© 2015 г. Е.Р. ГАФАРОВ, канд. физ.-мат. наук (axel73@mail.ru)
(Институт проблем управления им. В.А.Трапезникова РАН, Москва),
А. ДОЛГИЙ, док. физ.-мат. наук
(Высшая национальная горная школа Нанси, Франция),
А.А. ЛАЗАРЕВ, д-р физ.-мат. наук (jobmath@mail.ru)
(Институт проблем управления им. В.А.Трапезникова РАН, Москва,
Московский государственный университет им. М.В. Ломоносова,
Московский физико-технический институт
(Государственный университет),
Высшая школа экономики (Национальный
исследовательский университет), Москва),
Ф. ВЕРНЕР, д-р философии
(Факультет математики университета Отто фон Герике,
Магдебург, Германия)

НОВЫЙ ЭФФЕКТИВНЫЙ АЛГОРИТМ РЕШЕНИЯ ЗАДАЧИ ОБ ИНВЕСТИЦИЯХ[1]

В серии публикаций О'Нила (2010) показано, что динамическое программирование представляется наиболее перспективным методом решения задач комбинаторной оптимизации. Известны несколько модификаций алгоритмов динамического программирования (DPA, dynamic programming algorithms). Одна из таких модификаций – графический метод для задач с кусочно-линейными функциями Беллмана. В предыдущих работах было показано, что для некоторых задач графические алгоритмы имеют меньшую трудоёмкость и ряд других преимуществ в сравнении с классическими DPA. В данной работе представлен графический алгоритм точного решения и основанная на нем схема аппроксимации с полиномиальным временем работы для задачи об инвестициях. Данные алгоритмы имеют меньшую трудоемкость среди известных алгоритмов.

1. Введение

Выбор инвестиционных проектов и объема инвестиций является важной практической задачей. Обычно такая задача формулируется в виде задачи о Ранце в более общем виде – многомерной мультивыборной задачей о Ранце. Данные задачи имеют широкий спектр приложений в бизнесе, в задачах расчета рентабельности капиталовложений и планирования производства.

Известно [1], что для задачи о Ранце, все алгоритмы ветвей и границ с полиномиальными алгоритмами расчета верхних и нижних оценок имеют трудоемкость не меньше $\frac{3}{2}\frac{2^{n+3/2}}{\sqrt{\pi(n+1)}}$, т.е. $2^{O(x)}$ операций, где n – количество предметов, а x – длина входа.

[1]Работа выполнена при финансовой поддержке грантов: Российского фонда фундаментальных исследований № 13-01-12108, № 13-08-13190, № 15-07-03141, № 15-07-07489; DAAD A/1400328; факультета экономики НИУ Высшей Школы Экономики.

По сути, такие алгоритмы ветвей и границ не сильно отличаются по трудоемкости от полного перебора, имеющего экспоненциальную трудоемкость. Аналогичные результаты получены и для других задач.

В 2010 году, в работе О'Нила [2] показано, что широко известная модификация метода динамического программирования позволяет решать многие классические задачи за субэкспоненциальное время $2^{O(\sqrt{x})}$, где x – длина входа.

То есть на данный момент с теоретической точки зрения метод динамического программирования является более перспективным чем метод ветвей и границ, и его развитие является оправданным.

Стоит отметить следующие работы, посвященные развитию метода динамического программирования.

В работе [3] рассматриваются рекурсивные отношения вида $h(k) = \min\limits_{1 \leqslant j \leqslant k} a(k,j)$, $k = 1, \ldots, n$, где $a(k,j)$ – некоторая формула, в которой участвуют значения $h(j)$, $1 \leqslant j < k$. В этом случае трудоемкость классического алгоритма динамического программирования – $O(n^2)$ операций. Ее удается сократить до $O(n)$ операций, если выполняется следующее условие

$$a(k,j) + a(k+1,j+1) \leqslant a(k+1,j) + a(k,j+1), \quad 1 \leqslant j < k \leqslant n.$$

Похожие результаты представлены в работе [4], где рассматриваются следующие рекурсивные соотношения $E[j] = w(i,j) + \min\limits_{1 \leqslant i \leqslant j} D[i]$, $i < j$, $j = 1, \ldots, n$, где $D[i]$ вычисляется по $E[i]$, $i < j$, за линейное время. На случай вогнутой или выпуклой функции $w(i,j)$ трудоемкость алгоритма динамического программирования, использующего данные рекурсивные соотношения, удалось сократить с $O(n^2)$ до $O(n \log n)$ операций за счет специальной структуры хранения данных.

В работе [5] представлен улучшенный алгоритм динамического программирования для задачи теории расписаний минимизации максимального временно́го смещения для одного прибора. Трудоемкость решения задачи удалось сократить с $O(n^2)$ до $O(n \log n)$.

Рассматриваемая задача формулируется следующим образом. Даны множество N из n потенциальных инвестиционных проектов и доступный объем инвестиций $A > 0$. Для каждого проекта j, $j = 1, \ldots, n$, задана функция прибыли $f_j(t)$, $t \in [0, A]$. Значение $f_j(t')$ означает прибыль, полученную от проекта j если в него инвестировать t' денежных единиц. Необходимо определить объёмы инвестиций $\tau_j \in [0, A]$, $\tau_j \in Z$, для каждого проекта $j \in N$ так, чтобы $\sum\limits_{j:=1}^{n} \tau_j \leqslant A$, и значение $\sum\limits_{j:=1}^{n} f_j(\tau_j)$ было максимальным.

На практике встречается задача в расширенной постановке, где необходимо найти оптимальное решение (стратегию инвестирования) для всех $A \in [A', A'']$. То есть нужно построить эффективный алгоритм, находящий оптимальные решения для всех A из некоторого интервала.

Будем считать, что все функции $f_j(t)$, $j = 1, \ldots, n$, являются неубывающими **кусочно-линейными**. Если функция $f_j(t)$ определена только для некоторых значений t, например, $t^1, t^2, \ldots, t^{k_j}$, $t^1 < t^2 < \cdots < t^{k_j}$, тогда будем считать, что $f_j(t) = f_j(t^i)$ для всех $t \in [t^i, t^{i+1})$, $i = 1, \ldots, k_j$, где $t^{k_j+1} = A$.

Таблица 1. Функция $f_j(t)$

K	1	2	...	k_j
интервал K	$[t_j^1, t_j^2)$	$[t_j^2, t_j^3)$	...	$[t_j^{k_j}, A)$
b_j^K	b_j^1	b_j^2	...	$b_j^{k_j}$
u_j^K	u_j^1	u_j^2	...	$u_j^{k_j}$

Интервал $[0, A]$ может быть представлен в виде объединения множества интервалов

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

таких, что $f_j(x) = b_j^k + u_j^k(x - t_j^{k-1})$ для всех $x \in (t_j^{k-1}, t_j^k]$, где k – номер интервала, b_j^k – значение функции в начале интервала, и u_j^k – наклон графика функции. Без ограничения общности, пусть $b_j^1 \leqslant b_j^2 \leqslant \ldots \leqslant b_j^{k_j}$, $t_j^k \in Z$, $j \in N$, $k = 1, 2, \ldots, k_j$, и $t_j^{k_j} = A$, $j = 1, 2, \ldots, n$.

Известные следующие близкие задачи.

Задача

$$(1) \qquad \begin{array}{ll} \text{maximize} & \sum\limits_{j:=1}^{n} p_j x_j \\ & \sum\limits_{j:=1}^{n} w_j x_j \leqslant A, \\ & x_j \in [0, b_j],\ x_j \in Z,\ j = 1, 2, \ldots, n, \end{array}$$

для которой известен DPA трудоёмкости $O(nA)$ операций [6].

