Динамические системы, вып. 22 (2007), 21–36

ДИНАМИЧЕСКИЕ СИСТЕМЫ
Межведомственный научный сборник

УДК 519.68

Методологические аспекты динамического программирования

О.А. Щербина
University of Vienna,
Vienna 1090, Austria. *E-mail: oleg.shcherbina@univie.ac.at*

Аннотация. Рассмотрены методологические аспекты динамического программирования, в том числе анализируются основные графовые интерпретации динамического программирования, такие, как блочные диаграммы, выделение бесконтурных орграфов, лежащих в основе вычислительной процедуры динамического программирования, а также представление структуры задачи динамического программирования с помощью графа взаимосвязей. Описана классификация задач динамического программирования на основе анализа бесконтурных орграфов процедуры динамического программирования на сериальные и несериальные задачи, на монадические и полиадические задачи. Приведены примеры классификации задач динамического программирования.

1. Введение

Динамическое программирование (ДП) является чрезвычайно мощной алгоритмической парадигмой оптимизации последовательных процессов принятия решений, имеющей декомпозиционную природу. ДП, более чем другие оптимизационные подходы, обеспечивает общую схему анализа многих типов задач. Алгоритмическая схема ДП состоит в погружении решаемой сложной задачи в параметризованное семейство задач (иногда называемых подзадачами) и последующем решении этих подзадач, используя принцип оптимальности Беллмана и вытекающее из него рекуррентное уравнение Беллмана. **Принципом оптимальности** называется следующая интуитивная идея, предложенная R. Bellman [5]:

> Любая оптимальная стратегия имеет то свойство, что каково бы ни было текущее состояние и решение, последующие решения должны представлять собой оптимальную стратегию по отношению к состоянию, получающемуся в результате текущего решения.

Неотъемлемым свойством ДП является многошаговость процедуры оптимизации, в результате которой получаются и сохраняются оптимальные решения подзадач. При решении оптимизационных подзадач, порожденных в процессе решения всей

задачи с помощью ДП, могут быть использованы известные методы оптимизации, такие как линейное, нелинейное и дискретное программирование.

О происхождении самого названия "динамическое программирование (dynamic programming)" Dasgupta, Papadimitriou, Vazirani пишут [7]:

> Происхождение термина **динамическое программирование** имеет очень мало общего с программированием на компьютере. Этот термин впервые был введен Ричардом Беллманом в 1950-х годах, во времена, когда "компьютерное программирование было эзотерическим занятием, которым занималось настолько мало людей, что оно даже не имело имени". В те времена программирование означало "планирование" и под "динамическим программированием" подразумевался оптимальный многошаговый процесс планирования.

Параметрическое семейство подзадач, порождаемое в процедуре ДП, должно обладать следующими основными свойствами:

- **оптимальность подструктур**: оптимальное решение исходной задачи содержит оптимальные решения подзадач семейства (например, подпути рассматриваемой ниже классической задачи о кратчайшем пути являются также кратчайшими).

- **перекрывающиеся подзадачи**: семейство подзадач имеет перекрывающиеся подзадачи, т.е. одни и те же подзадачи используются для решения различных (обычно больших) подзадач.

В современных учебниках по исследованию операций описываются только ставшие классическими сериальные задачи ДП, более сложные несериальные динамические системы, обладающие разветвлениями и контурами, не рассматриваются. В то же время несериальные динамические системы имеют интересные приложения в задачах оптимизации водных систем и газопроводов [9]. Более того, создается впечатление, что сериальное ДП и несериальное ДП (НСДП) используют различные подходы: если сериальное ДП состоит в погружении решаемой задачи в параметризованное семейство подзадач (что требует известного "инсайта") и последующем решении этих задач, используя рекуррентное уравнение Беллмана, то несериальное ДП использует граф взаимосвязей задачи дискретной оптимизации и последовательно элиминирует переменные. Графовая интерпретация задачи динамического программирования описывается обычно в виде поиска кратчайшего пути в сети. Многие аспекты ДП остаются и сейчас не до конца ясными, в том числе возможности несериального ДП (НСДП), разница между сериальными и несериальными задачами, классификация задач ДП.

В настоящей работе предлагается использование графической интерпретации алгоритма ДП в виде ориентированного графа (орграфа) без контуров и унифицированный подход к решению задач ДП на основе анализа этого орграфа.

2. Графовые структуры в динамическом программировании

2.1. Блочные диаграммы в процедуре динамического программирования

Использование блочных функциональных диаграмм является одной из графовых интерпретаций и является одним из стандартных способов анализа и оптимизации дискретных динамических систем. Представление задачи ДП в виде блочных диаграмм часто используется для визуализации структуры задачи. n-шаговый последовательный процесс принятия решений состоит из трех компонентов: множества **шагов** (или **этапов**) $i = 1, \ldots, N$, множества **состояний** и множества **решений**. Описание многошагового процесса принятия решения с помощью диаграмм показано на рис. 1. В диаграммах шаги представлены схематически с помощью пронумерованных прямоугольников, стрелки показывают входы и выходы шагов. i-му шагу соответствуют **переменная состояния** s_i, **решающая переменная** d_i, **выход** $r_i = f_i(d_i, s_i)$, называемый **обобщенным доходом** шага i. Преобразование $s_{i+1} = t_i(d_i, s_i)$ называется **функцией перехода**.

