ISSN 2500-316X (Online)

https://doi.org/10.32362/2500-316X-2020-8-4-96-111

УДК 519.857

Динамическое программирование в прикладных задачах, допускающих сокращение перебора вариантов

Д.А. Карпов,
В.И. Струченков[@]

МИРЭА – Российский технологический университет, Москва 119454, Россия
[@]Автор для переписки, e-mail: str1942@mail.ru

В статье рассматривается разработанный Р. Беллманом алгоритм динамического программирования, основанный на поиске оптимальной траектории, соединяющей узлы предварительно заданной регулярной сетки состояний. Анализируются возможности резкого повышения эффективности применения динамического программирования при решении прикладных задач, обладающих специфическими особенностями, что позволяет отказаться от разбиения регулярной сетки состояний и реализовать алгоритм поиска оптимальной траектории при отбраковке не только бесперспективных вариантов путей, приводящих в каждое из состояний, и всех их продолжений, как в алгоритме Р. Беллмана, но и собственно бесперспективных состояний и всех вариантов исходящих из них путей. Сформулированы и обоснованы условия, при которых возможна отбраковка бесперспективных состояний. Установлено, что многие прикладные задачи удовлетворяют этим условиям. Для решения подобных задач предложен и реализован новый алгоритм динамического программирования. Приводятся конкретные примеры таких прикладных задач: оптимальное распределение однородного ресурса между несколькими потребителями, оптимальная загрузка транспортных средств, оптимальное распределение финансов при выборе инвестиционных проектов. Для решения этих задач ранее предлагались алгоритмы динамического программирования с отбраковкой бесперспективных путей, но без отбраковки состояний. Число бесперспективных состояний, появляющихся на различных этапах динамического программирования, и, соответственно, эффективность нового алгоритма, зависит от конкретных числовых значений исходных данных. Для двухпараметрической задачи оптимальной загрузки транспортных средств при ограничении по весу и объёму приведены результаты сопоставительных расчётов по алгоритму Р. Беллмана и

по новому алгоритму динамического программирования. В качестве исходных данных для серии расчётов использовались псевдослучайные числа. В результате анализа показано, что сравнительная эффективность алгоритма с отбраковкой состояний растёт при увеличении размерности задачи. Так в задаче оптимального выбора предметов для загрузки транспортного средства заданной грузоподъёмности при числе предметов 150 количество запоминаемых состояний и время счёта снижаются в 50 и 57 раз, соответственно, при использовании нового алгоритма по сравнению с классическим алгоритмом Р. Беллмана. Для 15 предметов соответствующие числа равны 13 и 4.

Ключевые слова: динамическое программирование, целевая функция, оптимальная траектория, уравнение Р. Беллмана.

Для цитирования: Карпов Д.А., Струченков В.И. Динамическое программирование в прикладных задачах, допускающих сокращение перебора вариантов. *Российский технологический журнал.* 2020;8(4):96-111. https://doi.org/10.32362/2500-316X-2020-8-4-96-111

Dynamic programming in applied tasks which are allowing to reduce the options selection

Dmitry A. Karpov,
Valeriy I. Struchenkov[@]

MIREA – Russian Technological University, Moscow 119454, Russia
[@]Corresponding author, e-mail: str1942@mail.ru

The article discusses the dynamic programming algorithm developed by R. Bellman, based on the search for the optimal trajectory connecting the nodes of a predefined regular grid of states. Possibilities are analyzed for a sharp increase in the effectiveness of using dynamic programming in solving applied problems with specific features, which allows us to refuse to split a regular grid of states and implement an algorithm for finding the optimal trajectory when rejecting not only unpromising options for paths leading to each of the states, and all of them continuations, as in R. Bellman's algorithm, but also actually hopeless states and all variants of paths emanating from them. The conditions are formulated and justified under which the rejection of hopeless states is possible. It has been established that many applied problems satisfy these conditions. To solve such problems, a new dynamic programming algorithm described in the article is proposed and implemented. Concrete examples of such applied problems are given: the optimal distribution of a homogeneous resource between several consumers, the optimal loading of vehicles, the optimal distribution of finances when choosing investment projects. To solve these problems, dynamic programming algorithms with rejecting unpromising paths, but without rejecting states, were previously proposed. The number of hopeless states that appear at various stages of dynamic programming and, accordingly, the effectiveness of the new algorithm depends on the specific numerical values of the source data. For the two-parameter problem of optimal loading of vehicles with weight and volume constraints, the results of comparative calculations by the R. Bellman algorithm and the new dynamic programming algorithm are presented. As a source of data for a series of calculations, pseudorandom numbers were used. As a result of the analysis, it was shown that the comparative efficiency of the algorithm with rejection of states increases with increasing dimension of the problem. So, in the problem of the optimal choice of items for loading a vehicle of a given carrying capacity with a number of items of 150, the number of memorized states and the counting time are reduced by 50 and 57 times, respectively, when using the new algorithm compared to the classical algorithm of R. Bellman. And for 15 items, the corresponding numbers are 13 and 4.

Keywords: dynamic programming, objective function, optimal trajectory, rejection of states.

For citation: Karpov D.A., Struchenkov V.I. Dynamic programming in applied tasks which are allowing to reduce the options selection. *Rossiiskii tekhnologicheskii zhurnal = Russian Technological Journal.* 2020;8(4):96-111 (in Russ.). https://doi.org/10.32362/2500-316X-2020-8-4-96-111

Введение

Метод динамического программирования для решения задач оптимизации был предложен американским математиком Р. Беллманом в середине прошлого века. Основная идея метода состоит в сведении исходной сложной задачи к последовательности относительно простых однотипных задач. Рассматривалась задача управления движением динамической системы, состояние которой описывается одним или несколькими параметрами. Внешние воздействия (управления), производимые в заданные дискретные моменты времени, переводят систему из одного состояния в другое. Последовательность переходов образует траекторию движения системы. Каждый переход сопровождается соответствующими затратами. Ставилась задача: найти такой набор управлений, который переводит систему из заданного начального состояния в заданное конечное состояние с наименьшими затратами.

