© 2021 г. В.П. КОРНЕЕНКО, канд. техн. наук (vkorn@ipu.ru)
(Институт проблем управления им. В.А. Трапезникова РАН, Москва)

ЭФФЕКТИВНЫЙ АЛГОРИТМ ТУПИКОВЫХ УПРАВЛЕНИЙ ДЛЯ РЕШЕНИЯ ЗАДАЧ КОМБИНАТОРНОЙ ОПТИМИЗАЦИИ

Предлагается алгоритм тупиковых управлений, предназначенный для точного решения *NP*-трудных задач комбинаторной оптимизации. Эффективность алгоритма демонстрируется на примерах решения задачи разбиения на равные части и задачи об одномерном рюкзаке. В статье также показано, что применение идеи тупиковых управлений при реализации метода динамического программирования позволяет значительно сократить на каждом шаге оптимизации число переменных состояний задачи. Проведен сравнительный анализ предлагаемого метода с известными алгоритмами решения этих задач.

Ключевые слова: тупиковое управление, функция Беллмана, алгоритм, задача разбиения, задача о рюкзаке.

DOI: 10.31857/S000523102110007X

1. Введение

В настоящее время на практике получили распространение два основных оптимальных метода решения задач комбинаторной оптимизации, к которым относится задача об одномерном рюкзаке (0-1 knapsack problem) и сводящаяся к ней задача разбиения на равные части (set-partition problem), а именно: метод ветвей и границ в рамках статичной модели, когда параметры задачи не меняются во времени, и различные модификации метода динамического программирования [1–3]. Подробный обзор различных методов и алгоритмов, разработанных для решения задачи о рюкзаке, изложены в [4].

Наряду с модификациями метода динамического программирования для решения задачи о рюкзаке применяются и приближенные алгоритмы, в частности жадный алгоритм и аппроксимационный алгоритм, подробно изложенные в [2, с. 448–478; 5, с. 417–424] соответственно.

Комбинированные эвристические алгоритмы для задачи о рюкзаке подробно изложены в [6]. В [7–9] представлен графический подход к решению задачи разбиения на равные части и задачи об одномерном рюкзаке.

Несмотря на то что данные задачи относятся к классу *NP*-трудных, в последнее время в большом количестве научных публикаций появляются новые алгоритмы для задачи о ранце в виде как точных алгоритмов, так и приближенных, включая и эвристические алгоритмы, обладающие различной временно́й трудоемкостью [10–12].

Рассмотрим математические постановки задач комбинаторной оптимизации.

1. Задача разбиения на равные части (partition) состоит в следующем: задано множество целых чисел $B = \{b_1, \ldots, b_n\}$. Требуется разбить множество B на два непересекающихся подмножества B_1 и B_2 так, чтобы минимизировать значение:

$$\left| \sum_{b_j \in B_1} b_j - \sum_{b_j \in B_2} b_j \right| \to \min . \tag{1}$$

2. Задача об одномерном рюкзаке (0-1 knapsack). В общем виде вербальная постановка задачи сводится к следующему: из заданного множества предметов, характеризующихся для j-го предмета "ценностью" p_j и "весом (объемом)" w_j, требуется отобрать такое число предметов, чтобы получить максимальную суммарную ценность при одновременном соблюдении ограничения b на суммарный вес или объем.

Математическую постановку целочисленной задачи представим в виде задачи булевого линейного программирования:

$$\begin{cases} f(x_1, \ldots, x_n) = \sum_{j=1}^{n} p_j x_j \to \max\limits_{x_1, \ldots, x_n}, \\ \sum_{j=1}^{n} w_j x_j \le b, \quad x_j \in \{0, 1\}, \quad j = 1, 2, \ldots, n. \end{cases} \tag{2}$$

Когда $p_j = w_j = b,\ j = \overline{1, n}$, и $b = \frac{1}{2}\sum_{j=1}^{n} b_j$, то задачи (1) и (2) являются эквивалентными.

Идея тупиковых управлений взята из задачи минимизации и получения сокращенной (тупиковой) дизъюнктивной нормальной формы (ДНФ) [13], представляющей собой произвольную дизъюнкцию элементарных конъюнкций логических функций булевой алгебры, которую нельзя упростить. По аналогии с ДНФ допустимое управление, в котором замена произвольной нулевой компоненты в булевском векторе на единицу приводит к нарушению ресурсных ограничений, в задаче о ранце будем называть тупиковым.

В данной статье предлагается новый оптимальный алгоритм тупиковых управлений, который по своей эффективности на данный момент превосходит опубликованные алгоритмы для решения задачи разбиения на равные части и задачи об одномерном рюкзаке.

Кроме того, предлагаемый алгоритм может быть применим в задачах выполнения комплекса взаимосвязанных работ, математическая постановка которых представлена на динамических управляемых моделях [14]. В статье демонстрируется применение тупиковых управлений при решении задачи о рюкзаке методом динамического программирования.

2. Алгоритм тупиковых управлений

Рассмотрим шаги *алгоритма* тупиковых управлений более подробно.

Шаг 1. Упорядочение номеров предметов в порядке убывания весов. Не умаляя общности, будем предполагать, что для любого предмета справедливо неравенство:

$$w_j \le b \quad \forall j = \overline{1,n}.$$

Построение тупиковых управлений начинается с упорядочения номеров предметов в порядке убывания весов (объемов):

$$w_{j_1} \ge \ldots \ge w_{j_k} \ge \ldots \ge w_{j_n}.$$

Пусть $u_k \equiv x_{j_k} \in \{0,1\}$ – переменная управления, принимающая значение единица, если k-й предмет по порядку помещается в рюкзак, и нулевое значение в противном случае. В связи с перенумерацией переменных математическая постановка задачи о рюкзаке сводится к нахождению такого вектора управления $\vec{u} = (u_1, \ldots, u_n)$, который доставляет максимум целевой функции:

$$f(\vec{u}) = \sum_{k=1}^{n} p_k u_k \to \max_{u_1,\ldots,\,u_n} \tag{3}$$

при условии

$$\sum_{k=1}^{n} w_k u_k \le b, \quad u_k \in \{0,1\}, \quad k = 1,2,\ldots,n. \tag{4}$$

Очевидно, что задача (3)–(4) эквивалентна задаче (2).