И задача

$$(2) \qquad \begin{array}{ll} \text{minimize} & \sum\limits_{j:=1}^{n} f_j(x_j) \\ & \sum\limits_{j:=1}^{n} x_j \geqslant A, \\ & x_j \in [0, A],\ x_j \in Z,\ j = 1, 2, \ldots, n, \end{array}$$

где функции $f_j(x_j)$ также кусочно-линейные. Для данной задачи известны DPA трудоёмкости $O(\sum k_j A)$ операций [7] и FPTAS (полностью полиномиальная схема приближённого решения) трудоёмкости $O((\sum k_j)^3/\varepsilon)$ операций [8].

Данная задача является NP-трудной в обычном смысле и может быть решена с помощью DPA, использующем следующие уравнения Беллмана:

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

Трудоёмкость DPA – $O(nA^2)$ операций. Очевидно, что функции $f_j(t)$, $j = 1, \ldots, n$, могут быть представлены в табличном виде, Табл. 1. В [9] приводится информация об альтернативном DPA других исследователей для задачи о размере партии продукции. Трудоёмкость этого алгоритма – $O(\sum k_j A)$ операций. Для данной задачи также известна FPTAS трудоёмкостью $O((\sum k_j)^3/\varepsilon)$ операций.

2. Алгоритм динамического программирования

Для каждого проекта j и состояния $t \in [0, A]$ определим $F_j(t)$ – максимальную прибыль, полученную для проектов $1, 2, \ldots, j$, когда сумма инвестиций для остальных проектов $j+1, j+2, \ldots, n$ равна t. То есть:

$$
\begin{aligned}
F_j(t) = \max \quad & \sum_{h:=1}^{j} f_h(x_h) \\
& \sum_{h:=1}^{j} x_h \leqslant A - t, \\
& x_h \geqslant 0,\ x_h \in Z,\ h = 1, 2, \ldots, j.
\end{aligned} \tag{3}
$$

Пусть $F_j(t) = 0$ для $t \notin [0, A],\ j = 1, 2, \ldots, n$ и $F_0(t) = 0$ для любого состояния t. Тогда имеем следующие функциональные уравнения:

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

Лемма 1. Все функции $F_j(t),\ j = 1, 2, \ldots, n$, невозрастающие на интервале $[0, A]$.

Доказательство леммы непосредственно следует из определения функций (3).

DPA, основанный на уравнениях (4), может быть организован следующим образом. Для каждой стадии $j = 1, \ldots, n$, для состояния $t = 0$ вычислим не более $k_j A$ значений $v_k^x = \{b_j^k - u_j^k t_j^{k-1} + u_j^k \cdot x + F_{j-1}(t+x)\}$, $1 \leqslant k \leqslant k_j$, $x \in (t_j^{k-1}, t_j^k]$ и поместим их в соответствующий список L_k (значения последовательно помещаются в конец списка). Если выполняется $v_k^x \leqslant v_k^{x-1}$, то не помещаем значение v_k^x в список. Таким образом элементы списка L_k упорядочены по неубыванию. Для очередной стадии $t = 1$, необходимо только исключить последний элемент из множества $L_k,\ k = 1, \ldots, n$, если он соответствует x не из интервала $(t_j^{k-1}, t_j^k] \bigcap [0, A - t]$ и сравнить новые k_j последних элементов из списков. Продолжая вычисления для всех $t = 2, \ldots, A$, получим функцию $F_j(t)$, $t = 1, 2, \ldots, A$, за $O(k_j A)$ операций. Трудоёмкость такого DPA – $O(\sum k_j A)$ операций. Похожая идея представлена в [7].

Алгоритмы, полученные авторами в [10] для исследуемой задачи, основаны на функциональных уравнениях (3) и другой технике выполнения графического метода. В отличие от этого подхода, представленные в данной работе алгоритмы основаны на уравнениях (4).

3. Графический алгоритм

Для краткости будем обозначать алгоритм динамического программирования через DPA (Dynamic Programming Algorithm), а графический алгоритм, соответственно, *GrA* (Graphical Algorithm). Пусть в DPA на каждом шаге (на каждой стадии) j вычисляются значения некоторой функции $F_j(t)$ (функции Беллмана) для каждого возможного значения аргумента t (для каждого состояния) процесса принятия решения, где $t \in [0, A]$, $t \in Z$, и A – числовой параметр задачи. Фактически, эта функция

соответствует значениям целевой функции для подзадачи размерности j, если она решается в условиях (при состоянии системы) t. На практике DPA реализуется в режиме "калькулятора", т.е. перебираются все состояния $t \in [0, A] \bigcap Z$. Для каждого состояния проводятся несложные вычисления, и полученное значение $F_j(t)$ сохраняется в ячейку памяти (в таблицу). Эти $A + 1$ числовых значений используются на следующем шаге $j + 1$.

Однако, зачастую нет необходимости вычислять (и сохранять в памяти) значения $F_j(t)$ для каждой точки $t \in [0, A] \bigcap Z$. Для многих задач на очередном интервале $[t_l, t_{l+1})$ вычисляемая функция представима аналитически в виде $F_j(t) = \varphi(t)$ (например, $F_j(t) = k \cdot t + b$, т.е. функция $F_j(t)$ определена в том числе для нецелых значений аргумента t). Тогда процесс вычисления функции $F_{j+1}(t)$ на шаге $j + 1$ можно организовать таким образом, чтобы учитывать не отдельные значения $F_j(t)$, а преобразовывать функцию $F_j(t)$ в $F_{j+1}(t)$ аналитически, согласно заданным рекурсивным уравнениям Беллмана.

Кусочно-линейные функции $\varphi(x)$, рассматриваемые в данном параграфе, могут быть определены с помощью трех множеств чисел: множества точек излома I (в каждой точке излома заканчивается некоторый кусочно-линейный фрагмент функции), множества наклонов графика функции U и множества значений функции в начале каждого интервала B. Через $I[k]$ будем обозначать k-й элемент в упорядоченном множестве I. Аналогичные обозначения будем использовать для множеств U и B. Множества I, U и B, задающие функцию $\varphi(x)$, будем обозначать $\varphi.I$, $\varphi.U$ и $\varphi.B$ соответственно. Пусть запись $\varphi.I[k]$ обозначает k-й элемент множества I для функции $\varphi(x)$. Тогда, например, для $x \in (t_j^{k-1}, t_j^k] = (f_j.I[k-1], f_j.I[k]]$, выполняется

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

Заметим, что $\varphi.I[k] < \varphi.I[k+1],\ k = 1, 2, \ldots, |\varphi.I| - 1$. Напомним, что $|f_j.I| = k_j$. Далее приводится доказательство, что все функции $F_j(t),\ j = 1, 2, \ldots, n$, являются кусочно-линейными. Для целых значений t значения функции $F_j(t)$ равны значениям функции Беллмана, вычисляемой в DPA.

Пусть $\varphi.I[0] = 0$ и $\varphi.I[|\varphi.I| + 1] = A$.

Точки $t \in \varphi.I$, в которых изменяется уравнение прямой, задающей кусочно-линейную функцию, будем называть *точками излома*. В GrA выражение *вычислить функцию* означает вычислить множества I, U и B для данной функции.

На каждой стадии $j = 1, \ldots, n$ алгоритма вычисляются временные функции $\Psi_j^k(t)$ и $\Phi_j^k(t)$, на основе которых вычисляется $F_j(t)$. Ключевая часть алгоритма – вычисление функции $\Phi_j^k(t)$.