Сериальная оптимизационная задача для дискретной динамической системы может быть записана в виде

$$\max_{\{s_1,\ldots,s_N;\ d_1,\ldots,d_{N-1}\}} \sum_{i=1}^{N-1} f_i(d_i, s_i) + f_N(s_N) \tag{2.1}$$

при ограничениях

$$s_{i+1} = t_i(d_i, s_i), \quad i = 1, \ldots, N-1. \tag{2.2}$$

$$d_i \in D_i, \quad i = 1, \ldots, N. \tag{2.3}$$

Блочная диаграмма и граф взаимосвязей этой задачи показаны на рис. 1 и рис. 6 соответственно. Эта модель представляет собой дискретный аналог задачи опти-

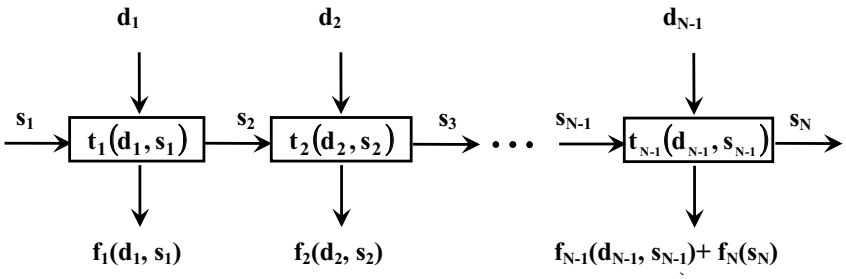

Рис. 1. Блочно-диаграммное представление задачи динамического программирования.

мального управления следующего вида:

$$\max_{U(t)} \int_{t_1}^{t_2} F(t,\ X,\ U)dt \tag{2.4}$$

при ограничениях

$$X'_t = g(t,\ X,\ U),\quad t_1 \le t \le t_2, \tag{2.5}$$

$$X(t_1) = X_1. \tag{2.6}$$

2.2. Бесконтурные орграфы в процедуре динамического программирования

ДП рассматривает исходную задачу в виде семейства взаимозависимых задач, которые решаются и результаты их решения используются при решении больших задач, пока исходная задача не будет решена. Решение любой подзадачи из семейства зависит от решений одной или нескольких подзадач на предшествующих уровнях. Зависимости между задачами в процедуре ДП может быть представлена в виде **бесконтурного орграфа** (directed acyclic graph (DAG)), который в дальнейшем называть просто орграфом. Зачастую этот орграф задан неявно. Его вершинами являются подзадачи, выделенные процедурой ДП, а ребрами – (информационные) зависимости между подзадачами: если подзадача B требует для своего решения информацию о решении подзадачи A, это может быть графически изображено с помощью ребра от A к B (рис. 2). Отсюда видно, как можно

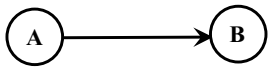

Рис. 2. Предшествование подзадач A и B.

выделить орграф, лежащий в основе алгоритмической схемы ДП, ставя в соответствие подзадачам, которые необходимо решить, вершины орграфа, а отношениям предшествования подзадач – ребра орграфа. Если имеются вершины $u_1, \ldots, u_k$, от которых стрелки указывают на v, то это означает: *подзадача v может быть решена лишь после нахождения решений для подзадач* $u_1, \ldots, u_k$ (рис. 3). Для иллюстрации преимуществ использования орграфов в виде основы вычислительной схемы ДП рассмотрим последовательность чисел Фибоначчи, определенную следующим образом:

$$F_0 = 1,\ F_1 = 1,\ F_N = F_{N-1} + F_{N-2},\ N \ge 2.$$

Нетрудно заметить, что вычисление $F_3 = F_1 + F_2$ и $F_4 = F_2 + F_3$ включает вычисление F_2. В связи с тем, что для вычисления F_5 необходимо знать F_3 и F_4, наивный подход к вычислению F_5 мог бы дважды использовать вычисление F_2, попусту теряя время на повторное вычисление решений подзадач, которые уже были решены.

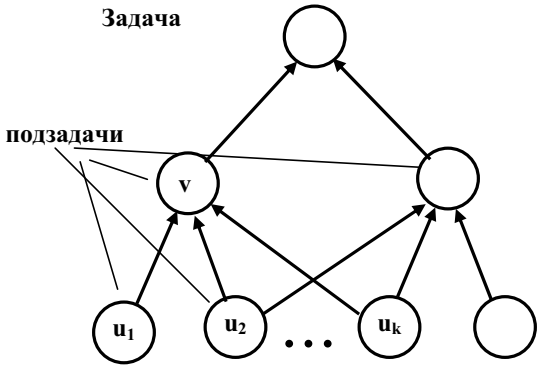

Рис. 3. Орграф подзадач схемы динамического программирования.