Шаг 2. Построение тупиковых управлений. В первое тупиковое управление включаем первый предмет, который соответствует управлению $u_1 = 1$, если он не нарушает ограничения (4), т.е. выполняется неравенство: $w_1 u_1 \le b$, в противном случае полагаем $u_1 = 0$. Точно так же поступаем со вторым, третьим и так далее по порядку и с n-м предметом в соответствии с формулой:

$$u_k = \begin{cases} 1, & \text{если} \quad \sum\limits_{i=1}^{k} u_i w_i \le b; \\ 0, & \text{если} \quad \sum\limits_{i=1}^{k} u_i w_i > b, \end{cases} \tag{5}$$

последовательно для $k = 1,2,\ldots,n$.

В результате получим первое тупиковое управление, состоящее из нулей и единиц

$$\vec{u}_1 = \left(u_1^{(1)}, u_2^{(1)}, \ldots, u_n^{(1)}\right). \tag{6}$$

Построенному вектору управления $\vec{u}_1$ (6) соответствует некоторое двоичное число

$$\xi_1 = (11010..1..01001)\,, \tag{7}$$

где единицы стоят в тех разрядах, номера которых совпадают с номерами предметов, включенных в управление $\vec{u}_1$ (6). Замена любого нуля единицей делает это управление недопустимым по ограничению (4).

С помощью первого тупикового управления построим второе. Для этого найдем самый младший разряд числа ξ_1, в котором записан ноль. Во всех разрядах справа от него вместо единиц записываем нули. В полученном двоичном числе первую справа единицу перенесем на один разряд вправо. Если полученное управление недопустимо по ограничению (4), то эту единицу сдвигаем еще на один разряд вправо до тех пор, пока управление не окажется допустимым.

Далее в разряды справа от этой единицы помещаем единицы по тому же правилу (5), что и при построении числа ξ_1 (7). В результате получаем двоичное число $\xi_2 < \xi_1$. Этому числу соответствует тупиковое управление $\vec{u}_2 = \left(u_1^{(2)}, u_2^{(2)}, \ldots, u_n^{(2)}\right)$. Точно таким же образом из тупикового управления $\vec{u}_2$ строится тупиковое управление $\vec{u}_3$, которому соответствует двоичное число $\xi_3 < \xi_2 < \xi_1$ и т.д. В результате получаем множество тупиковых управлений

$$U = \left\{ \vec{u}_l = \left(u_1^{(l)}, \ldots, u_k^{(l)}, \ldots, u_n^{(l)}\right) \mid l = 1, 2, \ldots, N \right\}, \tag{8}$$

которому соответствует упорядочение двоичных чисел:

$$\xi_N < \xi_{N-1} < \ldots < \xi_3 < \xi_2 < \xi_1.$$

Описанная процедура дает возможность получить все тупиковые управления, удовлетворяющие ограничению (4).

Шаг 3. Вычисление оптимального тупикового управления. Для каждого тупикового управления $\vec{u}_l$, $l = \overline{1, N}$, находим значение целевой функцией $f(\vec{u}_l)$ (3).

За оптимальное управление принимаем тупиковое $\vec{u}_* \in U$ (8), обеспечивающее максимальное значение целевой функции $f(\vec{u})$ (3):

$$\vec{u}_* = (u_1^*, \ldots, u_k^*, \ldots, u_n^*) = \arg\max_{\vec{u}_l \in U} f\left(u_1^{(l)}, \ldots, u_k^{(l)}, \ldots, u_n^{(l)}\right).$$

Обоснование изложенного алгоритма решения задачи о рюкзаке базируется на следующих теоремах.

Теорема 1 (о существовании оптимального решения). *Среди всех тупиковых управлений из множества U* (8) *найдется по крайней мере одно тупиковое управление, обеспечивающее максимум целевой функции задачи о рюкзаке* (3)–(4).

Доказательство. Пусть $\vec{u}_*$ – оптимальное тупиковое управление, являющееся решением задачи (3)–(4). Предположим обратное, что $\vec{u}_*$ не является оптимальным и тупиковым управлением, т.е. $\vec{u}_* \notin U$ (8). Дополним его до тупикового $\vec{u}_0 \in U$ (8).

Это значит, что найдется предмет, который войдет в рюкзак, и значение целевой функции при этом увеличится на величину ценности этого предмета. При этом для целевой функции $f(\vec{u})$ (3) будет выполняться неравенство, т.е.

$$f(\vec{u}_0) > f(\vec{u}_*).$$

Получили противоречие с тем, что нашлось решение $\vec{u}_0$ лучшее, чем $\vec{u}_*$. Следовательно, предположение $\vec{u}_*$ не верно, что и требовалось доказать. Теорема 1 доказана.

От исходной прямой задачи (3)–(4) максимизации ценности рюкзака перейдем к двойственной задаче минимизации остатка веса (объема) рюкзака по критерию:

$$g(\vec{u}_l) = b - \sum_{k=1}^{n} w_k u_k^{(l)} \to \min_{\vec{u}_l \in U}, \tag{9}$$

который равносилен максимизации суммы весов предметов, помещаемых в рюкзак.