Графический алгоритм

1. Пусть $F_0(t) = 0$, т.е. $F_0.I := \{A\},\ F_0.U := \{0\},\ F_0.B := \{0\}$;
2. Для $j := 1$ по n цикл
   - 2.1. Для $k := 1$ по k_j цикл
     - 2.1.1. Вычислим промежуточную функцию

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

     в соответствие с Процедурой 2.1.1;

2.1.2. Вычислим промежуточную функцию

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

в соответствие с Процедурой 2.1.2;

2.1.3. Если $k = 1$, тогда $F_j(t) := \Phi_j^k(t)$, иначе $F_j(t) := \max\{F_j(t), \Phi_j^k(t)\}$.

2.2. Модифицируем множества I, U, B функции $F_j(t)$ в соответствие с Процедурой 2.2.

3. Оптимальное значение целевой функции равно $F_n(0)$.

Ниже приводятся Процедуры 2.1.1 и 2.1.2.

Процедура 2.1.1.

Заданы k и j;

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

Для $i := 1$ по $|F_{j-1}.I|$ цикл

добавим значение $F_{j-1}.I[i]$ в множество $\Psi_j^k.I$;

добавим значение

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

в множество $\Psi_j^k.B$;

добавим значение $f_j.U[k] + F_{j-1}.U[i]$ в множество $\Psi_j^k.U$.

В Процедуре 2.1.1. "сдвигаем" функцию $F_{j-1}(t)$ вверх на значение $f_j.B[k] - f_j.U[k] \cdot f_j.I[k-1]$ и увеличиваем все наклоны линейных фрагментов на $f_j.U[k]$. Если все числа $t \in F_{j-1}.I$ являются целыми, то все числа из множества $\Psi_j^i.I$ – также целые. Очевидно, что Процедура 2.1.1 выполняется за $O(|F_{j-1}.I|)$ операций.

Перед описанием процедуры 2.1.2. представим используемую в ней Процедуру FindMax, в которой вычисляется функция максимума $\varphi(t)$ двух линейных функций $\varphi_1(t)$ и $\varphi_2(t)$.

Процедура FindMax

1. Заданы функции $\varphi_1(t) = b_1 + u_1 \cdot t$ и $\varphi_2(t) = b_2 + u_2 \cdot t$ и интервал $(t', t'']$. Пусть $u_1 \leqslant u_2$;

2. Если $t'' - t' \leqslant 1$, тогда вернуть $\varphi(t) = \max\{\varphi_1(t''), \varphi_2(t'')\} + 0 \cdot t$, определенную на интервале $(t', t'']$;

3. Найдем точку пересечения t^* функций $\varphi_1(t)$ и $\varphi_2(t)$;

4. Если t^* не существует или $t^* \notin (t', t'']$, тогда

   если $b_1 + u_1 \cdot t' > b_2 + u_2 \cdot t'$, тогда вернуть $\varphi(t) := \varphi_1(t)$, определенную на интервале $(t', t'']$;

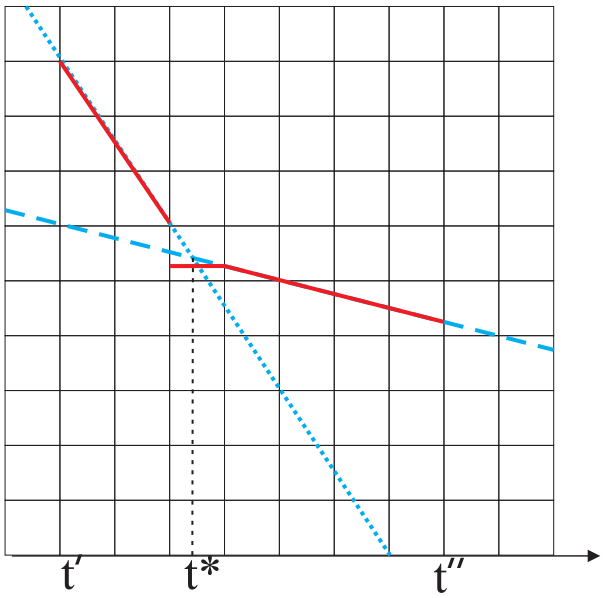

Рис. 1. Процедура FindMax. Исключение нецелых точек

иначе вернуть $\varphi(t) := \varphi_2(t)$, определенную на интервале $(t', t'']$;

5. Иначе

если $t^* \in Z$ тогда

$\varphi(t) := \varphi_1(t)$ на интервале $(t', t^*]$;
$\varphi(t) := \varphi_2(t)$ на интервале $(t^*, t'']$;
вернуть $\varphi(t)$;

иначе, если $t^* \notin Z$, тогда

$\varphi(t) := \varphi_1(t)$ на интервале $(t', \lfloor t^* \rfloor]$;
$\varphi(t) := b_2 + u_2 \cdot \lfloor t^* \rfloor + 0 \cdot t$ на интервале $(\lfloor t^* \rfloor - 1, \lfloor t^* \rfloor]$;
$\varphi(t) := \varphi_2(t)$ на интервале $(\lfloor t^* \rfloor, t'']$;
вернуть $\varphi(t)$;

Процедура выполняется за $O(1)$ операций.

Процедура 2.1.2.

2.1.2.1. Заданы k, j и $\Psi_j^k(t)$;

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

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. Пусть $T' = \{\Psi_j^k.I[v], \Psi_j^k.I[v+1], \ldots, \Psi_j^k.I[w]\}$ – подмножество максимальной мощности $\Psi_j^k.I$, где $t_{left} < \Psi_j^k.I[v] < \cdots < \Psi_j^k.I[w] < t_{right}$,

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

2.1.2.5. Пока $s' \leqslant A$ цикл

2.1.2.6. Если $T' = \emptyset$, тогда пусть
$w+1 = \underset{i=1,2,\ldots,|\Psi_j^k.I|}{argmax} \{\Psi_j^k.I[i] | \Psi_j^k.I[i] > t_{right}\}$ и
$v = \underset{i=1,2,\ldots,|\Psi_j^k.I|}{argmin} \{\Psi_j^k.I[i] | \Psi_j^k.I[i] > t_{left}\}$;

2.1.2.7. Если $w+1$ не определена, тогда $w+1 := |\Psi_j^k.I|$;

2.1.2.8. Если v не определена, тогда $v := |\Psi_j^k.I|$;

2.1.2.9. Если $t_{left} < A$, тогда $\varepsilon_{left} := \Psi_j^k.I[v] - t_{left}$ иначе $\varepsilon_{left} := A - s'$;

2.1.2.10. Если $t_{right} < A$, тогда $\varepsilon_{right} := \Psi_j^k.I[w+1] - t_{right}$
иначе $\varepsilon_{right} := +\infty$;

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

2.1.2.12. Если $t_{left} < A$, тогда

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

иначе $b_{left} := 0$;

2.1.2.13. Если $t_{right} < A$, тогда

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

иначе $b_{right} := 0$;

2.1.2.14. Если $T' = \emptyset$, тогда $b_{inner} := 0$ иначе

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

2.1.2.15. Определим функцию

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

Если $t_{left} = A$, тогда $\varphi_{left}(x) := 0$;

2.1.2.16. Определим функцию

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

Если $t_{right} = A$, тогда $\varphi_{right}(x) := 0$;

2.1.2.17. Определим функцию

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

Если $T' = \emptyset$, тогда $\varphi_{inner}(x) := 0$;

2.1.2.18. Вычислим кусочно-линейную функцию

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

в соответствие с Процедурой *FindMax*;

2.1.2.19. Добавим числа из $\varphi_{max}.I$ увеличенные на s' в множество $\Phi_j^k.I$;

2.1.2.20 Добавим числа из $\varphi_{max}.B$ в множество $\Phi_j^k.B$;

2.1.2.21. Добавим числа из $\varphi_{max}.U$ в множество $\Phi_j^k.U$;

2.1.2.22. Если $\varepsilon = \varepsilon_{left}$, тогда исключим $\Psi_j^k.I[v]$ из множества T и присвоим $v := v + 1$;

2.1.2.23. Если $\varepsilon = \varepsilon_{right}$, тогда включим $\Psi_j^k.I[w+1]$ в множество T и присвоим $w := w + 1$;

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

2.1.2.25. $t_{left} := s' + f_j.I[k-1]$, $t_{right} := \min\{s' + f_j.I[k], A\}$, актуализируем T'.