Метод основан на принципе оптимальности Р. Беллмана, смысл которого состоит в следующем: если в каждом из состояний, в котором может находиться система, её дальнейшее поведение не зависит от того, как она оказалась в этом состоянии, то дальнейшая траектория должна быть оптимальной. Другими словами, речь идёт о системах без влияния предыстории. Важно отметить, что, будет влиять предыстория или нет, зависит от формализации понятия «состояние системы» при решении конкретной прикладной задачи. Кроме того, должна быть возможность вычисления целевой функции для каждого этапа (управления) отдельно.

Реально встречаются задачи, в которых целевую функцию можно вычислить только для полного набора переменных, т.е. только после того, как имеется вариант всей траектории (вариант пути из начальной точки в конечную).

На русском языке книга Р. Беллмана была издана в 1960 г. [1] и вызвала большой интерес широкого круга советских исследователей, преподавателей, специалистов различных областей деятельности. Сложилось впечатление, что новый метод позволит решить многие прикладные задачи, которые в то время казались неразрешимыми.

Однако первоначальная эйфория вскоре рассеялась, так как этот метод далеко не универсальный и объём вычислений, необходимых для решения реальных прикладных задач, оказался столь значительным, что реализация метода на имевшихся в то время общедоступных ЭВМ, таких как БЭСМ 2, Минск 22, Урал и др. была невозможна.

Попытки упростить задачу при дополнительных допущениях с тем, чтобы получить хоть какое-то машинное решение далеко не всегда были успешными. Так, одной из первых попыток использования метода динамического программирования в проектировании продольного профиля новых железных дорог были разработки Института Кибернетики АН УССР и ЦНИИС Минтранстроя [2, 3], в которых использовался предложенный В.С. Михалевичем метод последовательного анализа вариантов [4]. По существу этот метод отличался от оригинального метода Р. Беллмана только тем, что оптимальная траектория строилась по направлению не от конечной точки к начальной, а от начальной точки к конечной.

Эта попытка оказалась неудачной из-за принятых допущений при реализации алгоритма. Ключевое понятие – состояние системы – было формализовано как узел сетки

варьирования [3]. При этом нарушался принцип оптимальности, так как множества возможных продолжений двух сравниваемых путей, приводящих в заданное состояние, не совпадали из-за влияния ограничений по разности уклонов [3]. В условиях пересечённого рельефа это приводило не только к отклонениям от оптимума, но и к остановке алгоритма из-за отсутствия допустимых продолжений (вырождение вариантов).

Корректный алгоритм был реализован только к 1975 г. [5], когда стали считать состоянием системы отрезок, соединяющий два узла, и сравнивать только такие варианты, у которых этот отрезок общий. Однако и в этом случае алгоритм можно было использовать только для простых моделей целевой функции, например, объёмов земляных работ, тогда как при оптимизации по строительной стоимости для вычисления целевой функции нужно иметь траекторию полностью. На участках, где насыпи сооружаются из грунта выемок, появляется дополнительная взаимосвязь элементов искомой проектной линии. Задача была решена по методу проекции градиента [6] с учётом этой взаимосвязи и особенностей структуры матрицы системы ограничений. Соответствующий пакет программ «Профиль» для ЭВМ БЭСМ 4 широко использовался при проектировании Байкало-Амурской магистрали.

В процессе решения задачи об оптимальном профиле дороги были выявлены недостатки метода динамического программирования, что позволило в дальнейшем избежать ошибок при его применении.

В начале 60-х годов прошлого века Р. Беллман с сотрудниками, располагая более мощными ЭВМ, чем доступные в то время для советских исследователей, реализовали свой метод при решении ряда практически важных задач [7].

В настоящее время алгоритмы динамического программирования успешно применяются для решения прикладных задач из различных областей практики [8–22]. Одно из последних предложений – это аппроксимация плоских кривых, заданных дискретно, сплайнами сложной структуры [23].

Алгоритм Р. Беллмана получил широкое практическое применение, излагается в учебной литературе и может считаться классическим.

Однако при использовании этого алгоритма объём вычислений резко возрастает с ростом размерности задачи и остаётся значительным даже для современных общедоступных компьютеров. Это особенно важно при разработке систем, в которых динамическое программирование встроено в многократно повторяющийся цикл расчётов. В этой связи поиск возможностей сокращения перебора вариантов в динамическом программировании при разработке новых алгоритмов с использованием особенностей конкретных прикладных задач продолжает оставаться актуальным.

Цель настоящей статьи – изложить новый алгоритм динамического программирования, который позволяет решить многие прикладные задачи с резким сокращением перебора вариантов за счёт отбраковки не только бесперспективных путей, приводящих в конкретное состояние, но и собственно бесперспективных состояний.

1. Динамическое программирование с разбиением регулярной сетки состояний

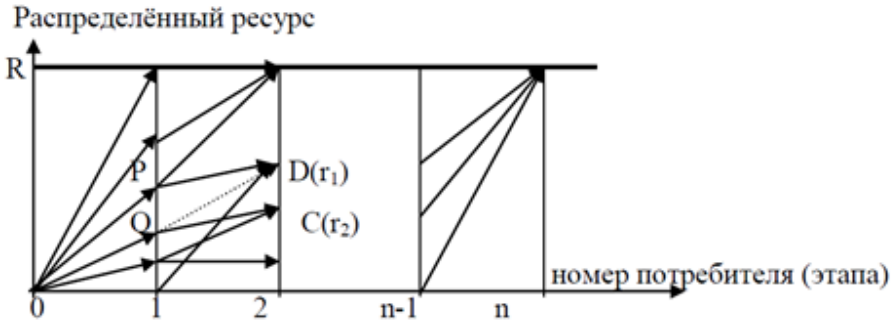
Одна из первых прикладных задач, решённых Р. Беллманом по его методу, – это задача оптимального распределения ограниченного объёма однородного ресурса между несколькими потребителями.

Задача состоит в следующем [1]:

Найти $\max \sum_{i=1}^{n} f_i(x_i)$ при $\sum_{i=1}^{n} x_i \le R$,

где R – это распределяемое количество ресурса, $x_i \ge 0$ – его количество, выделенное для i-го потребителя, а $f_i(x_i)$ – эффективность использования ресурса i-ым потребителем.

В рассматриваемой задаче очередной этап (шаг) – это определение количества ресурса для очередного потребителя, а ***состояние системы – это уже распределённый ресурс***, который не должен превышать заданной величины R (рисунок).