Очевидно, что для функции $g(\vec{u}_l)$ (9) должно выполняться неравенство

$$g(\vec{u}_l) = b - \sum_{k=1}^{n} w_k u_k^{(l)} \geq 0 \quad \forall\, l = 1, 2, \ldots, N, \tag{10}$$

где b – объем (вес) рюкзака, w_k – вес k-го предмета управления $\vec{u}_l \in U$ (8).

При этом имеет место следующее утверждение.

Теорема 2 (о связи прямой и двойственной задач). *Пусть*

$$\vec{u}_f = \left(u_1^{(f)}, \ldots, u_k^{(f)}, \ldots, u_{j_n}^{(f)}\right) \quad u \quad \vec{u}_g = \left(u_1^{(g)}, \ldots, u_k^{(g)}, \ldots, u_{j_n}^{(g)}\right)$$

— оптимальные решения прямой $f(\vec{u}_l)$ (3)–(4) и двойственной $g(\vec{u}_l)$ (9)–(10) задач соответственно.

Тогда для весов предметов, помещаемых в рюкзак, справедливо неравенство

$$\sum_{k=1}^{n} w_k u_k^{(f)} \leq \sum_{k=1}^{n} w_k u_k^{(g)}. \tag{11}$$

Доказательство. Пусть

$$U_{\min} = \left\{ \vec{u}_g \in U\,(8) \mid \vec{u}_g = \arg\min_{\vec{u}_l \in U} g(\vec{u}_l) \right\} \tag{12}$$

— подмножество множества U (8) тупиковых управлений, на которых критерий $g(\vec{u}_l)$ (9) достигает минимального значения, которое не всегда совпадает с точной нижней гранью U (8), а именно: $\inf g(\vec{u}_l) = 0$, $\vec{u}_l \in U$ (8).

Очевидно, что если оптимальные решения прямой $f(\vec{u}_l)$ (3)–(4) задачи $\vec{u}_f \in U_{\min}$ (12), то неравенство (11) выполняется как равенство.

Покажем, что если $\vec{u}_f \notin U_{\min}$ (12), то неравенство (11) выполняется как строгое. Действительно, в соответствии с определением минимума целевой функции [5] должно выполняться неравенство

$$g(\vec{u}_g) < g(\vec{u}_f) \ \forall \ \vec{u}_f \notin U_{\min} \quad (12),$$

т.е.

$$b - \sum_{k=1}^{n} w_k u_k^{(g)} < b - \sum_{k=1}^{n} w_k u_k^{(f)} \Rightarrow \sum_{k=1}^{n} w_k u_k^{(g)} > \sum_{k=1}^{n} w_k u_k^{(f)}.$$

Отсюда следует справедливость неравенства (11), что и требовалось доказать. Теорема 2 доказана.

3. Метод динамического программирования с тупиковыми управлениями на примере решения задачи о рюкзаке в прямом времени

Используя основную идею динамического программирования [1], сведем решение задачи (3)–(4), поставленной на статичной математической модели, к задаче оптимизации управляемой динамической системы, решение которой сводится к следующим этапам.

1. Этап инвариантного погружения. Для этого вложим задачу о рюкзаке в семейство задач той же природы, в результате чего получим управляемую систему в прямом времени:

$$Z = \left\{ Z_k : \max_{u_1,\ldots,u_k} \sum_{j=1}^{k} p_j u_j, \quad \sum_{j=1}^{k} w_j u_j \le s_k, \quad k = \overline{1,n} \right\}, \tag{13}$$

где $u_k \in \{0,1\}$ – управление на k-м шаге оптимизации, s_k – переменная состояния, характеризующая остаточный вес рюкзака.

Множество допустимых значений переменной состояния s_k управляемой системы (13) для u_k-го управления будем обозначать через S_k, являющееся подмножеством множества числовых значений параметров $S = \{0, 1, \ldots, b\}$, связанным с семейством задач (13). Исходная задача очевидным образом входит в рассматриваемое семейство, если в (13) положить $k = n$ и $s_n = b$. Поскольку из семейства задач выделяется исходная задача, то семейство задач Z (13) реализует принцип инвариантного погружения в прямом времени.

Пусть на первом шаге $k = 1$ осуществляется выбор переменной управления u_1 при некотором выборе переменных $u_2, u_3, \ldots, u_n$ таком, что

$$s_1 = b - \sum_{j=2}^{n} w_j u_j,$$

где $0 \le s_1 \le b$.

Величина s_1 характеризует тот остаток общего ресурса b, который можно использовать при выборе u_1. Перейдя ко второму шагу, а затем и к третьему и далее к k-му шагу, будем рассматривать остаток общего ресурса b на k-м шаге как s_k состояние процесса выбора управления u_k управляемой системы (13).

Из ресурсных ограничений

$$0 \le \sum_{j=1}^{k} w_j u_j \le s_k, \quad k = \overline{1,n}, \tag{14}$$

имеем:

а) параметры состояния управляемой системы на k-м шаге оптимизации

$$S_k = s_k = \left\{ b - \sum_{j=k+1}^{n} w_j u_j \,|u_j \in \{0,1\}, \;\; k = 1,\dots,n \right\}, \tag{15}$$

где $s_0 = 0$ — начальное состояние на первом шаге $k = 1$,

$s_n = b$ — конечное состояние на последнем шаге $k = n$;

б) уравнения состояний в прямом времени

$$s_{k-1} = s_k - w_k u_k,$$

связывающие переменные состояний функций Р. Беллмана на шагах оптимизации $k-1$ и k-м.