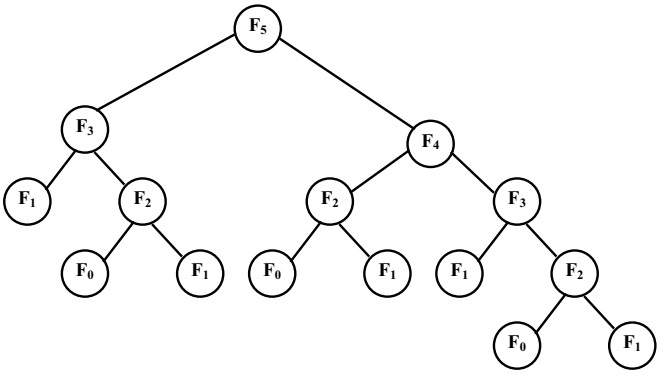

Рис. 4. Дерево обращений к задачам при вычислении пятого числа Фибоначчи. Это дерево имеет экспоненциальное число вершин.

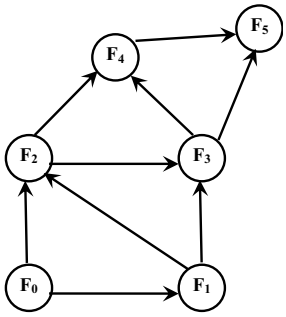

Рис. 5. Орграф процесса вычисления пятого числа Фибоначчи с линейным числом вершин.

На рис. 4 показано дерево обращений к подзадачам при вычислении F_5. Следует отметить, что одни и те же задачи решаются неоднократно. Чтобы избегнуть повторных вычислений, имеет смысл сохранять найденные решения. В этом случае, если возникает необходимость повторного решения задачи позже, можно просто найти и использовать ранее вычисленное и сохраненное решение. Этот подход называется **мемоизацией**. На рис. 5 показан орграф вычислительного процесса пятого числа Фибоначчи.

ДП использует мемоизацию, т.е. решает каждую подзадачу лишь один раз и сохраняет ее решение в таблице, что позволяет избегнуть излишней работы по повторному вычислению решений подзадач, которые были уже решены ранее.

2.3. Графовая интерпретация Бертеле–Бриоши процедуры динамического программирования

В работах Mitten, Nemhauser, Aris и Wilde ([3], [4], [13], [15]) классическое сериальное ДП было обобщено для несериальных динамических систем, содержащих контуры и разветвления, причем в этих работах использовано графическое представление динамической системы в виде блочных диаграмм.

Более общий и перспективный теоретико-графовый подход к решению несериальных оптимизационных задач был предложен в работах Bertele & Brioschi [6].

Рассмотрим несериальную задачу дискретной оптимизации (ДО) с ограничениями следующего вида:

$$F(x_1, x_2, \ldots, x_n) = \sum_{k \in K} f_k(Y^k) \to \max \tag{2.7}$$

при ограничениях

$$g_i(X^i)\ R_i\ 0, \quad i \in M = \{1, 2, \ldots, m\}, \tag{2.8}$$

$$x_j \in D_j, \quad j \in \{1, \ldots, n\}, \tag{2.9}$$

где

$$Y^k \subseteq \{x_1, x_2, \ldots, x_n\}, \quad k \in K = \{1, 2, \ldots, t\}\,; \tag{2.10}$$

$$X^i \subseteq \{x_1, x_2, \ldots, x_n\}, \quad R_i \in \{\leq, =, \geq\}, \quad i \in M = \{1, 2, \ldots, m\}. \tag{2.11}$$

Определение 1. Две переменные $x \in X$ и $y \in Y$ **взаимосвязаны** в несериальной задаче ДО с ограничениями, если они появляются вместе в одном компоненте целевой функции (ЦФ) или в одном и том же ограничении (другими словами, если переменные входят одновременно во множество X^i или во множество Y^k).

Введем графовую интерпретацию несериальной задачи ДО в виде графа взаимосвязей ([6], [2]), естественным образом представляющего структуру оптимизационной задачи.

Определение 2. [6]. **Графом взаимосвязей** несериальной задачи ДО (без ограничений или с ограничениями) называется неориентированный граф $G = (V, X)$, для которого

1. множество вершин X графа соответствует множеству переменных задачи ДО;

2. две вершины графа смежны тогда и только тогда, когда соответствующие им переменные взаимосвязаны.

Определение 3. Множество переменных, взаимосвязанных с переменной $x \in X$, обозначается $Nb(x)$ и называется **окрестностью** переменной x.

Рассмотрим задачу ДО с ограничениями и предположим без потери общности, что порядок элиминации переменных следующий: $x_1, \ldots, x_n$. Опишем процедуру элиминации для переменной x_1. Переменная x_1 входит в подмножество K_1 компонентов ЦФ: $K_1 = \left\{k \mid x_1 \in Y^k\right\}$ и во подмножество ограничений U_1: $U_1 = \{i \mid x_1 \in X^i\}$.

