPTAS with the best known running time.

The practical usefulness of the graphical approach is not limited to such a project investment problem or similar warehousing and lot sizing problems. The graphical approach can be applied to problems, for which a pseudo-polynomial algorithm exists and Boolean variables are used in the sense that yes/no decisions have to be made. This is the case for many applications of capital budgeting in science and engineering. However, e.g., for the knapsack problem, the graphical algorithm mostly reduces substantially the number of states to be considered but the time complexity of the algorithm remains pseudo-polynomial [26]. On the other side, e.g., for the single machine scheduling problem of maximizing total tardiness, such a graphical algorithm improved the complexity from $O(n \sum p_j)$ to $O(n^2)$ [30]. It seems to be challenging for future research to study other well-known combinatorial optimization problems with this approach to know exactly its performance for different problems and to deduce more general properties.

Acknowledgments

This work has been supported by grants RFBR 13-01-12108, 13-08-13190, 15-07-03141, 15-07-07489 and DAAD A/1400328 and HSE Faculty of Economics.

References

[1] Tavana, M., Khalili-Danghani, K., and Abtahi, A.R. 2013. "A fuzzy multi-dimensional multiple-choice model for project portfolio selection using an evolutionary algorithm." *Annals of Operations Research* 206 (1): 449 - 483.

[2] Chang, P.-T., and Lee, J.-H. 2012. "A fuzzy DEA and knapsack formulation integrated model for project selection." *Computers & Operations Research* 30: 112 - 125.

[3] Beaujon, G.J., Marin, S.P., McDonald, G.C. 2001. "Balancing and optimizing a portfolio of R & D projects." *Naval Research Logistics* 48: 18 - 40.

[4] Mavrotas, G., Diakoulaki, D. and Kourentzis, A. 2008. "Selection among ranked projects under segmentation, policy and logical constraints." *European Journal of Operational Research* 187 (1): 177 - 192.

[5] Herbots, J., Herroelen, W., and Leus, R. (2007). "Dynamic order acceptance and capacity planning on a single bottleneck resource." *Naval Research Logistics* 54: 874 - 889.

[6] Kellerer, H., and Pferschy, U. 2004. "Improved dynamic programming in connection with a FPTAS for the knapsack problem." *Journal of Combinatorial Optimization* 8: 5 - 11.

[7] Caprara, A., Kellerer, H., Pferschy, U. and Pisinger, D. 2000. "Approximation algorithms for knapsack problems with cardinality constraints." *European Journal of Operational Research* 123 (2): 333 - 345.

[8] Mastrolilli, M., and Hutter, M. 2006. "Hybrid rounding techniques for knapsack problems. " *Discrete Applied Mathematics* 154 (4): 640 - 649.

[9] Bazgan, C., Hugot, H., and Vanderpoorten, D. (2009). "Implementing an efficient FPTAS for the 0-1 multi-objective knapsack problem." *European Journal of Operational Research* 198 (1): 47 - 56.

[10] Sristava, V. and Bullo, F. 2014. "Knapsack problems with sigmoid utilities: Approximation algorithms via hybrid optimization." *European Journal of Operational Research* 236 (2): 488 - 498.

[11] Guler, A., Nuriyev, U.G., Berberler, M.E., and Nurieva, F.(2012) "Algorithms with guarantee value for knapsack problems." *Optimization* 61 (4): 477 - 488.

[12] Wu, F.L., Yen, Z., Hou, Y.H. and Ni, Y.X. 2004. "Applications of AI techniques to generation planning and investment." 2004, *IEEE Power Engineering Society General Meeting*, Denver, 936 - 940.

[13] Li, X. M., Fang, S.-C., Tian, Y., and Guo, X. L. 2014. Expanded Model of the Project Portfolio Selection Problem with Divisibility, Time Profile Factors and Cardinality Constraints." *Journal of the Operational Research*, doi:10.1057/jors.2014.75.

[14] Dolgui, A., and Proth, J-M. 2010. *Supply Chain Engineering: Useful Methods and Techniques*. Springer-Verlag.

[15] Schemeleva, K., Delorme, X., Dolgui, A., Grimaud, F., Kovalyov, M.Y. 2013. "Lot-sizing on a Single Imperfect Machine: ILP Models and FPTAS Extensions." *Computers and Industrial Engineering* 65 (4): 561 – 569.

[16] Kellerer, H., Pferschy, U., and Pisinger, D. 2004. *Knapsack Problems*. Springer-Verlag, Berlin.

[17] McLay, L.A., and Jacobson, S.H. 2007. "Algorithms for the bounded set-up knapsack problem." *Discrete Optimization* 4; 206 - 412.

[18] Posypkin, M.A., and Sigal, I.Kh. 2006. "Speedup Estimates for Some Variants of the Parallel Implementations of the Branch-and-Bound Method." *Mathematics and Mathematical Physics* 46 (12): 2189-2202.

[19] O'Neil, E.T., and Kerlin, S. 2010. "A Simple $2^{O(\sqrt{x})}$ Algorithm for PARTITION and SUBSET SUM.".http://www.lidi.info.unlp.edu.ar/WorldComp2011-Mirror/FCS8171.pdf

[20] Bar-Noy, A., Golin, M.J., and Zhang, Y. 2009. "Online Dynamic Programming Speedups." *Theory Comput. Syst* 45 (3): 429-445.

[21] Eppstein, D., Galil, Z., and Giancarlo, R. 1988. "Speeding up Dynamic Programming." *In Proc. 29th Symp. Foundations of Computer Science*.

[22] Wagelmans, A.P.M., and Gerodimos, A.E. 2000. "Improved Dynamic Programs for Some Batching Problems Involving the Maximum Lateness Criterion." *Operations Research Letters* 27: 109 –118.

[23] Shaw, D.X., and Wagelmans, A.P.M. 1998. "An Algorithm for Single-Item Capacitated Economic Lot Sizing with Piecewise Linear Production Costs and General Holding Costs." *Management Science* 44 (6): 831–838.

[24] Kameshwaran, S. and Narahari, Y. 2009. "Nonconvex Piecewise Linear Knapsack Problems." *European Journal of Operational Research* 192: 56–68.

[25] Gafarov, E.R., Dolgui, A., Lazarev, A.A., and Werner, F. 2014. "A Graphical Approach to Solve an Investment Optimization Problem." *Journal of Mathematical Modelling and Algorithms in Operations Research* 13 (4): 597 - 614.

[26] Lazarev, A.A., and Werner, F. 2009. "A Graphical Realization of the Dynamic Programming Method for Solving NP-hard Problems." *Computers & Mathematics with Applications* 58 (4): 619 – 631.

[27] Aho, A.V., Hopcroft, J.E., and Ullman, J.D. 1983. *Data Structures and Algorithms*. Addison-Wesley, London.

[28] Chubanov, S., Kovalyov, M.Y., and Pesch, E. 2006. "An FPTAS for a Single-Item Capacitated Economic Lot-Sizing Problem with Monotone Cost Structure." *Mathematical Programming* 106: 453 – 466.

[29] Gafarov, E.R., Dolgui, A., and Werner, F. 2014. "A Graphical Approach for Solving Single Machine Scheduling Problems Approximately." *International Journal of Production Research* 52 (13): 3762 - 3777.

[30] Gafarov, E.R., Lazarev, A.A., and Werner, F. 2012. "Transforming a Pseudo-Polynomial Algorithm for the Single Machine Total Tardiness problem into a Polynomial One." *Annals of Operations Research* 196: 247 – 261.