Множества значений функции Беллмана на k-м шаге оптимизации представим в виде

$$S_k = \{\underline{s}_k, \underline{s}_k + 1, \dots, b\}, \tag{16}$$

где минимальное значение переменной состояния $s_k \in S_k$ (15) определяется из условия:

$$\underline{s}_k = \min_{u_{k+1},\dots,u_n} \left\{ b - \sum_{j=k+1}^{n} w_j u_j \right\}, \tag{17}$$

причем с учетом неравенства (14) переменные управления $u_{k+1}, \dots, u_n$ должны удовлетворять соотношению (5).

Поскольку для любых переменных $u_{k+1}, \dots, u_n$ и не обязательно тупиковых на k-м шаге справедливо неравенство $\sum_{j=k+1}^{n} w_j u_j \le \sum_{j=k+1}^{n} w_j$, то минимальное значение переменной состояния $s_k \in S_k$ (15) для задач большой размерности можно определять и из условия:

$$\underline{s}_k = \min_{u_{k+1},\dots,u_n} \left\{ 0, \left| b - \sum_{j=k+1}^{n} w_j \right| \right\}.$$

Множество допустимых управлений k-го шага представим в виде:

$$U(s_k) = \left\{ u_k \in \{0,1\} \middle| \, 0 \le u_k \le \left[\frac{s_k}{w_k} \right] \right\},$$

где $\left[\frac{s_k}{w_k} \right]$– целая часть числа $\frac{s_k}{w_k}$.

Отсюда под действием управления u_k система, находящаяся в состоянии s_{k-1}, перейдет в состояние s_k. Показатель эффективности k-го шага определим как $f_k = p_k u_k$.

Заключаем, что задача (3)–(4) поставлена как задача динамического программирования оптимизации управляемой системы.

2. Этап построения рекуррентных функциональных уравнения Р. Беллмана. На решениях задач Z (13) определим функцию Беллмана от k переменных управления $u_1, \ldots, u_k$ в виде

$$B_k(s_k) = \max_{u_1,\ldots,u_k} \sum_{j=1}^{k} p_j u_j, \quad k = 1,\ 2, \ldots, n, \tag{18}$$

с областью определения S_k (16), характеризующую суммарную ценность рюкзака от первогого шага до k-го шага.

Так как вычисление последовательности функций Беллмана $B_k(s_k)$ (18) происходит в направлении возрастания дискретного аргумента k (идет слева направо: 1, 2, ...), то для k-го шага имеем рекуррентное уравнение Беллмана в прямом времени в виде:

$$B_k\left(s_k\right) = \max_{u_k \le \left[\frac{s_k}{w_k} \right]} \left\{ p_k u_k + B_{k-1}\left(s_k - w_k u_k\right) \right\}, \tag{19}$$

которое удовлетворяет начальному условию

$$B_1\left(s_k\right) = \max_{0 \le u_1 \le \left[\frac{s_1}{w_1} \right]} p_1 u_1.$$

Выбрав на k-м шаге некоторое произвольное управление u_k, система из состояния s_{k-1} придет в состояние s_k. Так как в оптимальном решении задач Z (13) должно быть либо 0, либо 1, то уравнения (19) запишутся в виде

$$B_k\left(s_k\right) = \max_{0 \le u_k \le \left[\frac{s_k}{a_k} \right]} \left\{ \begin{array}{ll} p_k, & u_k = 1 \\ B_{k-1}\left(s_{k-1}\right), & u_k = 0 \end{array} \right\}, \qquad k = 1, 2, \ldots, n.$$

Таким образом, для любого значения $s_k \in S_k$ определение величины $B_k(s_k)$ сводится к простейшей задаче оптимизации — сравнению двух чисел, начальное условие при этом для начального шага $k = 1$ запишется в виде:

$$B_1\left(s_1\right) = \max_{0 \le u_1 \le \left[\frac{s_1}{a_1} \right]} \left\{ \begin{array}{ll} p_1, & u_1 = 1 \\ 0, & u_1 = 0 \end{array} \right\}.$$

Дойдя до $k = n$, определим оптимальное значение целевой функции $B_n(s_n)$, совпадающей со значением целевой функции исходной задачи.

3. Этап решения рекуррентных функциональных уравнений. На данном этапе алгоритмом обратной прогонки, дойдя до $k = n$ шага, определим оптимальное значение функции $B_n(\overline{s}_n)$, $\overline{s}_n \in S_n$, совпадающей со значением целевой функции исходной задачи (3)–(4). Из условия $B_n(\overline{s}_n) = f_n(u_n^*) + B_{n-1}(\overline{s}_n - w_n u_n^*)$ имеем оптимальное управление $u_n^* = u_n^*(\overline{s}_n)$.

Далее последовательно на каждом шаге для $k = n, n-1, \ldots, 1$, определяем оптимальные управления:

$$B_n(\overline{s}_n) \to u_n^* = u_n^*(\overline{s}_n) \to \overline{s}_{n-1} = \overline{s}_n - w_n u_n^*;$$
$$\ldots$$
$$B_k(\overline{s}_k) \to u_k^* = u_k^*(\overline{s}_k) \to \overline{s}_{k-1} = \overline{s}_k - w_k u_k^*;$$
$$\ldots$$
$$B_1(\overline{s}_1) \to u_1^* = u_1^*(\overline{s}_1).$$

Тогда в конце работы пошаговой процедуры получим оптимальное управление

$$\vec{u}_* = (u_1^*, \ldots, u_n^*). \tag{20}$$

Необходимо заметить, что если на каждом шаге запоминать вектор управления вида $\vec{u}(k) = (u_1, \ldots, u_k)$, $k = \overline{1,n}$, то на последнем шаге сразу можно выделить оптимальное решение $\vec{u}^*$ (20).