2.1.2.26. Преобразуем функцию Φ_j^k в соответствие с Процедурой 2.2.

В процедуре 2.1.2, выполняется следующее. Когда сдвигаем s' вправо, то сдвигается интервал $I' = [t_{left}, t_{right}]$ длины $f_j.I[k] - f_j.I[k-1]$. Значения $\Psi_j^k(x)$ для $x \in T'$ используются для вычисления $\Phi_j^k(t)$ в точке $t = s'$. Так как $\Psi_j^k(x)$ – кусочно-линейная, то необходимо и достаточно рассмотреть значения $\Psi_j^k(x)$ в точках излома из множества T', а также в начале и конце интервала T'. То есть если s' сдвигать вправо на некоторую величину $x \in [0, \varepsilon]$, такую, что все точки излома из интервала остаются теми же, тогда функция $\Phi_j^k(t)$ изменяется согласно функции $\varphi_{max}(x)$.

Процедура 2.2.
Задана $F_j(t)$;
для $k := 1$ по $|F_j.I| - 1$ цикл

если $F_j.U[k] = F_j.U[k+1]$ и $F_j.U[k] \cdot (F_j.U[k] - F_j.U[k-1]) + F_j.B[k] = F_j.B[k+1]$ тогда

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

удалим k-е элементы из множеств $F_j.B$, $F_j.U$ и $F_j.I$;

В Процедуре 2.2. два смежных линейных фрагмента находящиеся на одной прямой, объединяются в один фрагмент. Если имеем два смежных линейных фрагмента с наклонами $F_j.U[k]$, $F_j.U[k+1]$ и $F_j.B[k], F_j.B[k+1]$, где $F_j.U[k] \cdot (F_j.U[k] - F_j.U[k-1]) + F_j.B[k] = F_j.B[k+1]$, (фрагменты на одной прямой), тогда для сокращения количества интервалов $|F_j.I|$ и, как следствие, для сокращения трудоёмкости алгоритма необходимо объединить их в один фрагмент.

Лемма 2. Процедура 2.1.2. выполняется за $O(|F_{j-1}.I|)$ операций.

Доказательство: Шаг [2.1.2.14] и пересчет T' на шаге [2.1.2.25] необходимо выполнять с использованием следующей структуры. Пусть $\{q_1, q_2, \ldots, q_r\}$ – максимальное подмножество множества T' следующего свойства

$q_1 < q_2 < \cdots < q_r$;

нет такого $q \in T'$, что $q_i \leqslant q < q_{i+1}$ и
$\Psi_j^k.B[q] + \Psi_j^k.U[q] \cdot (\Psi_j^k.I[q] - \Psi_j^k.I[q-1]) \geqslant \Psi_j^k.B[q_{i+1}] + \Psi_j^k.U[q_{i+1}] \cdot (\Psi_j^k.I[q_{i+1}] - \Psi_j^k.I[q_{i+1}-1])$,
$i = 1, \ldots, r-1$.

Множество будем хранить в виде списка, в котором элементы упорядочены по возрастанию так, что элементы в начале списка будут только удаляться, в то время, как в конце списка они будут удаляться и добавляться [11]. Тогда добавление и удаление элементов будет выполняться за фиксированное количество операций, и шаги [2.1.2.14] и [2.1.2.25] могут быть выполнены за фиксированное количество операций (ограниченной некоторой константой).

Шаги [2.1.2.6]–[2.1.2.25] выполняются за фиксированное количество операций. Цикл [2.1.2.5] может быть выполнен за $O(|\Psi_j^k.I|)$ операций, где $|\Psi_j^k.I| = |F_{j-1}(t).I|$, т.к. каждая точка излома из $\Psi_j^k.I$ добавляется или удаляется.

Лемма доказана. □

В Процедуре 2.1.1. вычисляется функция

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

а в Процедуре 2.1.2. – функция

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

В отличие от DPA для вычисления $\Phi_j^k(t)$ в GrA нет необходимости рассматривать все целые точки $x \in (t_j^{k-1}, t_j^k] \bigcap [0, A-t]$, но только точки излома из интервала, т.к. только они оказывают влияние на значения $\Phi_j^k(t)$ (как и t_{left}, t_{right}). Шаг [2.1.3] также может быть выполнен в соответствие с Процедурой FindMax, т.е. для вычисления функции $F_j(t) := \max\{F_j(t), \Phi_j^i(t)\}$, линейные фрагменты функций сравниваются на всех интервалах, ограниченных их точками излома. Очевидно, что для целых точек t производимые операции аналогичны операциям в DPA. Тогда значения $F_j(t),\ t \in Z$, равны в GrA и DPA и можно утверждать следующее.

Лемма 3. Значения $F_j(t),\ j = 1, 2, \ldots, n$, в точках $t \in [0, A] \bigcap Z$ равны значениям $F_j(t)$ рассмотренным в DPA.

Все функции $F_j(t),\ j = 1, 2, \ldots, n$, являются кусочно-линейными на интервале $[0, A]$. Все их точки излома – целые числа.

Доказательство: Для $F_0(t)$ лемма верна. После выполнения Процедуры 2.1.1. все элементы множества $\Psi_1^i.I$ – целые. Так как все элементы $f_1.I$ – целые числа, тогда $\varepsilon \in Z$ и как следствие $s' \in Z$. В соответствии с процедурой FindMax все точки $\varphi_{\max}.I$, рассмотренные в Процедуре 2.1.2, целые. Тогда все точки $\Phi_j^i.I,\ i = 1, 2, \ldots, k_j$, также целые, и точки излома функции $F_1(t) := \max\{F_0(t), \Phi_1^i(t)\}$ целые, если процедура FindMax использована при вычислении $\max\{F_0(t), \Phi_1^i(t)\}$. Аналогично можно показать, что все точки излома функции $F_2(t)$ целые.

Все функции $F_j(t),\ j = 1, 2, \ldots, n$, вычисляемые в GrA, являются кусочно-линейными. Таким образом лемма верна.

□

Теорема 1. С помощью GrA оптимальное решение будет найдено за

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

операций.