Многоэтапный процесс распределения ресурса.

Первый потребитель может получить $x_1 = 0, \delta, 2\delta, 3\delta, ..., k\delta \le R$ единиц ресурса с оценкой эффективности $f_1(x_1)$. Второй потребитель может получить не более, чем $x_2 = R - x_1$ (это остаток ресурса). Сумма $r = x_1 + x_2$ – это состояние системы после второго этапа. В него можно перейти разными путями с различной эффективностью. Её максимальное значение по наилучшему пути обозначим $\Phi_2(r) = f_1(x_1) + f_2(x_2)$. Далее $\Phi_i(r)$ – это максимальная эффективность за i этапов при распределённом ресурсе r. Предполагается, что все функции $f_i(x_i)$ неубывающие. В соответствии с принципом оптимальности должно выполняться уравнение Р. Беллмана, которое даёт рекуррентную связь между $\Phi_i(r)$ и $\Phi_{i-1}(r)$

$$\Phi_i(r) = \max_{x_i}(f_i(x_i) + \Phi_{i-1}(r - x_i)).$$

Это означает, что в любое состояние r на i-ом этапе надо переходить из такого состояния на (i–1)-ом этапе, при котором суммарная эффективность перехода в состояние r из начальной точки была бы максимальной. Такой выбор x_i (рисунок) означает выбор наиболее эффективного из всех путей, сходящихся в одной точке i-ой вертикали, которой соответствует ресурс r. Значение эффективности по этому пути $\Phi_i(r)$ и состояние $r - x_i$ запоминаются для всех r. Сравнение путей, приводящих в одно и то же состояние, с отбраковкой бесперспективных путей ***и всех их продолжений*** делают динамическое программирование более эффективным, чем полный перебор вариантов. Но при решении задач большой размерности (мелкой сетке) по этому алгоритму затраты машинного времени могут оказаться неприемлемо велики.

Возникает вопрос об ***отбраковке не только путей***, приводящих в одно и то же состояние, ***но и собственно бесперспективных состояний (хотя бы в некоторых задачах)***.

Применительно к рассматриваемой задаче этот вопрос означает: могут ли на одном и том же этапе быть два состояния с $r_1 > r_2$ и $\Phi_i(r_1) < \Phi_i(r_2)$? Пусть в состояние r_1 (точка D

на рисунке) по оптимальному пути переходим из точки P предыдущего этапа, а в состояние r_2 (точка C) из точки Q. Но переходя из точки Q в точку D, получим большую (или равную) эффективность, чем $\Phi_i(r_2)$ так как все функции $f_i(x_i)$ неубывающие.

В данной задаче ответ на поставленный вопрос об отбраковке состояний отрицательный, и с увеличением затрат ресурса увеличивается и эффективность. Однако существуют задачи, в которых в отличие от рассматриваемой далеко не все переходы возможны.

2. Отбраковка бесперспективных состояний

Рассмотрим классическую задачу целочисленного (бинарного) линейного программирования, известную как задача о рюкзаке:

Найти $\max z = c_1x_1 + c_2x_2 + ... + c_nx_n$ при

$$a_1x_1 + a_2x_2 + ... + a_nx_n \le b,\ x_i \in \{0,1\},\ a_j > 0;\ c_j > 0;\ j = 1, ..., n. \quad (1)$$

Можно считать, что задача состоит в оптимальном распределении заданного ресурса b между n потребителями, j-ый потребитель получает a_j единиц ресурса или ничего. Но можно говорить о задаче оптимального выбора из n предметов. При выборе j-го предмета затрачивается ресурс в количестве a_j и достигается эффект c_j. Выбирать нужно так, чтобы не израсходовать ресурса больше b и достигнуть максимального суммарного эффекта z.

Если может быть несколько $k_j > 1$ экземпляров некоторых или всех предметов, то есть $x_j \in \{0,1, ..., k_j\}$, то принципиальных сложностей по сравнению с задачей (1) не возникает. От рассмотренной выше задачи распределения ресурса данная задача отличается тем, что ресурс распределяется неравными долями, то есть при формализации понятия «состояние системы» как количество уже распределённого ресурса мы не получаем регулярной сетки состояний. Именно это обстоятельство может дать и, как будет показано ниже на конкретных задачах, дает возможность отбраковки бесперспективных состояний. Состояние A бесперспективно и может далее не рассматриваться, если на том же этапе есть другое состояние B, которому соответствует меньший затраченный ресурс, но не меньшая эффективность. При этом должно выполняться дополнительное условие: из состояния B в конечное состояние можно перейти с не меньшей эффективностью, чем по наилучшему пути из состояния A.

В рассматриваемых задачах это условие выполнено, так как возможности выбора и соответствующие последствия (затраты ресурса и эффективность) на оставшихся этапах одинаковы для всех состояний одного и того же этапа.

Можно рассматривать задачу (1) при любом количестве экземпляров каждого предмета, как двухкритериальную: один критерий – эффективность (требуется максимум), а другой критерий – суммарный использованный ресурс (требуется минимум), тогда на каждом этапе оставшиеся состояния должны образовывать множество Парето. При этом остаётся и отбраковка путей, приводящих в одно и то же состояние. Поэтому в каждое состояние из паретовского множества после отбраковки приводит только один путь. Как уже отмечалось, для отбраковки путей в классическом алгоритме Р. Беллмана требовалось отсутствие влияния предыстории. Для отбраковки состояний на каждом шаге для любых двух состояний равные приращения ресурса (воздействия на систему)

должны приводить к равным изменениям эффективности (целевой функции). В этом случае возможность отбраковки неэффективных (по Парето) состояний очевидна.

Задание всего множества допустимых состояний в виде регулярной сетки не требуется. Множество Парето [24] на каждом этапе формируется в процессе счёта.

Остаётся показать, что для многих прикладных задач условия отбраковки состояний выполняются, и при этом повышается эффективность в смысле меньшего объёма вычислений и требуемой памяти по сравнению с классическим динамическим программированием.

Для начала рассмотрим простую задачу, в которой выявлена высокая эффективность алгоритма, использующего отбраковку состояний.

3. Оптимальное использование транспортных средств

Будем рассматривать задачу.