Одновременно с x_1 в компоненты ЦФ $Y^k,\ k \in K_1$ и в ограничения U_1 входят переменные из окрестности $Nb(x_1)$.

Переменной x_1 соответствует следующая подзадача P_1 исходной задачи ДО:

$$h_1(Nb(x_1)) = \max_{x_1} \left\{ \sum_{k \in K_1} f_k(Y^k) \mid g_i(X^i)\ R_i\ 0,\ i \in U_1,\ x_j \in D_j,\ x_j \in Nb(x_1) \right\}.$$

Исходная задача ДО может быть преобразована следующим образом:

$$\max_{x_1,\ldots,x_n} \left\{ \sum f_k(Y^k) \mid g_i(X^i)\ R_i\ 0,\ \ i \in M,\ x_j \in D_j,\ j \in N \right\} =$$

$$= \max_{x_1,\ldots,x_{n-1}} \left\{ \sum_{k \in K - K_1} f_k(Y^k) + h_1(Nb(x_1)) \mid g_i(X^i)\ R_i\ 0,\ i \in M - U_1, \right.$$

$$\left. x_j \in D_j,\ j \in X - \{x_1\} \right\}.$$

В новой задаче $n-1$ переменная; по сравнению с исходной задачей в ней исключены ограничения с индексами из U_1 и компоненты ЦФ $\sum_{k \in K_1} f_k(Y^k)$, но появился новый компонент ЦФ $h_1(Nb(x_1))$.

Процедура элиминации элиминирует оставшиеся переменные одну за другой аналогичным образом. При этом необходимо запоминать таблицы с оптимальными частичными решениями на каждом шаге процесса.

На шаге n описанного процесса элиминируется переменная x_n и находится оптимальное значение ЦФ. Затем нужно выполнить обратную часть процедуры ДП для нахождения оптимального решения.

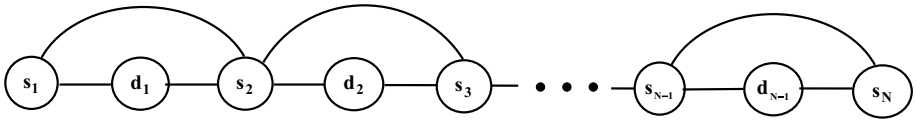

Рис. 6. Эквивалентное графовое представление многошагового процесса принятия решений.

Решение задачи ДО с помощью НСДП подробно описано в [1], [2]. Процесс преобразования графа взаимосвязей, соответствующий процедуре НСДП, известен как элиминационный процесс [6]. Описанная выше задача (2.1)–(2.3) имеет граф взаимосвязей (рис. 6) и может быть решена с помощью элиминации переменных [2] в очередности $\alpha = (s_N, d_{N-1}, s_{N-1} \ldots,\ d_1, s_1)$.

3. Классификация задач динамического программирования

Li & Wah [11], [14] предложили классификацию задач ДП на основе характера зависимостей между подзадачами и функцией композиции. Если решение задачи использует решение единственной задачи предыдущих уровней, то формулировка задачи ДП является **монадической**, иначе она называется **полиадической**. В орграфе без контуров вершины могут быть разбиты на уровни, так что задачи на данном уровне зависят только от подзадач предыдущих уровней. В зависимости от связей задач на разных уровнях, формулировка задачи ДП может быть отнесена либо к **сериальным**, либо к **несериальным** задачам. Если задачи семейства на всех уровнях зависят только от результатов решения задач, находящихся на непосредственно предшествующем уровне, то такая задача ДП будет называться **сериальной** задачей ДП, иначе она называется **несериальной** задачей ДП. Используя эти два критерия классификации, можно относить задачи ДП к одному из следующих видов:

- сериально-монадическая,
- сериально-полиадическая,
- несериально-монадическая,
- несериально-полиадическая.

3.1. Монадическая и полиадическая формулировки

Формулировка задачи ДП называется **монадической**, если ее рекуррентное уравнение содержит лишь один рекурсивный член, иначе она называется **полиадической**. Это отличие показано на примере поиска кратчайшего пути в сети. Обозначим c_{ij} — длина ребра (i, j). Длина пути от источника s до пункта назначения (стока) t равна сумме длин ребер пути. Пусть $f_1(i)$ — минимальная длина

пути из i до t. Тогда длина пути из i до t через соседнюю вершину j составляет $c_{ij} + f_1(j)$. Для нахождения $f_1(i)$ необходимо сравнить пути, проходящие через все возможные промежуточные вершины. Таким образом,

$$f_1(i) = min_j[c_{ij} + f_1(j)]. \tag{3.1}$$

Рекуррентное уравнение (3.1) является монадическим, так как включает лишь один рекуррентный член $f_1(i)$.