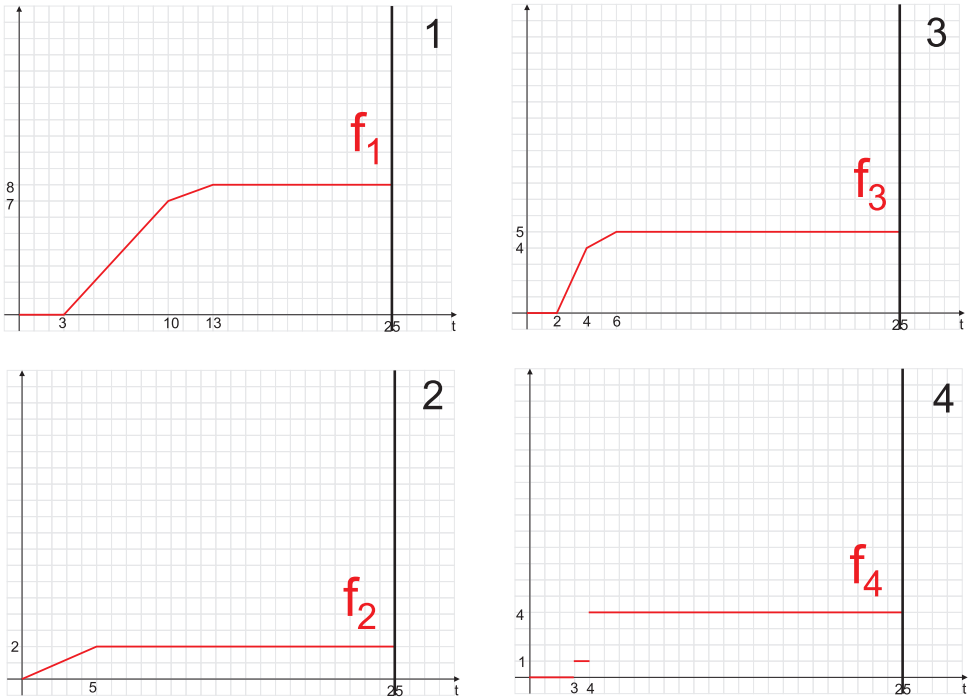

Рис. 2. Функции $f_j(t)$

Доказательство: Аналогично доказательству леммы 3. После каждого шага [2.1.3] GrA функция $F_j(t),\ j = 1, 2, \ldots, n$, имеет только целые точки излома из интервала $[0, A]$. Каждая функция $\Phi_j^i.I,\ j = 1, 2, \ldots, n, i = 1, 2, \ldots, k_j$, также имеет только целые точки излома из интервала $[0, A]$. Тогда для выполнения шага [2.1.3] необходимо выполнить Процедуру FindMax на не более чем $A + 1$ интервалах. Поэтому трудоёмкость шага [2.1.3] – $O(A)$ операций. В соответствии с леммами 1 и 2 трудоёмкость шагов [2.1.1] и [2.1.2] – $O(F_j.I)$ операций, где $F_j.I \leqslant A$. Трудоёмкость шага [2.2.] – $O(F_j.I)$ операций.

Несложно показать, что функции $F_j(t),\ j = 1, 2, \ldots, n$, невозрастающие. Поэтому,

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

Тогда в соответствие с Процедурой 2.2 имеем не более $2 \cdot |F_j.B|$ различных элементов в множестве $F_j.I$.

Поэтому трудоёмкость GrA –

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

операций. □

Более того, трудоёмкость ограничена величиной $O(\sum k_j \min\{A, F^*\})$, где F^* – оптимальное значение целевой функции, т.к. $\max\limits_{j=1,2,\ldots,n} |F_j.B| \leqslant F^*$.

4. Пример

Проиллюстрируем идею графического алгоритма GrA, используя численный пример, представленный на рис. 2. Полное описание всех вычислений приведено в [10].

Таблица 2. Функции $f_j(t)$

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

Далее представлено краткое описание. В данном случае рассмотрены четыре проекта с функциями прибыли $f_j(t),\ j = 1, 2, 3, 4$ (см. Таблица 2).

Шаг $j = 1,\ k = 1$. В соответствии с процедурой 2.1.1, $\Psi_j^k(x) = 0$, $\Psi_j^k.I = \{0\}$, $\Psi_j^k.U = \{0\}$ и $\Psi_j^k.B = \{0\}$.
Далее опишем каждую итерацию в цикле [2.1.2.5] в Процедуре 2.1.2.
Вначале рассмотрим $s' = 0$. Перед первой итерацией $T' = \emptyset$, $t_{left} = 0$, $t_{right} = 3$, и на шаге [2.1.2.11] $\varepsilon = \min\{25-0, 25-3\} = 22$. На шагах [2.1.2.12. $-$ 14] $b_{left} = 0$, $b_{right} = 0$ и $b_{inner} = 0$. На шагах [2.1.2.15. $-$ 17] $\varphi_{left}(x) = 0$, $\varphi_{right}(x) = 0$, $\varphi_{inner}(x) = 0$ и, как следствие, $\varphi_{max}(x) = 0$. На шаге [2.1.2.24] $s' = s' + 22 = 22$.

Поэтому $\Phi_1^1(x) = \varphi_{max}(x) = 0$ при $x = [0, 22]$ (с предыдущего до текущего значения s').

Далее, рассмотрим $s' = 22$. После шагов [2.1.2.22. $-$ 25] на предыдущей итерации, получаем $T' = \{25\}$, $t_{left} = 22$ и $t_{right} = 25$. Эти значения используются в данной итерации. На шаге [2.1.2.11] $\varepsilon = 25 - 22 = 3$. Затем получаем $b_{left} = 0$, $b_{right} = 0$ и $b_{inner} = 0$. Кроме того, $\varphi_{left}(x) = 0$, $\varphi_{right}(x) = 0$ и $\varphi_{inner}(x) = 0$. Таким образом, $\varphi_{\max}(x) = 0$. Получаем $s' = 22 + 3 = 25$.

Также находим функцию $\Phi_1^1(x) = \varphi_{\max}(x) = 0$ при $x = [22, 25]$, и, следовательно, $\Phi_1^1(x) = 0$, $\Phi_1^1.I = \{0\}$, $\Phi_1^1.U = \{0\}$ и $\Phi_1^1.B = \{0\}$. Отметим, что вместо того чтобы рассматривать примерно 25 значений t в DPA, в данном случае было рассмотрено только два состояния s'. Далее в деталях описано вычисление функций Φ_1^2, Φ_1^3 и Φ_1^4.

Шаг $j = 1,\ k = 2$. Функции имеют значения $\Psi_j^k(x) = x-3$, $\Psi_j^k.I = \{25\}$, $\Psi_j^k.U = \{1\}$ и $\Psi_j^k.B = \{-3\}$.
Рассмотрим $s' = 0$. Имеем $T' = \emptyset$, $t_{left} = 3$, $t_{right} = 10$ и $\varepsilon = \min\{25 - 3, 25 - 10\} = 15$. Кроме того, $b_{left} = 0$, $b_{right} = 7$ и $b_{inner} = 0$. Поэтому $\varphi_{left}(x) = 0 + (1-1)x$, $\varphi_{right}(x) = 7 + (1-1)x$ и $\varphi_{inner}(x) = 0$. Следовательно $\varphi_{\max}(x) = 7$. Получаем $s' = s' + 15 = 15$.

Далее, рассмотрим $s' = 15$. Имеем $T' = \{25\}$, $t_{left} = 15 + 3 = 18$, $t_{right} = 15 + 10 = 25$ и $\varepsilon = 25 - 18 = 7$. Получаем $b_{left} = -3 + 1 \cdot 18 - 1 \cdot 15 = 0$, $b_{right} = 0$ и $b_{inner} = -3 + 1 \cdot (25 - 0) - 1 \cdot 15 = 7$. Функции принимают значения $\varphi_{left}(x) = 0 + (1-1)x$, $\varphi_{right}(x) = 0$ и $\varphi_{inner}(x) = 7 - x$. В результате $\varphi_{max}(x) = 7 - x$ и $s' = s' + 7 = 22$.