4. Сравнительная оценка эффективности алгоритма

Для оценки эффективности алгоритма сравним его трудоемкость с традиционными методами решения задачи о рюкзаке. Продемонстрируем работу алгоритма на примерах решения задачи разбиения и задачи о рюкзаке.

Пример 1. На данных примера из [9, с. 320–323] решим задачу разбиения. Пусть числа множества $B = \{100, 70, 50, 20\}$, пронумерованы по невозрастанию, $n = 4$. Сведем задачу разбиения (1) к эквивалентной задаче о рюкзаке (3), что можно представить в виде

$$f(u_1, u_2, u_3, u_4) = 100u_1 + 70u_2 + 50u_3 + 20u_4 \to \max_{u_1,\ldots,u_4},$$
$$100u_1 + 70u_2 + 50u_3 + 20u_4 \le 120,$$

где $b = \frac{1}{2}\sum_{j=1}^{4} b_j = 120$.

Выполнив шаги 2 и 3 алгоритма, тупиковые управления и соответствующие им значения целевой функции представим в виде табл. 1.

В результате, построив четыре тупиковых управления, получаем два оптимальных решения:

$$B_1 = \{b_1, b_4\} = \{100, 20\},\ B_2 = \{b_2, b_3\} = \{70, 50\};$$
$$B_1 = \{b_2, b_3\} = \{70, 50\},\ B_2 = \{b_1, b_4\} = \{100, 20\}.$$

Таблица 1

l	U	u_1	u_2	u_3	u_4	$f(\vec{u}_l)$
1	$\vec{u}_1$	1	0	0	1	120
2	$\vec{u}_2$	0	1	1	0	120
3	$\vec{u}_3$	0	1	0	1	70
4	$\vec{u}_4$	0	0	1	1	70

Таблица 2

l	U	u_1	u_2	u_3	u_4	$f(\vec{u}_l)$
1	$\vec{u}_1$	1	0	0	1	8
2	$\vec{u}_2$	0	1	1	0	13
3	$\vec{u}_3$	0	1	0	1	11
4	$\vec{u}_4$	0	0	1	1	12

Как видно из табл. 1, понадобилось всего лишь четыре тупиковых управления, чтобы найти точное оптимальное решение. Для решения данной задачи разбиения графическим алгоритмом понадобилось рассмотреть семь точек (см. [9, с. 322]), а для алгоритма тупиковых управлений потребовалось построить всего лишь четыре управления (см. табл. 1).

Пример 2. Продемонстрируем работу алгоритма при решении задачи о рюкзаке на данных примера из [9, с. 326–333].

Постановку исходной задачи представим с учетом убывания весов предметов в виде

$$\begin{cases} f(\vec{u}) = 3u_1 + 6u_2 + 7u_3 + 5u_4 \to \max\limits_{u_1,\ldots,u_4}, \\ 7u_1 + 5u_2 + 3u_3 + 2u_4 \le 9, \\ u_j \in \{0,1\}, \quad j = 1,2,3,4. \end{cases}$$

Выполнив шаги 2 и 3 алгоритма, тупиковые управления и соответствующие им значения целевой функции представим в виде табл. 2.

Построив четыре тупиковых управления, получаем оптимальное решение:

$$\vec{u}_* = \vec{u}_2 = (0,1,1,0),$$

где

$$u_1 = x_4 = 0, \quad u_2 = x_3 = 1, \quad u_3 = x_2 = 1, \quad u_4 = x_1 = 0.$$

Для решения данной задачи о рюкзаке графический алгоритм вычисляет только 14 элементов [9, с. 333], в то же время для алгоритма тупиковых управлений потребовалось построить всего лишь четыре управления (см. табл. 2).

Пример 3. Рассмотрим задачу о рюкзаке вместимостью $b = 10$ для множества из 7 предметов, т.е. $J_0 = \{1,2,3,4,5,6,7\}$, вес и стоимость которых представлены в табл. 3 (исходные данные взяты из [15, с. 437]).

Таблица 3. Исходные данные задачи о рюкзаке

j	1	2	3	4	5	6	7
w_j	4	1	2	3	2	1	2
p_j	299	73	159	221	137	89	157

Таблица 4. Результаты решения методом тупиковых управлений

l	$\vec{u}_l = \left(u_1^{(l)}, \ldots, u_7^{(l)}\right)$	ξ_l – значение в десятичной системе счисления	$\sum\limits_{k=1}^{n} w_k u_k$	$f(\vec{u}_l)$
1	1110010	114	10	752
2	1110001	113	10	768
3	1101010	106	10	730
4	1101001	105	10	746
5	1100110	102	10	750
6	1100101	101	10	766
7	1100011	99	9	682
8	1011100	92	10	752
9	1011011	91	10	757
10	**1010111**	**87**	**10**	**777**
11	1001111	79	10	755
12	0111110	62	10	747
13	0111101	61	10	763
14	0111011	59	9	679
15	0110111	55	9	699
16	0101111	47	9	677
17	0011111	31	8	615

1. Решение псевдополиномиальным алгоритмом динамического программирования ДП-III. Результатом решения является подмножество

$$J_* = \{1, 2, 3, 6, 7\}, \tag{21}$$

где $p = \sum_{j \in\ J_*} p_j = 777$.

При этом алгоритм проходит через построение 91 пары (J, p), где $J \subset J_0$. Трудоемкость данного алгоритма $O\left(n^2 p\right)$, где p – значение оптимальной стоимости [15, с. 436].