Найти $\max z = c_1 x_1 + c_2 x_2 + ... + c_n x_n$ при

$$a_1 x_1 + a_2 x_2 + ... + a_n x_n \le b,\ a_j > 0;\ c_j > 0;\ j = 1, ..., n.$$

Это задача (1) при

$$x_i \in \{0, 1, ..., k_j\},\ j = 1, ..., n. \qquad (2)$$

Целочисленность a_j и c_j $(j = 1, ..., n)$ не требуется. Здесь a_j – вес предмета j-го вида, а c_j – его стоимость. Если все $k_j = 1$ то это задача о рюкзаке. Переменными являются x_j – число предметов j-го вида, которые надо взять, чтобы не превысить максимальную грузоподъёмность b и получить максимальную суммарную стоимость.

Задача оптимальной загрузки транспортного средства была решена Р. Беллманом [1] с отбраковкой вариантов путей (траекторий), приводящих в одно и то же состояние, но без отбраковки бесперспективных состояний. Этот же алгоритм приводится при рассмотрении других примеров и задач в учебной литературе [25, 26].

Отметим, что нет необходимости в разбиении сетки состояний, а при отсутствии целочисленности a_j из двух близких состояний, получаемых в процессе счёта, можно оставить одно. Мера близости – это аналог дискрета, используемого при разбиении регулярной сетки, к узлам которой притягиваются состояния с нецелыми значениями параметров, которыми они определяются (в данной задаче это суммарный вес выбранных предметов). Эта мера зависит от требуемой точности решения задачи и от точности исходных данных.

Оптимальную траекторию (путь) удобно строить «от начала к концу», в качестве состояния системы принимать суммарный вес выбранных предметов и вычислять соответствующую суммарную стоимость.

Независимо от количества экземпляров каждого предмета алгоритм решения задачи состоит из следующих пунктов:

1. Упорядочиваем предметы по возрастанию веса.

2. Если в соответствующей последовательности стоимостей нарушается монотонность, то исключаем предмет, имеющий больший вес, но меньшую стоимость, чем предшествующий.

3. На первом шаге рассматриваем все состояния, получаемые при выборе допустимого числа экземпляров первого предмета. Для каждого состояния вычисляем суммарное значение стоимости. Никакой отбраковки нет.

4. На последующих шагах (с номерами $t = 2, 3, ..., n$) решается вопрос о добавлении к уже выбранным одного или нескольких экземпляров предмета с номером t при соблюдении ограничения по грузоподъёмности. Если этот предмет не брать, то на шаге t фиксируются все состояния шага $t - 1$. Если брать, то для каждого состояния шага $t - 1$ последовательно рассматривается целесообразность дополнения к нему 1, 2, 3, ..., k_t числа экземпляров без превышения ресурса грузоподъёмности. При этом к каждому состоянию $(t - 1)$-го шага добавляется соответствующий вес, и определяется положение нового состояния в массиве (списке) имеющихся состояний с сохранением упорядоченности по возрастанию. Возможны различные ситуации при вычислении нового состояния на шаге t:

4.1. Новое состояние совпадает с одним из имеющихся. В этом случае остаётся то из них, которому соответствует бóльшая стоимость (как в классическом алгоритме).

4.2. Новое состояние сравнивается с предшествующим ему. Если новому состоянию соответствует бóльшая стоимость, то оно помещается в массив состояний шага t, иначе исключается из рассмотрения.

4.3. Новое состояние сравнивается со следующим за ним (если оно есть). Если новому состоянию соответствует бóльшая (или равная) стоимость, то оно ставится в массив вместо имеющегося. При этом возможна отбраковка нескольких состояний с бóльшим использованным ресурсом, но с меньшей (или равной стоимостью) по сравнению с новым состоянием.

5. Для каждого состояния запоминаются суммарные вес и стоимость, а также связь с предшествующим этапом, то есть состояние, из которого пришли в данное состояние.

6. После последнего этапа допустимое состояние с большей стоимостью даёт ответ к задаче (максимальное значение целевой функции) и обратным разворотом по цепочке связей восстанавливается оптимальная траектория. В данной задаче это количество экземпляров каждого из предметов, которые надо выбрать.

Для иллюстрации применимости и эффективности динамического программирования с отбраковкой состояний рассмотрим несколько задач.

Задачу об оптимальной загрузке транспортного средства будем рассматривать в её самой простой постановке [1], т.е. как задачу о рюкзаке (1). Решение этой задачи приводится в учебной литературе как пример применения метода динамического программирования [12].

Численный пример заимствован из книги Е.С. Вентцель [25], где он описывается следующим образом. Имеется автомашина грузоподъёмностью $Q = 35$ единиц веса и шесть предметов, каждый в одном экземпляре, веса и стоимости которых указаны в табл. 1.

Суммарный вес предметов превышает грузоподъёмность машины и поэтому требуется эту грузоподъёмность использовать оптимальным образом, то есть взять такие предметы, суммарный вес которых не превышает $Q = 35$, а суммарная стоимость максимальна.

Согласно [25] рассматривается шесть этапов (шагов), на каждом из которых принимается решение брать соответствующий предмет в машину или не брать. Номер этапа соответствует номеру предмета в табл. 1. На каждом этапе всего лишь два возможных

Таблица 1. Исходные данные

Предмет $П_i$	**$П_1$**	**$П_2$**	**$П_3$**	**$П_4$**	**$П_5$**	**$П_6$**
Вес q_i	4	7	11	12	16	20
Стоимость c_i	7	10	15	20	27	34

решения (управления): 0 – не брать соответствующий предмет и 1 – брать. Состояние системы перед очередным этапом характеризуется оставшимся ресурсом грузоподъёмности. В [25] задача решается с помощью классического алгоритма Р. Беллмана [1] – разбивается регулярная сетка, в которой на каждом из 6 этапов 36 состояний, и строится оптимальная траектория от «конца к началу».

Заметим, что если строить траекторию от «начала к концу», то после первого этапа есть только 2, после второго – 4 и т.д., а не 36 состояний, а сетка состояний вообще не нужна.

Можно считать состоянием системы уже использованный ресурс, строить все реально достижимые состояния по этапам, вычисляя для каждого из них суммарную стоимость взятых предметов, и при попадании в одно и то же состояние двух (или более) путей оставлять тот, которому соответствует максимальная стоимость.