Рассмотрим $s' = 22$. Имеем $T' = \emptyset$, $t_{left} = 25$, $t_{right} = 22 + 10 = 32$ и $\varepsilon = A - s' = 25 - 22 = 3$. Получаем $\varphi_{left}(x) = \varphi_{right}(x) = \varphi_{inner}(x) = 0$. Затем $\varphi_{max}(x) = 0$. Следовательно $s' = s' + 3 = 25$.
Были получены $\Phi_1^2.I = \{15, 22, 25\}$, $\Phi_1^2.U = \{0, -1, 0\}$ и $\Phi_1^2.B = \{7, 7, 0\}$.

Шаг $j = 1,\ k = 3$. Имеем $\Psi_j^k(x) = x + 3\frac{2}{3}$, $\Psi_j^k.I = \{25\}$, $\Psi_j^k.U = \{\frac{1}{3}\}$ и $\Psi_j^k.B = \{3\frac{2}{3}\}$. Этот шаг выполняется аналогично предыдущим. Необходимо рассмотреть $s' = 0, 12, 15$.
Получаем $\Phi_1^3.I = \{12, 15, 25\}$, $\Phi_1^3.U = \{0, -\frac{1}{3}, 0\}$ и $\Phi_1^3.B = \{8, 7, 0\}$.

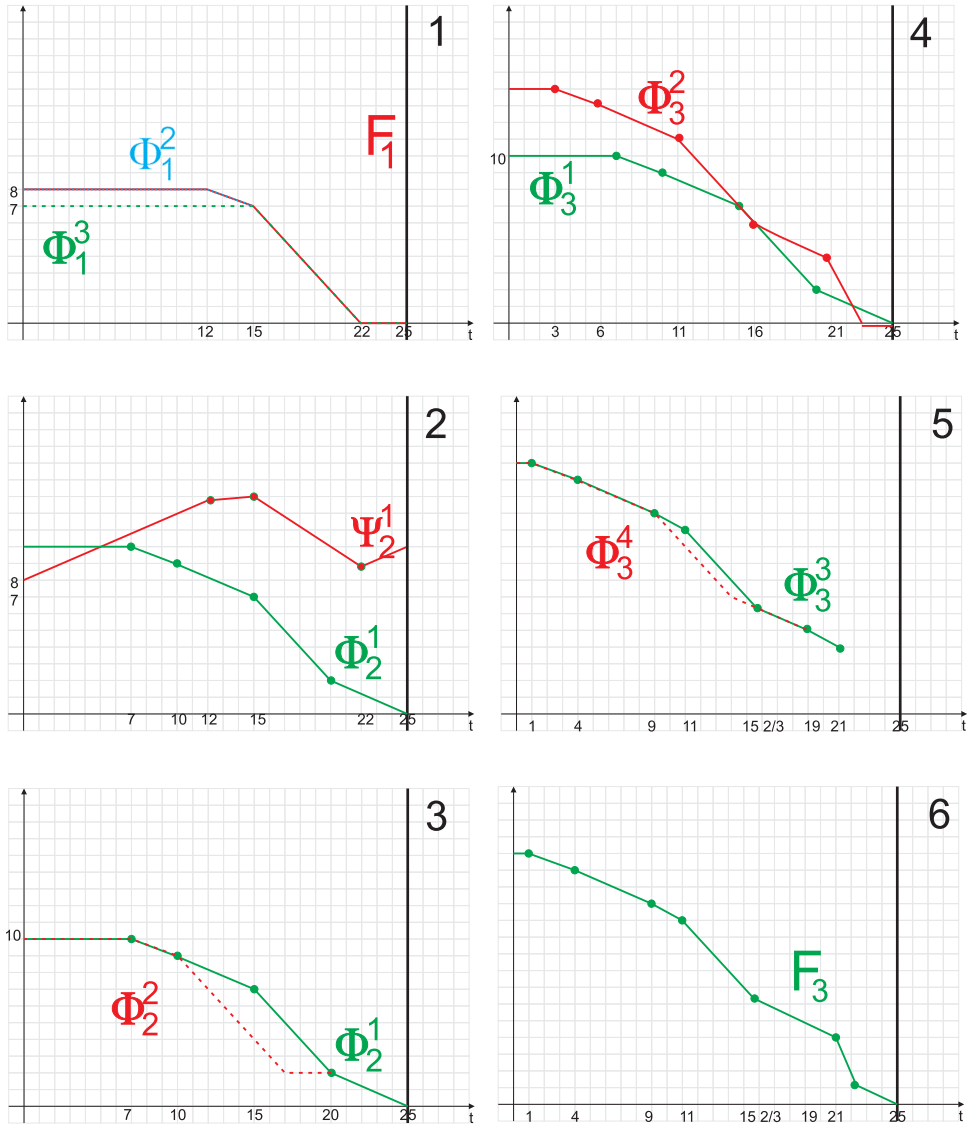

Рис. 3. Вычисления в примере

Шаг $j = 1$, $k = 4$. Имеем $\Psi_j^k(x) = 8$, $\Psi_j^k.I = \{25\}$, $\Psi_j^k.U = \{0\}$ и $\Psi_j^k.B = \{8\}$. Действия те же, что и в предыдущем случае. Необходимо рассмотреть $s' = 0, 12$. Получаем $\Phi_1^4.I = \{12, 25\}$, $\Phi_1^4.U = \{0, 0\}$ и $\Phi_1^4.B = \{8, 0\}$.

Итак, после шага $j = 1$ имеем $F_1(t) = \max\{\Phi_1^1, \Phi_1^2, \Phi_1^3, \Phi_1^4\}$, $F_1.I = \{12, 15, 22, 25\}$, $F_1.U = \{0, -\frac{1}{3}, -1, 0\}$ и $F_1.B = \{8, 8, 7, 0\}$, см. рис. 3.1. Фактически, функция $F_1(t)$ получается только из двух функций Φ_1^2 и Φ_1^3, где Φ_1^2 – максимум функции на интервале $[0, 15]$, и Φ_1^3 – максимум функции на интервале $[15, 25]$.

Далее описаны только рассмотренные состояния и рассчитанные функции на шагах $j = 2, 3, 4$. Как было отмечено ранее, полное описание всех вычислений приведено в [10].

Шаг $j = 2, k = 1$. Рассматриваются состояния $s' = 0, 7, 10, 12, 15, 17, 20, 22$. Имеем $\Phi_2^1.I = \{7, 10, 15, 20, 25\}$, $\Phi_2^1.U = \{0, -\frac{1}{3}, -\frac{2}{5}, -1, -\frac{2}{5}\}$ и $\Phi_2^1.B = \{10, 10, 9, 7, 2\}$, см.

рис. 3.2.

Шаг $j = 2,\ k = 2$. Поскольку $f_2.U[2] = 0$, этот шаг может быть выполнен более простым способом – необходимо сдвинуть диаграмму функции $F_1(t)$ налево на значение 5 и вверх на значение 2. В результате имеем: $\Phi_2^2.I = \{12-5, 15-5, 22-5, 25-5\}$, $\Phi_2^2.U = \{0, -\frac{1}{3}, -1, 0\}$ и $\Phi_2^2.B = \{8+2, 8+2, 7+2, 0+2\}$.
На рис. 3.3 представлен максимум функции. На самом деле, $F_2(t) = \Phi_2^1(t)$, т.е. $F_2.I = \{7, 10, 15, 20, 25\}$, $F_2.U = \{0, -\frac{1}{3}, -\frac{2}{5}, -1, -\frac{2}{5}\}$ и $F_2.B = \{10, 10, 9, 7, 2\}$.

Шаг $j = 3,\ k = 1$. Поскольку $f_3.U[1] = 0$, этот шаг также можно выполнить более простым способом. Для получения функции $\Phi_3^1(t)$ необходимо сдвинуть диаграмму функции $F_2(t)$ налево на значение 0 и вверх на значение 0.