Рассмотрим более общую задачу поиска оптимального пути из любой вершины i до любой другой вершины j. Можно свести эту задачу к поиску такой промежуточной вершины k, чтобы кратчайшие пути из i до k и из k до j составляли кратчайший путь из i до j. Рекуррентное уравнение имеет вид:

$$f_3(i, j) = min_k[f_3(i, k) + f_3(k, j)], \tag{3.2}$$

где функция $f_3(i, j)$ — минимальная длина пути из i до j. Эта функция полиадическая, так как она содержит более, чем один рекуррентный член. Для полиадических формулировок принцип оптимальности Беллмана может быть обобщен добавлением фразы о том, что "все подпоследовательности оптимальной стратегии также оптимальны" [10]. Например, согласно уравнению (3.2), если известно, что кратчайший путь из i до j проходит через k, то подпуть из i до k этого оптимального пути должен быть оптимальным по отношению ко всем подпутям из i до k; то же справедливо для подпути из k до j.

3.2. Сериальные и несериальные формулировки

Разница между сериальной и несериальной формулировками задач ДП основана как на виде их целевых функций, так так и на типе рекурсии. По виду целевой функции, задача оптимизации называется сериальной, если каждый компонент целевой функции имеет одну общую переменную с предыдущим компонентом (за исключением первого компонента) и другую общую переменную с последующим компонентом (за исключением последнего компонента), иначе задача называется несериальной. Сериальная задача ДО имеет граф взаимосвязей с сериальной структурой (рис. 7). Рассмотрим задачу ДП со следующей целевой фунцией

Рис. 7. Граф взаимосвязей сериальной задачи.

$$\max_X \sum_{i=1}^{N-1} g_i\left(x_i, x_{i+1}\right), \tag{3.3}$$

где X — множество дискретных переменных $\{x_i, \ldots, x_N\}$. В уравнении (3.3) каждый компонент целевой функции содержит две переменные, связанные лишь с переменными в двух других компонентах, причем взаимосвязи являются сериальными. Так, компонент $g_i(x_i, x_{i+1})$ связан только с компонентами $g_i(x_{i-1}, x_i)$ посредством переменной x_i и $g_{i+1}(x_{i+1}, x_{i+2})$ посредством переменной x_{i+1}. В результате, уравнение (3.3) является сериальной задачей оптимизации.

ЦФ несериальной задачи ДО имеет следующий вид:

$$f(X) = \otimes_{l=1}^{k} g_l(X^l), \tag{3.4}$$

где $X = \{x_1, \ldots, x_N\}$ — множество дискретных переменных, $X^l \subset X$, $\otimes$ — монотонная функция, объединяющая функции g_l вместе (например, сумма или произведение).

4. Примеры задач динамического программирования

В описанных ниже примерах задач ДП выделяются орграфы, лежащие в основе вычислительного процесса, и произведена классификация задач.

4.1. Задача о кратчайшем пути

Нахождение кратчайшего пути в сети является известной задачей комбинаторной оптимизации [8], [12]. ДП и задача о кратчайшем пути очень схожи. Как показано выше, рекуррентные соотношения ДП могут быть представлены как задачи поиска оптимального пути в орграфе подзадач ДП. Решение оптимизационной задачи – последовательность состояний и решений, определяющих путь в сети (рис. 6), начинающийся в начальном состоянии s_1 и оканчивающийся в конечном состоянии s_N.

Рассмотрим бесконтурный орграф $G = (V, E)$, состоящий из непустого конечного множества вершин V и множества дуг $E \subseteq \{(u, v) | u, v \in V,\ u \neq v\}$. Сетью $N = (G, l)$ называется орграф G с заданной на нем вещественной функцией $l : E \to R$. Действительное число $l(u, v)$ называется длиной дуги $(u, v) \in E$. Путь из u до v в G — это конечная последовательность $q_{uv} = [u = v_1, v_2, \ldots, v_k = v]$ вершин G, для которой $(v_i, v_{i+1}) \in E$ при $i = 1, 2, \ldots, k-1$. Длина пути q_{uv} равна:

$$L(q_{uv}) = \sum_{i=1}^{k-1} l(v_i, v_{i+1}).$$

Кратчайшим путем из u в v в сети N является путь q_{uv}, для которого длина $L(q_{uv})$ минимальна среди длин всех путей из u до v.

Рассмотрим задачу нахождения кратчайшего пути между вершинами s и t в орграфе (рис. 8). В данно случае орграф вычислительного процесса задан; он совпадает с данным орграфом G. Обозначим через $h(v)$ длину кратчайшего пути от s до v, где v — произвольная вершина сети. Начальные значения

$$h(v) = \begin{cases} 0, & \text{если } v = s; \\ \infty, & \text{в противном случае.} \end{cases}$$

Рекуррентное уравнение Беллмана связывает вычисление различных функций

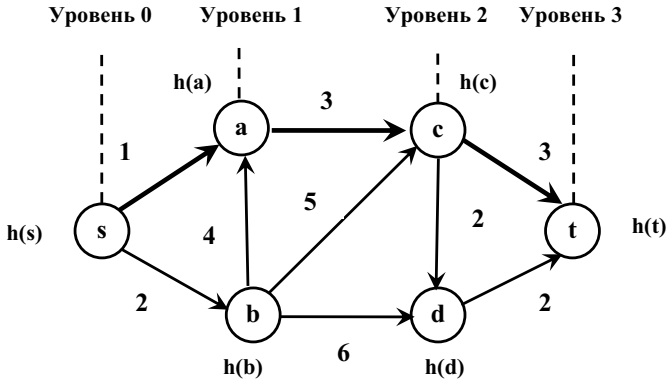

Рис. 8. Кратчайший путь в орграфе.