В рассматриваемой задаче такое состояние возникает после третьего этапа: использовано 11 единиц ресурса. В него приводят два пути: (0, 0, 1) – взять только третий предмет и (1, 1, 0) – взять только два первых. По первому пути суммарная стоимость составляет 15, а по второму пути – 17 (табл. 1). Первый путь и все его продолжения отбраковываются.

Отметим, что при дробных весах предметов двух путей, приводящих в одно и то же состояние, могло и не быть, если не округлять веса, то есть не притягиваться к регулярной сетке, что сопряжено с ошибками. Однако динамическое программирование с отбраковкой состояний эффективно и в этом случае.

Рассмотрим два состояния после четвёртого шага: путь (1, 0, 0, 1) – взяли только первый и четвёртый предмет и путь (0, 1, 1, 0) – взяли только второй и третий предмет.

Для первого из них использовано 16 единиц веса (4 + 0 + 0 + 12) и получена суммарная стоимость 27 (7 + 0 + 0 + 20), а для второго использовано 18 единиц веса (0 + 7 + 11 + 0) и получена стоимость 25 (0 +10 + 15 + 0). Второе состояние 18 (25) (использованный ресурс и в скобках суммарная стоимость) бесперспективно и может быть отбраковано, так как имея запас ресурса грузоподъёмности проще размещать оставшиеся предметы, а возможности выбора те же.

На пятом этапе бесперспективно состояние 22 (32) (так как есть состояние 20 (34)), а также состояния 27 (42), 30 (45) и 34 (52). Вообще после пятого этапа останется только 15 (а не 32 и не 36) состояний. Поэтому алгоритм с отбраковкой состояний в данной задаче эффективнее классического алгоритма Р. Беллмана.

С шестым предметом весом 20 и стоимостью 34 единицы всё просто. Если его не брать, то наилучшим окажется состояние 35 (57), иначе можно получить состояние 35 (56).

Оптимальный вариант загрузки машины состоит в том, что надо взять предметы с номерами 2, 4 и 5 общим весом 35 и стоимостью 57.

4. Отбраковка состояний в двухпараметрических задачах

Увеличение числа параметров состояния резко увеличивает объём вычислений при реализации классического алгоритма динамического программирования и может создать

вычислительные трудности даже при использовании современных персональных компьютеров. В значительной мере эти трудности можно преодолеть при решении задач, допускающих отбраковку состояний.

Рассмотрим задачу о загрузке транспортного средства в следующей постановке. Имеется N различных предметов, каждый в нескольких экземплярах. Вес, объем и стоимость каждого предмета известны. Нужно выбрать предметы так, чтобы при загрузке ими транспортного средства, грузоподъемность которого равна W, а вместительность равна V, суммарная стоимость этого набора была максимальной. Задача формализуется следующим образом: найти $\max \sum_{i=1}^{N} k_i c_i$ при ограничениях

$$\begin{cases} \sum_{i=1}^{N} k_i v_i \le V, \\ \sum_{i=1}^{N} k_i w_i \le W, \\ k_i \ge 0, \end{cases}$$

где c_i, v_i, w_i – соответственно, стоимость, объем и вес i-го предмета, k_i – количество взятых предметов i-го вида. Предполагается, что количество предметов каждого вида достаточно для загрузки транспортного средства при использовании только этого вида предметов. Как вариант может рассматриваться задача при ограничениях $k_i \le k_i^{\max}$.

Неизвестными являются значения k_i.

Очередной этап – это определение числа предметов соответствующего вида в дополнение к тем, которые уже выбраны. Состояние системы формализуется двумя параметрами: суммарные вес и объём уже выбранных предметов.

Рассматриваемая задача однокритериальная, так как требуется максимизировать суммарную стоимость при ограничении на суммарные объём и вес. Мы будем рассматривать её как трехкритериальную задачу: первый критерий – суммарная стоимость выбранных предметов (нужен максимум), а второй и третий – суммарные использованные вес и объем (нужен минимум).

Задача решается с помощью изложенного выше алгоритма динамического программирования с отбраковкой состояний, то есть с формированием на каждом этапе множества Парето трёхкритериальной задачи.

Были выполнены сопоставительные расчёты по новому алгоритму и классическому алгоритму Р. Беллмана (без разбиения регулярной сетки состояний и с отбраковкой бесперспективных путей, приводящих в конкретное состояние). Последовательно увеличивалось число предметов, и выполнялся расчёт при фиксированной грузоподъемности 600 единиц веса и вместительности 500 единиц объёма. Исходные данные получались как псевдослучайные числа. Они представлены в табл. 2, а результаты – в табл. 3.

Расчёты по двум программам производились на персональном компьютере с процессором Intel Core 2 DUO2400 МГц и оперативной памятью 2048 МБ.

Таблица 2. Исходные данные для решения задачи

Предметы				
№	**Предмет**	**Вес**	**Объем**	**Стоимость**
1	$П_1$	31	83	11
2	$П_2$	45	18	96
3	$П_3$	86	49	27
4	$П_4$	97	59	72
5	$П_5$	22	9	61
6	$П_6$	89	22	38
7	$П_7$	54	20	60
8	$П_8$	91	86	87
9	$П_9$	10	49	72
10	$П_{10}$	80	85	20
11	$П_{11}$	50	96	94
12	$П_{12}$	35	11	80
13	$П_{13}$	96	62	9
14	$П_{14}$	12	13	35
15	$П_{15}$	37	9	38
16	$П_{16}$	67	65	62
17	$П_{17}$	52	74	1
18	$П_{18}$	96	89	9
19	$П_{19}$	35	48	59
20	$П_{20}$	75	70	90
21	$П_{21}$	84	35	42
22	$П_{22}$	91	23	66
23	$П_{23}$	80	91	68
24	$П_{24}$	48	72	16
25	$П_{25}$	79	34	38
26	$П_{26}$	65	33	41
27	$П_{27}$	43	37	61
28	$П_{28}$	59	43	34
29	$П_{29}$	16	17	83
30	$П_{30}$	57	22	93

Из табл. 3 следует вывод о более высокой эффективности нового алгоритма по сравнению с классическим алгоритмом Р. Беллмана при решении двухпараметрических задач данного типа. Важно отметить, что с ростом размерности задачи различие этих алгоритмов, как по объёму памяти, так и по времени счёта возрастает. Так уже при числе предметов равном 30 новый алгоритм работает в 52 раза быстрее.