Шаг $j = 3,\ k = 2$. Рассмотрены состояния $s' = 0, 3, 5, 6, 8, 11, 13, 16, 18, 21, 23$. Имеем $\Phi_3^2.I = \{7-4, 10-4, 15-4, 20-4, 25-4, 23, 25\}$, $\Phi_3^2.U = \{0, -\frac{1}{3}, -\frac{2}{5}, -1, -\frac{2}{5}, -2, 0\}$ и $\Phi_3^2.B = \{14, 14, 13, 11, 6, 4, 0\}$.

Шаг $j = 3,\ k = 3$. Рассмотрены состояния $s' = 0, 1, 3, 4, 6, 9, 11, 14, 16, 19, 21$. Имеем $\Phi_3^3.I = \{1, 4, 9, 11, 15\frac{2}{3}, 19, 21, 25\}$, $\Phi_3^3.U = \{0, -\frac{1}{3}, -\frac{2}{5}, -\frac{1}{2}, -1, -\frac{2}{5}, -\frac{1}{2}, 0\}$ и $\Phi_3^3.B = \{15, 14, 12, 11, 6\frac{1}{3}, 5, 0\}$. В данном примере точка $15\frac{2}{3}$ не вырезана, это представлено на рис. 1. В данном случае есть две нецелочисленные точки излома.

Шаг $j = 3,\ k = 4$. Поскольку $f_3.U[4] = 0$, возможно упростить выполнение этого шага. Для получения $\Phi_3^4(t)$ необходимо сдвинуть диаграмму функции $F_2(t)$ налево на величину 6 и вверх на величину 5.
Функции $\Phi_3^1(t)$ и $\Phi_3^2(t)$ изображены на рис. 3.4, а функции $\Phi_3^3(t)$ и $\Phi_3^4(t)$ представлены на рис. 3.5. На рис. 3.6 отмечен максимум функции

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

Получаем $F_3.I = \{1, 4, 9, 11, 15\frac{2}{3}, 21, 22\frac{1}{2}, 25\}$, $F_3.U = \{0, -\frac{1}{3}, -\frac{2}{5}, -\frac{1}{2}, -1, -\frac{2}{5}, -\frac{1}{2}, -\frac{2}{5}\}$ и $F_3.B = \{15, 14, 12, 11, 6\frac{1}{3}, 4, 1\}$.

Шаги $j = 4,\ k = 1, 2, 3$ выполняются тем же упрощенным методом, т.е. для получения функций $\Phi_4^1(t), \Phi_4^2(t)$ и $\Phi_4^3(t)$ необходимо сдвинуть диаграмму функции $F_3(t)$ налево на величину $0, 3, 4$ и вверх на величину $0, 1, 4$. На рис. 4 представлен максимум функции $F_4(t)$.

Для получения оптимального решения в точке $s = 0$ можно использовать обратный ход. Имеем $x_4 = 4$ и $f_4(x_4) = 4$, $x_3 = 6$ и $f_3(x_3) = 5$, $x_2 = 5$ и $f_2(x_2) = 2$, а также $x_1 = 10$ и $f_1(x_1) = 7$. Получаем оптимальное значение целевой функции $F^*(0) = 18$.

В GrA были рассмотрены наборы состояний s' : $2 + 3 + 3 + 2 = 10$ (для $j = 1$), $8+4 = 12$ (при $j = 2$, где 4 состояния были рассмотрены при $k = 2$), $5+10+11+5 = 31$ (при $j = 3$, где 5 состояний были рассмотрены при $k = 1$ и $k = 4$), $7 + 7 + 7 = 21$ (при $j = 4$, т.е. при сдвиге диаграммы). Всего было рассмотрено $10 + 12 + 31 + 21 = 74$ состояний s'. В DPA рассматривается $25(3 + 2 + 4 + 3) = 300$ состояний. Если изменить размер примера, т.е. увеличить все входные данные на M, трудоёмкость DPA вырастет в соответствии с M, а трудоёмкость GrA останется прежней. Следует отметить, что для каждого состояния в GrA необходимо больше вычислений, однако это число постоянно, поэтому GrA имеет меньшее время выполнения.

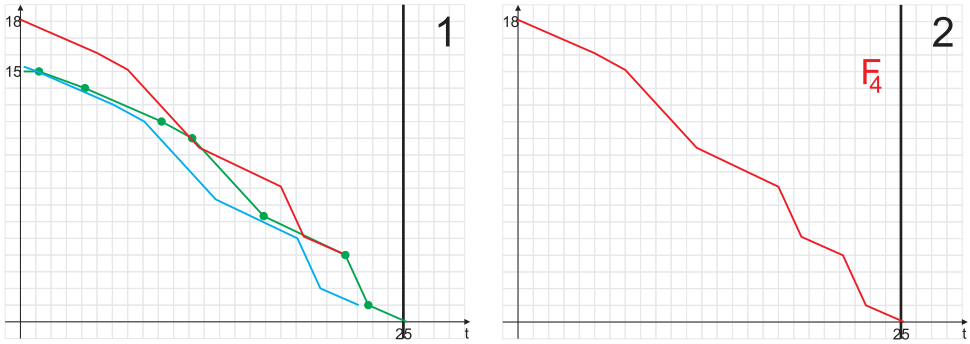

Рис. 4. Функция $F_4(t)$

5. Алгоритм приближенного решения с заданной погрешностью

В этом разделе описан приближенный полиномиальный алгоритм решения с заданной погрешностью, построенный на основе представленного выше GrA.

Для задач минимизации функции $F(\pi)$, полиномиальный алгоритм, который находит решение π' такое, что $F(\pi')$ не более чем в $\rho \geqslant 1$ раз больше $F(\pi^*)$, называется *ρ-аппроксимационным алгоритмом. Значение ρ называется максимальной относительной погрешностью*. Если для задачи существует ρ-аппроксимационный алгоритм, то говорят, что задача аппроксимируема с относительной погрешностью ρ. Набор ρ-аппрокимационных алгоритмов называется полной полиномиальной аппроксимацонной схемой (fully polynomial-time approximation scheme или FPTAS), если $\rho = 1 + \varepsilon$ для любого $\varepsilon > 0$ и трудоёмкость алгоритма полиномиально зависит только от размерности задачи и значения $1/\varepsilon$.

Точный графический алгоритм можно модифицировать во FPTAS.

Данная модификация применима для задач, для которых кусочно-линейная функция Беллмана является неубывающей. Пусть функция Беллмана может быть задана в табличном виде, где $0 = b_j^1 \leqslant b_j^2 \leqslant b_j^3 \leqslant \ldots \leqslant b_j^{m_j+1}$.

Графический алгоритм может быть модифицирован в FPTAS следующим образом. Обозначим через $\delta = \frac{\varepsilon UB}{n}$, где UB – некоторая известная верхняя оценка оптимального значения целевой функции. Пусть также известна верхняя оценка UB такая, что $\frac{UB}{LB}$ не превышает некоторую константу c. Чтобы сократить трудоёмкость GrA, необходимо сократить количество колонок в таблицах $F_j(t)$. Причем количество этих колонок равно количеству различных значений $0 = b_j^1 \leqslant b_j^2 \leqslant b_j^3 \leqslant \ldots \leqslant b_j^{m_j+1}$ в этих колонках. Причем $b_j^{m_j+1} \leqslant UB$.