$h(v)$:

$$h(v) = \min_{u:(u,v)\in E}\{h(u) + l(u,v)\}, \tag{4.1}$$

где $l(u,v)$ – расстояние между вершинами u и v. Обозначим через $pr(v)$ вершину, предшествующую вершине v в оптимальном пути: $pr(v) = u^* : h(u^*) + l(u^*, v) = \min_u(h(u) + l(u,v))$. Решим подзадачи из семейства задач $\{h(v),\ v \in V\}$, начиная с $h(s)$ (рис. 8).

$$h(b) = h(s) + l(s,b) = 0 + 2 = 2; pr(b) = s;$$

$$h(a) = \min\left(h(s) + l(s,a),\ h(b) + l(b,a)\right) = \min\left(0+1,\ 2+4\right) = 1;\ pr(a) = s;$$

$$h(c) = \min\left(h(a) + l(a,c),\ h(b) + l(b,c)\right) = \min\left(1+3,\ 2+5\right) = 4;\ pr(c) = a;$$

$$h(d) = \min\left(h(b) + l(b,d),\ h(c) + l(c,d)\right) = \min\left(2+6,\ 4+2\right) = 6;\ pr(d) = c;$$

$$h(t) = \min\left(h(c) + l(c,t),\ h(d) + l(d,t)\right) = \min\left(4+3,\ 6+2\right) = 7;\ pr(t) = c;$$

Таким образом, длина кратчайшего пути равна 7. Обратная часть процедуры ДП позволяет найти кратчайший путь: $t,\ pr(t) = c,\ pr(c) = a,\ pr(a) = s$. Кратчайший путь: $[s,\ a,\ c,\ t]$.

Анализируя орграф рис. 8, можно сделать вывод, что задача о кратчайшем пути в орграфе является сериальной полиадической задачей ДП, так как задачи из каждого уровня связаны лишь с задачами из предыдущего уровня (уровни на рисунке показаны пунктиром).

4.2. Задача о ранце

Рассмотрим **бинарную задачу о ранце**

$$\max\{\sum_{j=1}^{n} c_j x_j | \sum_{j=1}^{n} w_j x_j \le W,\ x_j \in \{0,\ 1\},\ j = 1, \dots, n.\}$$

Задача о ранце может быть решена с помощью погружения исходной задачи в параметризованное семейство подзадач:

$$h(k, w) = \max\{\sum_{j=1}^{k} c_j x_j | \sum_{j=1}^{k} w_j x_j \le w,\ x_j \in \{0,\ 1\},\ j = 1, \dots, k, \}$$

где $h(0, 0) = 0, k = 0, \dots, n;\ w = 0, \dots, W.$

Задачи этого семейства $\{h(k, w)\}$ связаны с помощью рекуррентного соотношения

$$h(k + 1, w) = \max\{h(k, w),\ h(k, w - w_j) + c_{k+1}\},\ k = 0, \dots, n - 1, \qquad (4.2)$$

где $h(0, 0) = 0.$

Вычисление функции $h_k(k, w)$ – это подзадача вычислительного процесса ДП. Чтобы выявить орграф, лежащий в основе процедуры ДП, представим подзадачи в виде графа, в котором вершинами будут (k, w). Рекуррентное уравнение Беллмана 4.2 имеет графовое представление (рис. 9). Ребро из (k, w) до $(k+1, w)$ соответству-

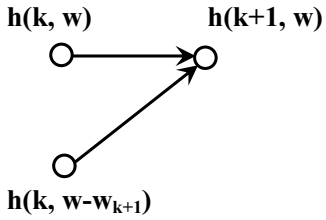

Рис. 9. Подзадачи рекуррентного уравнения задачи о ранце.

ет переменной x_{k+1}, принимающей значение 0, а ребро из (k, w) до $(k+1, w-w_{k+1})$ — переменной x_{k+1}, принимающей значение 1. На рис. 10 показан орграф вычислительного процесса ДП для задачи о ранце с ограничением

$$2x_1 + 3x_2 + x_3 \le 4.$$

Согласно классификации формулировок задач ДП, описанной в разделе 3, задача о ранце является монадической сериальной формулировкой ДП.