Дополнительно были выполнены расчёты при большем количестве предметов. Их результаты приведены в табл. 4.

Характерно, что при малом числе предметов (3 и 4) классический алгоритм эффективнее нового, так как при этом число отбракованных состояний относительно невелико и затраты времени на их поиск больше, чем эффект от их отбраковки.

Таблица 3. Результаты расчётов

Количество предметов (шт.)	Динамическое программирование (метод Р. Беллмана)		Динамическое программирование с отбраковкой состояний	
	Количество состояний	Время счета (сек)	Количество состояний	Время счета (сек)
3	303	0.02	61	0.05
4	865	0.03	88	0.06
5	1316	0.14	116	0.14
6	5264	0.59	144	0.20
7	15689	3.04	172	0.23
8	18961	6.63	201	0.25
9	21454	8.67	423	0.27
10	23514	11.02	646	0.30
11	28621	14.03	869	0.33
12	32013	17.81	1967	0.53
13	58214	22.64	3065	0.84
14	86213	28.78	6216	3.65
15	107501	36.59	8270	7.91
16	160640	46.52	10040	8.86
17	212562	59.14	11809	9.72
18	283504	75.18	13578	10.28
19	352981	95.58	15347	11.56
20	479248	121.51	17116	12.23
21	547665	154.47	18885	13.12
22	618956	196.37	20654	14.02
23	717536	249.63	22423	14.79
24	774144	317.34	24192	15.52
25	856713	403.43	25961	16.44
26	970552	512.85	27730	17.57
27	1115954	651.97	29499	19.01
28	1250720	828.82	31268	20.05
29	1622112	1053.63	33794	23.53
30	1816000	1339.43	36320	25.30

Таблица 4. Результаты расчётов при большом числе предметов

Количество предметов (шт.)	Динамическое программирование (метод Р. Беллмана)		Динамическое программирование с отбраковкой состояний	
	Количество состояний	Время счета (сек)	Количество состояний	Время счета (сек)
50	7409484	7314.21	145284	138.97
75	10589540	8580.46	203645	156.80
100	13871196	10773.10	256874	189.52
150	21588545	13108.23	392519	226.65

Рассмотренными примерами список задач, в которых целесообразно использовать динамическое программирование с отбраковкой состояний, далеко не исчерпывается. Так, известная задача об оптимальном распределении финансовых ресурсов между конкурирующими инвестиционными проектами описывается той же моделью, что

и задача о рюкзаке (1) и может быть решена с использованием изложенного выше алгоритма.

Задача выбора поставщиков товара, поставляемого партиями различного объёма и стоимости, отличается от задачи об оптимальной загрузке транспортных средств (2) только знаком неравенства – ограничения на суммарный вес – и тем, что вместо максимизации требуется минимизация, что не мешает использованию алгоритма с отбраковкой состояний.

Заключение

В итоге можно констатировать следующее:

1. Сформулированные выше условия применимости алгоритма с отбраковкой состояний выполняются для многих задач.

2. Число бесперспективных состояний и, соответственно, эффективность нового алгоритма по сравнению с классическим алгоритмом Р. Беллмана трудно оценить теоретически. Она зависит от конкретной математической модели и конкретных числовых значений исходных данных.

3. Для задач, допускающих отбраковку состояний:

– планирование использования частично возобновляемых ресурсов [26];

– расчёт оптимальных сроков замены оборудования;

– выбор способов (механизмов) для производства работ для определения эффективности нового алгоритма требуются экспериментальные расчёты.

4. В задачах, допускающих отбраковку состояний, новый алгоритм можно рекомендовать при большой размерности задачи, когда классический алгоритм Р. Беллмана оказывается неприемлемым из-за больших затрат машинного времени. В таких случаях новый алгоритм может дать снижение объёма используемой памяти и времени счёта, но класс таких задач существенно уже, чем для классического алгоритма. В частности, если исходная задача непрерывная и дискретность вводится искусственно [3, 27], то приходится разбивать регулярную сетку и использовать классический алгоритм Р. Беллмана [1, 7].

Литература:

1. Беллман Р. Динамическое программирование. М.: ИЛ., 1960. 402 с.

2. Ляховский В.Н., Михалевич В.С., Быков В.И. Определение на ЭВМ наивыгоднейшего положения красной линии продольного профиля на вольном ходу. *Транспортное строительство.* 1962;4:41-43.

3. Михалевич В.С., Шор Н.З. Математические основы решения задачи выбора оптимального очертания продольного профиля. *Труды Всесоюзного НИИ транспортного строительства.* 1964;51:12-14.

4. Михалевич В.С. Последовательные алгоритмы оптимизации и их применение. *Кибернетика.* 1965;1:45-56.

5. Михалевич В.С., Быков В.И., Сибирко А.Н. К вопросу проектирования оптимального продольного профиля дороги. *Транспортное строительство.* 1975;6:39-40.

6. Космин В.В., Струченков В.И., Фрадков Е.Б. Проектирование продольного профиля дороги на ЭВМ. *Транспортное строительство.* 1971;4:38-42.

7. Беллман Р., Дрейфус С. Прикладные задачи динамического программирования. М.: Наука, 1965. 460 с.

8. Лежнев А.В. Динамическое программирование в экономических задачах. М.: Бином, 2010. 176 с.