В таблице $F_j(t)$ мы будем хранить не оригинальные значения b_j^k, но значения $\overline{b_j^k}$ являющиеся ближайшим к b_j^k значениями, делящимися без остатка на δ. Существует не более чем $\frac{UB}{\delta} = \frac{c \cdot n}{\varepsilon}$ различных величин $\overline{b_j^k}$. Тогда можно будет преобразовать таблицу функции $F_j(t)$ в таблицу приближённой функции с не более чем $2\frac{c \cdot n}{\varepsilon}$ колонками (см. рис.5). Причем для данной модифицированной функции $F'_j(t)$ выполняется $|F_j(t) - F'_j(t)| < \delta \leqslant \frac{\varepsilon F(\pi^*)}{n}$. Если мы будем выполнять подобную аппроксимацию после каждой стадии GrA, то накопленная ошибка не превысит значения $n\delta \leqslant \varepsilon F(\pi^*)$, и трудоёмкость приближённого GrA будет $O(\frac{n^2}{\varepsilon})$, если трудоёмкость исходного GrA равна $O(nA)$.

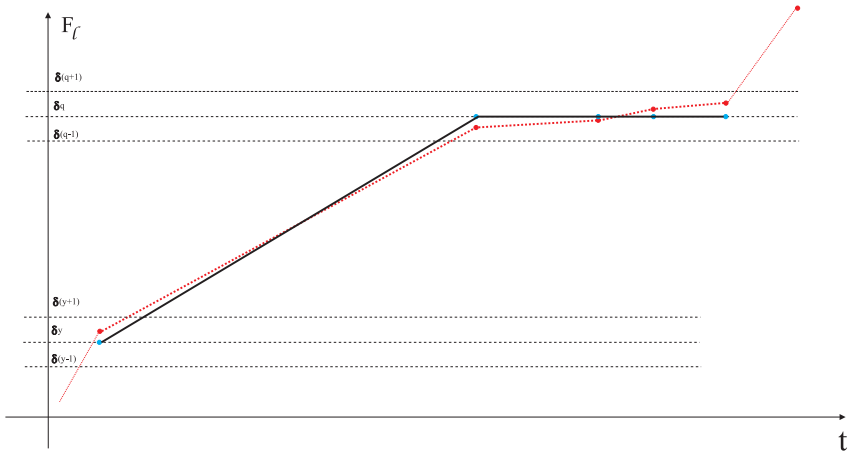

Рис. 5. Замена колонок и модификация $F_l(t)$

Пусть $LB = \max\limits_{j=1,\ldots,n} f_j(A)$ и $UB = n \cdot LB$ – нижняя и верхняя оценки оптимального значения целевой функции.

Примем $\delta = \frac{\varepsilon LB}{n}$. Трудоёмкость FPTAS – $O\left(\frac{n^2 \sum k_j}{\varepsilon}\right)$, операций.

В работе [12] предложена следующая техника по минимизации трудоемкость алгоритмов приближенного решения. Если для задачи существует FPTAS с трудоемкостью $P(L, \frac{1}{\varepsilon}, \frac{UB}{LB})$, где L – длина входа, UB, LB – известные верхняя и нижняя оценки, и значение $\frac{UB}{LB}$ не ограничено некоторой константой, тогда техника позволяет за $P(L, \log\log\frac{UB}{LB})$ операций найти такие значения UB_0 и LB_0, что $LB_0 \leqslant F^* \leqslant UB_0 < 3LB_0$, т.е. такие верхнею и нижнею оценки, что отношение $\frac{UB_0}{LB_0}$ не превышает константу 3. С использованием таких значений UB_0 и LB_0 трудоёмкость FPTAS может быть сокращена до $P(L, \frac{1}{\varepsilon})$ операций, где P – тот же полином.

С помощью данной техники трудоёмкость может быть сокращена до $O\left(\frac{n \cdot \sum k_j}{\varepsilon}(1 + \log\log n)\right)$ операций.

Для одноприборных задач теории расписаний FPTAS, основанная на GrA, представлена в [13].

6. Выводы

В данной работе представлены алгоритмы точного и приближённого решения задачи об инвестициях, трудоёмкость которых – минимальная среди других известных алгоритмов точного или приближённого решения.

С помощью данного метода можно эффективно решать задачи, для которых может быть построен псевдо-полиномиальный алгоритм решения, основанный на принципе оптимальности Беллмана. Для многих известных задач комбинаторной оптимизации (задача о Ранце, задача об инвестициях) и, в частности, задач теории расписаний с одним прибором графический метод позволяет существенно сократить трудоёмкость по сравнению с классическими алгоритмами динамического программирования. На основе графического метода также можно создавать аппроксимационные схемы с лучшей трудоёмкостью среди известных аппроксимационных схем.

Результаты экспериментальных исследований также свидетельствуют об эффективности метода. Таким образом, графический метод имеет как теоретическое, так и практическое значение.

СПИСОК ЛИТЕРАТУРЫ

1. *Posypkin M.A., Sigal I.Kh.* Speedup estimates for some variants of the parallel implementations of the branch-and-bound method // J. Math. Math. Physics. 2006. P. 46 No 12. P. 2189-–2202.

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

3. *Bar-Noy A., Golin M.J., Zhang Y.* Online Dynamic Programming Speedups // J. Theory Comput. Syst. 2009. V. 45. No 3. P. 429–445.

4. *Eppstein D., Galil Z., Giancarlo R.* Speeding up Dynamic Programming // In Proc. 29th Symp. Found. Comput. Sci. 1988.

5. *Wagelmans A.P.M., Gerodimos A.E.* Improved dynamic programs for some batching problems involving the maximum lateness criterion
Oper. Res. Lett. 2000. V. 27. P. 109–118.

6. *Kellerer H., Pferschy U., Pisinger D.* Knapsack Problems. Springer-Verlag, Berlin. 2004.

7. *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 // Management Sci. 1998. V. 44 No 6. P. 831–838.

8. *Kameshwaran S., Narahari Y.* Nonconvex Piecewise Linear Knapsack Problems // Eur. J. Oper. Res. 2009. V. 192. P. 56–68.

9. *Schemeleva K., Delorme X., Dolgui A., Grimaud F., Kovalyov M.Y.* Lot-sizing on a single imperfect machine: ILP models and FPTAS extensions // J. Comput. Indust. Engin. 2013. V. 65. No 4. P. 561–569.

10. *Gafarov E.R., Dolgui A., Lazarev A.A., Werner F.* A Graphical Approach to Solve an Investment Optimization Problem // J. Math Model Algor. 2014. V. 13. No 4. P. 597–614.

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

12. *Kovalyov M.Y.* Improving the complexities of approximation algorithms for optimization problems // Operations Research Letters 1995. V. 17. P. 85–87.

13. *Gafarov E.R., Dolgui A., Werner F.* A Graphical Approach for Solving Single Machine Scheduling Problems Approximately // Int. J. Prod. Res. 2014. http://dx.doi.org/10.1080/00207543.2014.922708.

Гафаров Е.Р., *Институт проблем управления им. В.А.Трапезникова РАН, старший научный сотрудник, канд. физ.-мат. наук, Москва*, axel73@mail.ru

Долгий А., *Высшая национальная горная школа Нанси, док. физ.-мат. наук, Нанси, Франция*, alexandre.dolgui@mines-nantes.fr

Лазарев А.А., *Институт проблем управления им. В.А.Трапезникова РАН, заведующий лабораторией, д-р физ.-мат. наук, Москва; Московский государственный университет им. М.В. Ломоносова; Московский физико-технический институт (Государственный университет); Высшая школа экономики (Национальный исследовательский университет), Москва*, jobmath@mail.ru

Вернер Ф., *Факультет математики университета Отто фон Герике, д-р философии, Магдебург, Германия*, Frank.Werner@Mathematik.Uni-Magdeburg.DE