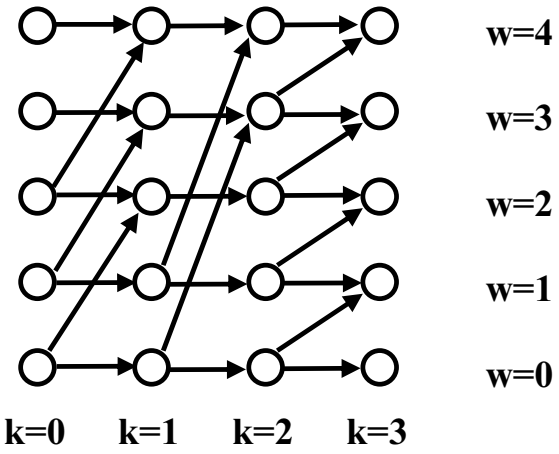

Рис. 10. Орграф подзадач задачи о ранце.

4.3. Задача умножения матриц

Предположим, что нам нужно перемножить 4 матрицы $A_1 \times A_2 \times A_3 \times A_4$, размерности которых 50×20, 20×1, 1×10, и 10×100, соответственно. Возникает проблема оптимальной организации процесса вычислений, перемножая за один раз две матрицы. Умножение матриц ассоциативно: $A_1 \times (A_2 \times A_3) = (A_1 \times A_2) \times A_3$. Можно вычислить произведение четырех матриц разными способами, в зависимости от порядка расстановки скобок. Умножение матрицы $m \times n$ на матрицу $n \times p$ требует $m \cdot n \cdot p$ умножений. Используя эту оценку, можно сравнить по трудоемкости различные способы вычисления $A_1 \times A_2 \times A_3 \times A_4$ (табл.1 [7]):
Таблица 1. Трудоемкость вычисления произведения матриц различными способами

Способ расстановки скобок	Расчет трудоемкости вычисления	Трудоемкость
$A_1 \times ((A_2 \times A_3) \times A_4)$	$20 \cdot 1 \cdot 10 + 20 \cdot 10 \cdot 100 + 50 \cdot 20 \cdot 100$	120200
$(A_1 \times (A_2 \times A_3)) \times A_4)$	$20 \cdot 1 \cdot 10 + 50 \cdot 20 \cdot 10 + 50 \cdot 10 \cdot 100$	60200
$(A_1 \times A_2) \times (A_3 \times A_4)$	$50 \cdot 20 \cdot 1 + 1 \cdot 10 \cdot 100 + 50 \cdot 1 \cdot 100$	7000

Для задачи умножения n матриц $A_1, A_2, \ldots, A_n$, где матрица A_i имеет размерность p_i, рассмотрим структуру оптимального решения. Оптимальная расстановка скобок в произведении $(A_1 \cdot A_2 \cdot \ldots \cdot A_n)$ разбивает произведение между A_k и A_{k+1} для некоторого целого k, $1 \leq k < n$: $(A_1 \cdot A_2 \cdot \ldots \cdot A_k) \cdot (A_{k+1} \cdot \ldots \cdot A_n)$. Предпосылкой применения ДП к задаче умножения матриц является следующее наблюдение: подцепь $(A_1 \cdot A_2 \cdot \ldots \cdot A_k)$ внутри оптимальной расстановки скобок в произведении $(A_1 \cdot A_2 \cdot \ldots \cdot A_n)$ должна быть оптимальной расстановка скобок в $A_1 \cdot A_2 \cdot \ldots \cdot A_k$ (подобное свойство справедливо и для второй подцепи).

Введем $m[i, j]$ – минимальное число скалярных умножений, требуемых для вы-

численния матричного произведения $(A_i \cdot \ldots \cdot A_j)$. Выведем рекуррентное уравнение Беллмана:

$$m[i,j] = \begin{cases} 0, & \text{если } i = j; \\ \min_{i \le k < j}\{m[i,k] + m[k+1,j] + p_{i-1}p_j p_k\}, & \text{если } i < j \end{cases}$$

Анализ орграфа на рис. 11 позволяет сделать вывод, что задача умножения мат-

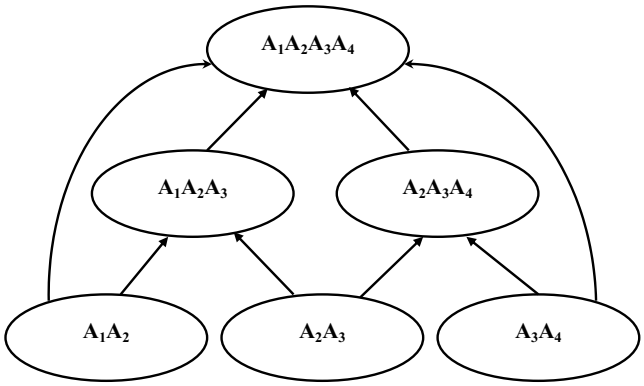

Рис. 11. Орграф вычислительного процесса для задачи о произведении матриц.

риц является полиадической несериальной задачей ДП.

4.4. Несериальная задача оптимизации

Примером несериальной задачи оптимизации может служить следующая задача:

$$\max_X\{g_1(x_1,x_2,x_4) + g_2(x_3,x_4) + g_3(x_2,x_5)\},$$

где $X = \{x_1, \ldots, x_5\}$.