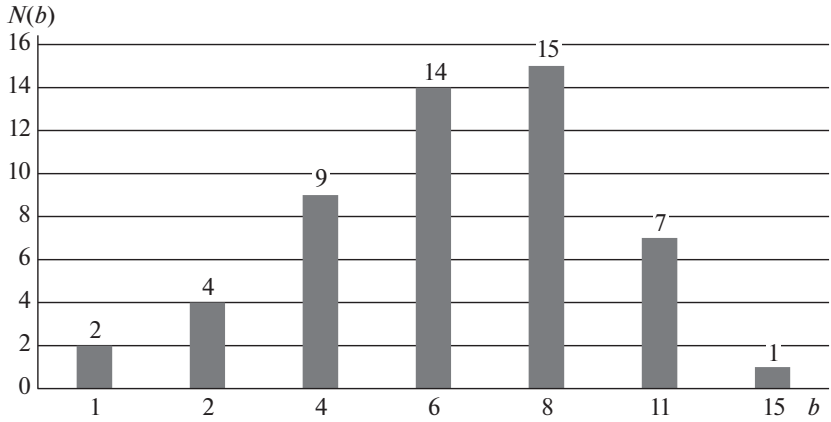
2. Решение задачи о рюкзаке методом тупиковых управлений. Упорядочим номера предметов в порядке убывания весов (объемов)

$$w_{1_1} > w_{4_2} > w_{3_3} \geq w_{5_4} \geq w_{7_5} > w_{2_6} \geq w_{6_7},$$

где $w_{1_1} = 4,\ w_{4_2} = 3,\ w_{3_3} = 2,\ w_{5_4} = 2,\ w_{7_5} = 2,\ w_{2_6} = 1,\ w_{6_7} = 1$.

Пусть $u_k \in \{0, 1\}$ – переменная управления, принимающая значение единица, если k-й предмет по порядку помещается в рюкзак, и нулевое значение в противном случае. Тогда постановку задачи о ранце представим в виде:

$$299u_1 + 221u_2 + 159u_3 + 137u_4 + 157u_5 + 73u_6 + 89u_7 \to \max$$



Зависимость числа тупиковых управлений от веса рюкзака.

при ограничениях

$$4u_1 + 3u_2 + 2u_3 + 2u_4 + 2u_5 + 1u_6 + 1u_7 \le 10.$$

Построенные тупиковые управления и ценность предметов, попадающих в рюкзак, представлены в табл. 4.

Решением задачи является тупиковое управление $\vec{u}_{10} = (1010111)$, которое совпадает с решением J_* (21) в исходных обозначениях с величиной ценности рюкзака:

$$\sum_{j \in J_*} p_j = 299 + 159 + 157 + 73 + 89 = 777.$$

Возникает вопрос о зависимости числа тупиковых управлений от веса рюкзака. Можно предположить, что максимальное число тупиковых управлений будет приходить на вес рюкзака, равный примерно половине суммы весов всех предметов.

На рисунке приведен график зависимости числа тупиковых управлений $N(b)$ от веса b рюкзака для исходных данных из табл. 3, где величина веса (объема) рюкзака определялась по формуле

$$b_k = \sum_{j=1}^{k} w_j, \quad k = 1, 2, \ldots, 7; \quad 1 \le w_1 \le w_2 \le \ldots \le w_7 \le 4.$$

Из рисунка видно, что максимальное число тупиковых управлений приходится на вес рюкзака, равный не менее половины суммы весов всех предметов, и убывает, когда вес рюкзака возрастает, т.е.

$$\left[\frac{1}{2} \sum_{j=1}^{n} w_j \right] \le b < \sum_{j=1}^{n} w_j,$$

где $\left[\frac{1}{2} \sum_{j=1}^{n} w_j \right]$ – целая часть числа $\frac{1}{2} \sum_{j=1}^{n} w_j$.

Таблица 5. Рекуррентные функциональные уравнения Р. Беллмана

Номер шага k	Функция Р. Беллмана $B_1(s_1)$	Тупиковое управление $\vec{u}_l$	Множество состояний $S_k(\vec{u}_l)$
1	$B_1(s_1) = \max\limits_{0 \le u_1 \le [s_1/4]} 299u_1,\ s_1 \in S_1$	0111110	$0 \div 10$
2	$B_2(s_2) = \max\limits_{0 \le u_2 \le \left[\frac{s_2}{3}\right]} \{221u_2 + B_1(s_2 - 3u_2)\}$	0011111	$2 \div 10$
3	$B_3(s_3) = \max\limits_{0 \le u_3 \le \left[\frac{s_3}{2}\right]} \{159u_3 + B_2(s_3 - 2u_3)\}$	0001111	$4 \div 10$
4	$B_4(s_4) = \max\limits_{0 \le u_4 \le \left[\frac{s_4}{2}\right]} \{137u_4 + B_3(s_4 - 2u_4)\}$	0000111	$6 \div 10$
5	$B_5(s_5) = \max\limits_{0 \le u_5 \le \left[\frac{s_5}{2}\right]} \{157u_5 + B_4(s_5 - 2u_5)\}$	0000011	$8 \div 10$
6	$B_6(s_6) = \max\limits_{0 \le u_6 \le \left[\frac{s_6}{1}\right]} \{73u_6 + B_5(s_6 - u_6)\}$	0000001	9, 10
7	$B_7(s_7) = \max\limits_{0 \le u_7 \le \left[\frac{s_7}{2}\right]} \{89u_7 + B_7(s_7 - u_7)\}$	0000001	10