9. Кремер Н.Ш. Исследование операций в экономике. М.: Юрайт, 2012. 430 с. ISBN 978-5-9916-1849-6

10. Cavagnari G., Marigonda A., Piccoli B. Generalized dynamic programming principle and sparse mean-field control problems. *J. Math. Anal. Appl.* 2020;481(1): Article No. 123437. https://doi.org/10.1016/j.jmaa.2019.123437

11. Ramahatana F., David M. Economic optimization of micro-grid operations by dynamic programming with real energy forecast. *J. Phys. Conf.* 2019;1343(1): Article No. 012067.
https://doi.org/10.1088/1742-6596/1343/1/012067

12. Firdausiyah N., Taniguchi E., Qureshi A.G. Impacts of Urban Consolidation Centres for Sustainable City Logistics Using Adaptive Dynamic Programming Based Multi-Agent Simulation. In: IOP Conference Series: Earth and Environmental Science. 2019;328(1): Article No. 012071.
https://doi.org/10.1088/1755-315/328/1/012071

13. He S., Shin H.-S., Tsourdos A. Computational guidance using sparse Gauss-Hermite quadrature differential dynamic programming. *IFAC-PapersOnLine.* 2019;52(12):13-18.
https://doi.org/10.1016/j.ifacol.2019.11.062

14. He S., Guo S., Chen, K., Deng L., Liao Z., Xiong F., Yin J. Dataset for reservoir im-poundment operation coupling parallel dynamic programming with importance sampling and suc-cessive approximation. *Data in Brief.* 2019;26: Article No. 104440.
https://doi.org/10.1016/j.dib.2019.104440

15. Fayaed S.S., Fiyadh S.S., Khai W.J., Ahmed A.N., Afan H.A., Ibrahim R.K. Improving dam and reservoir operation rules using stochastic dynamic programming and artificial neural network integration model. *Sustainability.* 2019;11(19):5367.
https://doi.org/10.3390/su11195367

16. Işik H., Sintunavarat W. An investigation of the common solutions for coupled systems of functional equations arising in dynamic programming. *Mathematics.* 2019;7(10):977.
https://doi.org/10.3390/math7100977

17. Jia S., Sun J.Q., Ding Q. Flutter Control of a Two-dimensional Airfoil based on Adaptive Dynamic Programming. In: IOP Conference Series: Materials Science and Engineering. 2019;531(1): Article No. 012033.
https://doi.org/10.1088/1757-899X/531/1/012033

18. Kozlowski K.M., Sharma G.K., Chen J.J., Qi L., Osann K., Jing J.C., Ahuja G.S. Dynamic programming and automated segmentation of optical coherence tomography images of the neonatal subglottis: Enabling efficient diagnostics to manage subglottic stenosis. *J. Biomed. Opt.* 2019;24(9):096001.
https://doi.org/10.1117/1.JBO.24.9.096001

19. Durdán, M., Kačur, J., Laciak, M., Flegner, P. Thermophysical properties estimation in annealing process using the iterative dynamic programming method and gradient method. *Energies.* 2019;12(17):3267.
https://doi.org/10.3390/en12173267

20. Wang P., Peng Y., Gao X.-J., Gao H.-H. Train Speed Trajectory Optimization using Dynamic Programming with speed modes decomposition. In: IOP Conference Series: Materials Science and Engineering. 2019;569(4):042019.
https://doi.org/10.1088/1757-899X/569/4/042019

21. Ikeda S. Ooka, R. Comparison of metaheuristics and dynamic programming for district energy optimization. In: IOP Conference Series: Earth and Environmental Science. 2019:294(1):9.
https://doi.org/10.1088/1755-1315/294/1/012040

22. Narendra Kumar P.V., Chengaiah C., Prasad J.V.K. Fuzzy dynamic programming based solarthermal load scheduling of Andhra Pradesh power generation using Matlab. *International Journal of Recent Technology and Engineering (IJRTE).* 2019;8(2S8):1242-1247.
https://doi.org/10.35940/ijrte.B1046.0882S819

23. Карпов Д.А., Струченков В.И. Динамическое программирование как метод сплайн-аппроксимации в САПР линейных сооружений. *Российский технологический журнал.* 2019;7(3):77-88.
https://doi.org/10.32362/2500-316X-2019-7-3-77-88

24. Романовский И.В. Дискретный анализ. СПб,: Невский Диалект, БХВ – Петербург, 2003. 320 с. ISBN 5-7940-0114-3

25. Вентцель Е.С. Исследование операций: задачи, принципы, методология. М.: КноРус, 2010. 191 с. ISBN 978-5-406-00682-5

26. Косоруков О.А., Мищенко А.В. Исследование операций: Учебник для вузов. М.: Экзамен, 2013. 445 с. ISBN 5-94692-363-3

27. Струченков В.И. Использование параболических сплайнов в САПР линейных сооружений. *Российский технологический журнал.* 2018;6(1):40-52.
https://doi.org/10.32362/2500-316X-2018-6-1-40-52

References:

1. Bellman R. Dinamicheskoe programmirovanie (Dynamic programming). Moscow: Inostrannaya literatura Publ., 1960. 402 p. (in Russ.).

2. Lyakhovskii V.N., Mikhalevich V.S., Bykov V.I. Determination on the computer of the most advantageous position of the red line of the longitudinal profile on the free run. *Transportnoe stroitel'stvo = Transport Construction.* 1962;4:41-43.

3. Mikhalevich V.S., Shor N.Z. The mathematical foundations of solving the problem of choosing the optimal outline of the longitudinal profile. *Trudy Vsesoyuznogo NII transportnogo stroitel'stva = Proceedings of the Institute of Transport Construction.* 1964;51:12-24 (in Russ.).

4. Mikhalevich V.S. Sequential optimization algorithms and their application. *Kibernetika*. 1965;1:45-56 (in Russ.).

5. Mikhalevich V.S., Bykov V.I., Sibirko A.N. To the question of designing the optimal longitudinal profile of the road. *Transportnoe stroitel'stvo = Transport Construction.* 1975;6:39-40 (in Russ.).

6. Kosmin V.V., Struchenkov V.I., Fradkov E.B. Computeraided longitudinal road profile design. *Transportnoe stroitel'stvo = Transport Construction.* 1971;4:38-42 (in Russ.).

7. Bellman R., Dreifus S. *Prikladnye zadachi dinamicheskogo programmirovaniya* (Applied problems of dynamic programming). Moscow: Nauka; 1965. 460 p. (in Russ.).

8. Lezhnev A.V. *Dinamicheskoe programmirovanie v ekonomicheskikh zadachakh* (Dynamic programming in economic problems). Moscow: Binom; 2016. 176 p.