Граф взаимосвязей задачи показан на рис. 12а.

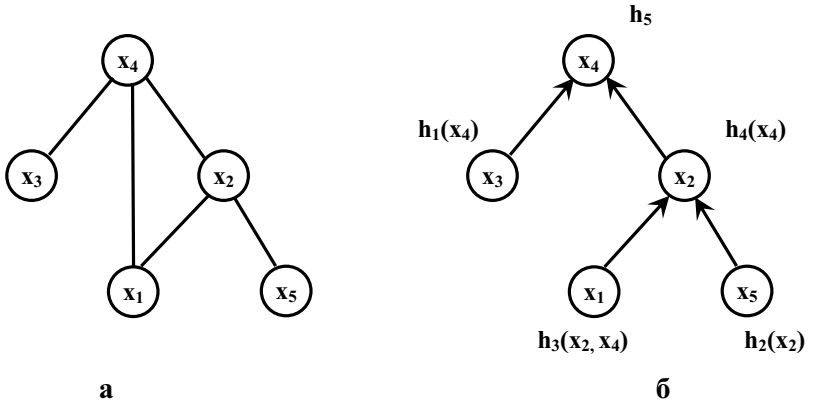

Рис. 12. Граф взаимосвязей (а) и орграф вычислительного процесса (б) для несериальной задачи оптимизации.

Для порядка элиминации $\alpha = \{x_3,\ x_5,\ x_1,\ x_2,\ x_4\}$ вычисления выполняются следующим образом (рис. 12а):

$$h_1(x_4) = \max_{x_3}[g_2(x_3,\ x_4)];$$

$$h_2(x_2) = \max_{x_5}[g_3(x_2,\ x_5)];$$

$$h_3(x_2,\ x_4) = \max_{x_1}[g_1(x_1,\ x_2,\ x_4)];$$

$$h_4(x_4) = \max_{x_2}[h_2(x_2) + h_3(x_2,\ x_4)];$$

$$h_5 = \max_{x_4}[h_4(x_4) + h_1(x_4)].$$

Орграф вычислительного процесса является ориентированным стягивающим деревом графа взаимосвязей (рис. 12б).Согласно приведенной выше классификации, данная задача является несериальной и полиадической.

Заключение

Таким образом, графовая интерпретация алгоритма ДП в виде орграфа без контуров дает возможность выработки единого подхода к анализу и классификации задач ДП. *Представляется перспективным* использование графовых интерпретаций при решении задачи динамического программирования, что позволяет использовать унифицированный подход при решении задач динамического программирования, имеющих как сериальную, так и несериальную структуру.

Список цитируемых источников

1. *Щербина О.А.* О несериальной модификации локального алгоритма декомпозиции задач дискретной оптимизации // Динамические системы. — 2005. — Вып.19. — С.179—190.
2. *Щербина О.А.* Элиминационные алгоритмы декомпозиции задач дискретной оптимизации // Таврический вестник информатики и математики. — 2006. — №2. — С. 28–41.
3. *Aris R.* The optimal design of chemical reactors. – New York: Academic Press, 1961.
4. *Aris R., Nemhauser G.L., Wilde D.* Optimization of multistage cyclic and branching system serial procedures // Journal of American Institute of Chemical Engineering. – 1964. – V. 10, N6. – P. 913–919.
5. *Bellman R., Dreyfus S.* Applied Dynamic Programming. - Princeton: Princeton University Press, 1962.
6. *Bertele U., Brioschi F.* Nonserial Dynamic Programming. — New York: Academic Press, 1972. — 235 p.
7. *Dasgupta S., Papadimitriou C.H., Vazirani U.V.* Algorithms. — McGraw Hill, 2006. — 336 p.

8. *Dreyfus S.E.* An appraisal of some shortest-path algorithms // Operations Research. – 1969. – 17. – P. 395–412
9. *Esogbue A.O., Marks B.* Non-serial dynamic programming – A survey // Operational Research Quarterly. – l974. – 25. – P.253–265.
10. *Ibaraki T.* Solvable classes of discrete dynamic programming // J. Math. Anal. Appl. - 1973. - 43. - P. 642-693.
11. *Li G.-J., Wah B.W.* Parallel processing of serial dynamic programming problems // Proceedings of COMPSAC 85. - 1985. — P. 81–89.
12. *Lawler E.L.* Combinatorial Optimization Networks and Matroids. – New York: Holt, Rinehart, and Winston, 1976.
13. *Mitten L.G., Nemhauser G.L.* Multistage optimization // Chemical Engineering Progress. – 1963. – 54. – P. 52–60.
14. *Wah B.W., Li G.-J.* Systolic processing for dynamic programming problems // Circuits Systems Signal Process. – 1988. – 7. – P.119–149.
15. *Wilde D.* Strategies for optimization macrosystems. Chemical Engineering Progress. – 1965. – V.61, N3. – P. 86–93.

Получена 29.03.07