Таблица 6. Результаты расчетов функций Р. Беллмана в прямом времени

k	Шаг 1		Шаг 2		Шаг 3		Шаг 4		Шаг 5		Шаг 6		Шаг 7	
s_k	u_1	$B_1(s_1)$	u_2	$B_2(s_2)$	u_3	$B_3(s_3)$	u_4	$B_4(s_4)$	u_5	$B_5(s_5)$	u_6	$B_6(s_6)$	u_7	$B_7(s_7)$
0	0	0	–	–	–		–		–		–		–	
1	0	0	–	–	–		–		–		–		–	
2	0	0	0	0	–		–		–		–		–	
3	0	0	0 1	 221	–		–		–		–		–	
4	0 **1**	 **299**	**0** 1	**299**	0 1	299	–		–		–		–	
5	0 1	 299	0 1	299	0 1	 380	–		–		–		–	
6	0 1	 299	0 1	299	0 **1**	 **458**	**0** 1	**458**	–		–		–	
7	0 1	 299	0 1	 520	0 1	520	0 1	520	–		–		–	
8	0 1	 299	0 1	 520	0 1	520	0 1	 595	0 **1**	 **615**	–		–	
9	0 1	299	0 1	520	0 1	679	0 1	657	0 1	677 1	0 1	**688**	–	
10	0 1	 299	0 1	 520	0 1	 679	1 0	679	0 1	 752	0 1	 750	0 **1**	 **777**

В этом случае можно предположить, что мощность $|U|$ множества тупиковых U (8), как правило, значительно меньше числа bn, т.е. $|U| \le cn$, где $c > 0$, поэтому трудоемкость алгоритма можно оценить как $O(n)$.

Таблица 7. Результаты сравнений методов решения задачи о рюкзаке

Метод решения	Число прямых вычислений	Процент от полного перебора
Полный перебор	128	100%
Метод динамического программирования ДП III	91	$\frac{91}{128} \times 100\% \approx 71\%$
Метод динамического программирования с ТУ	38	$\frac{38}{128} \times 100\% \approx 30\%$
Метод тупиковых управлений	17	$\frac{17}{128} \times 100\% \approx 13\%$

Представляет интерес нахождение функциональной зависимости числа тупиковых управлений от веса рюкзака. Данную проблему, хотя бы для частных случаев, автор предлагает исследовать читателю.

3. Решение задачи методом динамического программирования с тупиковым управлением. Расчет значений функций Беллмана ведется по всем допустимым u_k от начала к концу, $k = 1, 2, \ldots, 7$.

Исходное дискретное множество области значений функции Р. Беллмана $S_0 = \{0, 1, 2, \ldots, 10\}$. В табл. 5 сведены функции Р. Беллмана на каждом шаге оптимизации и множество состояний в зависимости от тупиковых управлений рекуррентных функциональных уравнений Р. Беллмана.

Результаты расчетов $B_k(s_k)$ представлены в табл. 6, где оптимальное управление выделено подчеркиванием (в скобках после значения функции Р. Беллмана текущего шага указано значение переменной состояния предшествующей функции Р. Беллмана).

Алгоритмом обратной прогонки находим оптимальное управление:

$$B_7(\overline{s}_7)_{|\overline{s}_7=10} = 777 \to u_7^*(\overline{s}_7)_{|\overline{s}_7=10} = 1 \to \overline{s}_6 = \overline{s}_7 - 1u_7^* = 10 - 1 = 9;$$

$$B_6(\overline{s}_6)_{|\overline{s}_6=9} = 688 \to u_6^*(\overline{s}_6)_{|\overline{s}_6=9} = 1 \to \overline{s}_5 = \overline{s}_6 - 1u_6^* = 9 - 1 = 8;$$

$$B_5(\overline{s}_5)_{|\overline{s}_5=8} = 615 \to u_5^*(\overline{s}_5)_{|\overline{s}_5=8} = 1 \to \overline{s}_4 = \overline{s}_5 - 2u_5^* = 8 - 2 = 6;$$

$$B_4(\overline{s}_4)_{|\overline{s}_4=6} = 458 \to u_4^*(\overline{s}_4)_{|\overline{s}_4=6} = 0 \to \overline{s}_3 = \overline{s}_4 - 2u_4^* = 6 - 0 = 6;$$

$$B_3(\overline{s}_3)_{|\overline{s}_3=6} = 458 \to u_3^*(\overline{s}_3)_{|\overline{s}_5=6} = 1 \to \overline{s}_2 = \overline{s}_3 - 2u_3^* = 6 - 2 = 4;$$

$$B_2(\overline{s}_2)_{|\overline{s}_2=4} = 299 \to u_2^*(\overline{s}_2)_{|\overline{s}_2=4} = 0 \to \overline{s}_1 = \overline{s}_2 - 3u_2^* = 4 - 0 = 4;$$

$$B_1(\overline{s}_1)_{|\overline{s}_1=4} = 299 \to u_1^*(\overline{s}_1)_{|\overline{s}_1=4} = 1.$$

Отсюда $\vec{u}_* = \left(u_1^*, u_2^*, u_3^*, u_4^*, u_5^*, u_6^*, u_7^*\right) = (1010111)$, что соответствует в исходных обозначениях решению J_* (21). Сравнительный анализ по трудоемкости методов решений задачи о рюкзаке представлен в табл. 7.

Из табл. 7 следует, что применение тупиковых управлений оказалось эффективнее в $\frac{91}{17} \approx 5{,}3$ раза, чем алгоритм ДП III [2, 16], и в $\frac{38}{17} \approx 2{,}2$ раза, чем метод динамического программирования с тупиковым управлением.

Традиционно считается, что временна́я сложность метода динамического программирования линейна по числу этапов, что является его достоинством. Если число состояний на каждом шаге ограничено константой b, то временная сложность для задачи распределения ресурсов с небулевским управлением может быть оценена как $O\left(b^2 n\right)$ [5]. Временная сложность алгоритма с булевским управлением обычно не превышает величины $O(nb)$ [8]. Покажем, что если определять множество допустимых состояний k-го шага по формуле S_k (16), то временную сложность вычислений можно оценить как $O(n)$ за n шагов алгоритма.