9. Kremer N.Sh. *Issledovanie operatsii v ekonomik*e (The study of operations in the economy). Moscow: Yurait; 2012. 430 p. ISBN 978-5-9916-1849-6

10. Cavagnari G., Marigonda A., Piccoli B. Generalized dynamic programming principle and sparse meanfield control problems. *J. Math. Anal. Appl.* 2020;481(1): Article No. 123437.
https://doi.org/10.1016/j.jmaa.2019.123437

11. Ramahatana F., David M. Economic optimization of micro-grid operations by dynamic programming with real energy forecast. *J. Phys. Conf.* 2019;1343(1): Article No. 012067.
https://doi.org/10.1088/1742-6596/1343/1/012067

12. Firdausiyah N., Taniguchi E., Qureshi A.G. Impacts of Urban Consolidation Centres for Sustainable City Logistics Using Adaptive Dynamic Programming Based MultiAgent Simulation. In: IOP Conference Series: Earth and Environmental Science. 2019;328(1): Article No. 012071.
https://doi.org/10.1088/1755-315/328/1/012071

13. He S., Shin H.-S., Tsourdos A. Computational guidance using sparse Gauss-Hermite quadrature differential dynamic programming. *IFAC-PapersOnLine*. 2019;52(12):13-18.
https://doi.org/10.1016/j.ifacol.2019.11.062

14. He S., Guo S., Chen, K., Deng L., Liao Z., Xiong F., Yin J. Dataset for reservoir impoundment operation coupling parallel dynamic programming with importance sampling and successive approximation. *Data in Brief.* 2019;26: Article No. 104440.
https://doi.org/10.1016/j.dib.2019.104440

15. Fayaed S.S., Fiyadh S.S., Khai W.J., Ahmed A.N., Afan H.A., Ibrahim R.K. Improving dam and reservoir operation rules using stochastic dynamic programming and artificial neural network integration model. *Sustainability.* 2019;11(19):5367.
https://doi.org/10.3390/su11195367

16. Işik H., Sintunavarat W. An investigation of the common solutions for coupled systems of functional equations arising in dynamic programming. *Mathematics.* 2019;7(10):977.
https://doi.org/10.3390/math7100977

17. Jia S., Sun J.Q., Ding Q. Flutter Control of a Two-dimensional Airfoil based on Adaptive Dynamic Programming. In: IOP Conference Series: Materials Science and Engineering. 2019;531(1): Article No. 012033. https://doi.org/10.1088/1757-899X/531/1/012033

18. Kozlowski K.M., Sharma G.K., Chen J.J., Qi L., Osann K., Jing J.C., Ahuja G.S. Dynamic programming and automated segmentation of optical coherence tomography images of the neonatal subglottis: Enabling efficient diagnostics to manage subglottic stenosis. *J. Biomed. Opt.* 2019;24(9):096001.
https://doi.org/10.1117/1.JBO.24.9.096001

19. Durdán, M., Kačur, J., Laciak, M., Flegner, P. Thermophysical properties estimation in annealing process using the iterative dynamic programming method and gradient method. Energies. 2019;12(17):3267.
https://doi.org/10.3390/en12173267

20. Wang P., Peng Y., Gao X.-J., Gao H.-H. Train Speed Trajectory Optimization using Dynamic Programming with speed modes decomposition. In: IOP Conference Series: Materials Science and Engineering. 2019;569(4):042019.
https://doi.org/10.1088/1757-899X/569/4/042019

21. Ikeda S. Ooka, R. Comparison of metaheuristics and dynamic programming for district energy optimization. In: IOP Conference Series: Earth and Environmental Science. 2019:294(1):9.
https://doi.org/10.1088/1755-1315/294/1/012040

22. Narendra Kumar P.V., Chengaiah C., Prasad J.V.K. Fuzzy dynamic programming based solarthermal load scheduling of Andhra Pradesh power generation using Matlab. *International Journal of Recent Technology and Engineering (IJRTE).* 2019;8(2S8):1242-1247.
https://doi.org/10.35940/ijrte.B1046.0882S819

23. Karpov D.A., Struchenkov V.I. Dynamic Programming as a Method of Spline Approxi-mation in the CAD Systems of Linear Constructions. *Rossiiskii tekhnologicheskii zhurnal = Russian Technological Journal.* 2019;7(3):77-88 (in Russ.).
https://doi.org/10.32362/2500-316X-2019-7-3-77-88

24. Romanovskii I.V. *Diskretnyi analiz* (Discrete analysis). St. Petersburg: Nevskii Dialekt, BKhV–Peterburg; 2003. 320 p. (in Russ.). ISBN 5-7940-0114-3

25. Venttsel' E.S. *Issledovanie operatsii: zadachi, printsipy, metodologiya* (Operations research: tasks, principles, methodology). Moscow: KnoRus; 2010. 191 p. (in Russ.). ISBN 978-5-406-00682-5

26. Kosorukov O.A., Mishchenko A.V. *Issledovanie operatsii: Uchebnik dlya vuzov* (Operations Research). Moscow: Ekzamen; 2013. 445 p. (in Russ.). ISBN 5-94692-363-3

27. Struchenkov V.I. The use of parabolic splinees in CAD of linear structures. *Rossiiskii tekhnologicheskii zhurnal = Russian Technological Journal.* 2018;6(1):40-52 (in Russ.).
https://doi.org/10.32362/2500-316X-2018-6-1-40-52

Об авторах:

Карпов Дмитрий Анатольевич, кандидат технических наук, заведующий кафедрой общей информатики Института кибернетики ФГБОУ ВО «МИРЭА – Российский технологический университет» (119454, Россия, Москва, пр-т Вернадского, д. 78).

Струченков Валерий Иванович, доктор технических наук, профессор кафедры общей информатики Института кибернетики ФГБОУ ВО «МИРЭА – Российский технологический университет» (119454, Россия, Москва, пр-т Вернадского, д. 78).

About the authors:

Dmitry A. Karpov, Cand. Sci. (Engineering), Head of the Department of General Informatics of the Institute of Cybernetics, MIREA – Russian Technological University (78, Vernadskogo pr., Moscow 119454, Russia).

Valeriy I. Struchenkov, Dr. Sci. (Engineering), Professor of the Department of General Informatics of the Institute of Cybernetics, MIREA – Russian Technological University (78, Vernadskogo pr., Moscow 119454, Russia).

Поступила: 26.12.2019; Получена после доработки: 20.01.2020; Принята к опубликованию: 06.07.2020.