Теорема 3 (о трудоемкости метода динамического программирования с тупиковым управлением). *Пусть нижняя граница переменной состояния $s_k \in S_k$ (15) на k-м шаге определяется по формуле $\underline{s}_k$ (17), тогда временна́я сложность алгоритма динамического программирования с тупиковым управлением решения задачи о рюкзаке в прямом времени будет удовлетворять неравенству*

$$\sum_{k=1}^{n} |S_k| \le cn, \quad 0 < c < b, \tag{22}$$

что равносильно оценке временной сложности как $O(n)$, где $|S_k|$ – число состояний на k-м шаге оптимизации.

Доказательство. Поскольку для переменной состояний на k-м шаге выполняется неравенство

$$s_k = b - \sum_{j=k+1}^{n} w_j u_j \ge 0,$$

то для любых (тупиковых) управлений

$$\vec{u}(k) = (u_k, u_{k+1}, \ldots, u_n) \;\text{ и }\; \vec{u}(k+1) = (u_{k+1}, u_{k+2}, \ldots, u_n)$$

справедливо неравенство

$$b = \sum_{j=1}^{n} w_j \ge \ldots \ge \sum_{j=k}^{n} w_j u_j \ge \sum_{j=k+1}^{n} w_j u_j \ge \ldots \ge w_n > 0 \;\; \forall k = \overline{1, n-1},$$

т.е.

$$b \ge |S_1| \ge |S_2| \ge \ldots \ge |S_n| = 1.$$

Отсюда следуют справедливость неравенства (22) и оценка временной сложности $O(n)$. Теорема 3 доказана.

5. Заключение

В статье рассмотрен эффективный алгоритм тупиковых управлений для решения задач комбинаторной оптимизации, относящийся к классу точных оптимальных алгоритмов с временной сложностью $O(n)$. В [8, 9] показано, что графический алгоритм решения задач комбинаторной оптимизации обладает временной сложностью $O(n)$, однако при этом, как показано в примерах 1 и 2, алгоритм тупиковых управлений оказался более эффективным. По своей эффективности, на данный момент, алгоритм тупиковых управлений превосходит известные алгоритмы, включая алгоритм *Balsub*, представленный в [4].

Также показано, что применение идеи тупиковых управлений при реализации метода динамического программирования позволяет значительно сократить на каждом шаге оптимизации число переменных состояний задачи. Достоинствами метода тупиковых управлений являются его вычислительная простота и более высокое быстродействие по сравнению с известными алгоритмами, что позволяет решать с его помощью характерные для практики задачи большой размерности.

СПИСОК ЛИТЕРАТУРЫ

1. *Беллман Р.* Динамическое программирование. М.: Мир, 1960.
2. *Кормен Т., Лейзерсон Ч., Ривест Р., Штайн К.* Алгоритмы: построение и анализ. М.: Издательский дом “Вильямс”, 2013.
3. *Pisinger D.* A Minimal Algorithm for the 0-1 Knapsack Problem // University of Copenhagen. Oper. Res. 1997. V. 46. No. 5. P. 758–767.
4. *Kellerer H., Pferschy U., Pisinger D.* Knapsack Problems. Springer Science. Business Media, 2010.
5. *Корнеенко В.П.* Методы оптимизации. М.: Высш. шк., 2007.
6. *Сигал И.Х., Иванова А.П.* Введение в прикладное дискретное программирование: модели и вычислительные алгоритмы. М.: Физматлит, 2002.
7. *Гафаров Е.Р., Долгий А., Лазарев А.А., Вернер Ф.* Новый эффективный алгоритм решения задачи об инвестициях // АиТ. 2016. № 9. С. 150–166.

   Gafarov E.R., Dolgui A., Lazarev A.A., et al. A New Effective Dynamic Program for an Investment Optimization Problem // Autom. Remote Control. 2016. V. 77. No. 9. P. 1633–1648. https://doi.org/10.1134/S0005117916090101
8. *Лазарев А.А.* Графический подход к решению задач комбинаторной оптимизации // АиТ. 2007. № 4. С. 13–23.
9. *Лазарев А.А.* Теория расписаний. Методы и алгоритмы. М.: ИПУ РАН, 2019.
10. *Bretthauer K.M., Shetty B.* The Nonlinear Knapsack Problem – Algorithms and applications // Eur. J. Oper. Res. 2002. V. 138. Iss. 3. P. 459–472.
11. *Riedhammer K., Gillick D., Favre B., Hakkani-Tür D.* Packing the Meeting Summarization Knapsack // Proc. Interspeech. Brisbane, Australia, 2008.
12. *Robson J.M.* Finding a Maximum Independent set in Time O(2n/4) // Technical Report 1251-01, LaBRI, Universitte de Bodeaux I, 2001.
13. *Яблонский С.В.* Введение в дискретную математику. М.: Наука, 1986.

14. *Korneenko V.P., Nazyuta S.V., Chursin A.A.* System for Uncertainty Factors Accounting When Optimizing and Choosing Effective Options for Network Work Schedules on a Dynamic Model with Dead-End Controls / IOP Conference Series: Earth and Environmental Science // Proc. Int. Science and Technology Conf. on Earth Science. Vladivostok, Russian: IOP Publishing Ltd, 2021. Sci. 666 062129. https://doi.org/10.1088/1755-1315/666/6/062129.

15. *Пападимитриу Х., Стайглиц К.* Комбинаторная оптимизация. Алгоритмы и сложность. М.: Мир, 1985.

Статья представлена к публикации членом редколлегии А.А. Лазаревым.

Поступила в редакцию 20.01.2020

После доработки 17.03.2021

Принята к публикации 30.06.2021