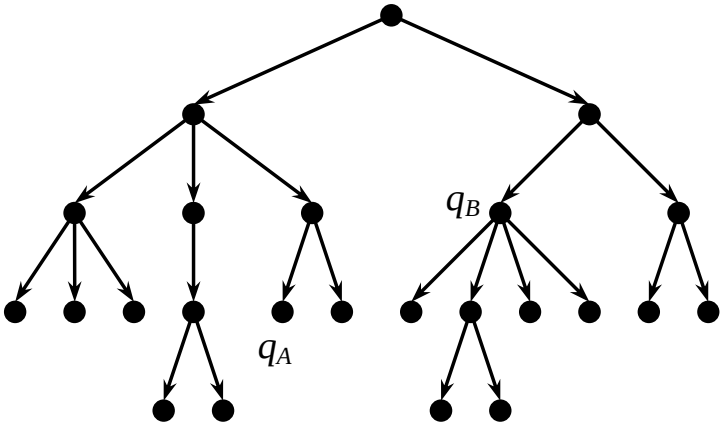
Тема 3. ИСЧЕРПЫВАЮЩИЙ ПОИСК

3.2. Метод ветвей и границ

Метод ветвей и границ является разновидностью поиска с возвратом и применяется для решения задач оптимизации. В таких задачах предполагается, что существует некоторый параметр (назовем его стоимостью), характеризующий качество решения. То есть каждое решение связано с определенной стоимостью, и необходимо найти оптимальное решение (решение с наименьшей стоимостью).

Пусть U – множество решений некоторой задачи. Обозначим через $cost(u)$ стоимость некоторого решения $u \in U$. Тогда оптимальным решением будет такое, которое дает минимальное значение $q_{\min} = \min_{u \in U} cost(u)$ стоимости.

Пусть на некотором этапе исследовано некоторое подмножество A множества U решений и определена стоимость q_A наилучшего на данный момент решения и пусть процесс поиска с возвратом достиг некоторой вершины дерева поиска. Тогда стоимость q_B соответствующего частичного решения является *нижней границей* стоимости множества B всех вытекающих из него решений (всех потомков этой вершины). При $q_B \geq q_A$ процесс продвижения по дереву прекращается, поскольку уже ясно, что все решения из множества B не лучше имеющегося, т. е. исключаются из рассмотрения (отсекаются) соответствующие ветви в дереве поиска. Если $q_B < q_A$, то можно предположить, что среди потомков этой вершины может оказаться оптимальное решение. Поэтому продолжается дальнейшее исследование (продвижение по дереву поиска). Если стоимость нового полученного решения окажется меньше q_A, то это решение сохраняется в качестве наилучшего на данный момент, а его стоимость становится новым значением q_A. Таким образом, метод ветвей и границ состоит в последовательном улучшении оценки q_A стоимости, пока не будет получено оптимальное решение со стоимостью $q_{\min}$. Эффективность метода во многом зависит от стратегии исследования решений. Следует стремиться к тому, чтобы находить решения, для которых оценки стоимости близки к оптимальному решению, на ранних этапах поиска.


q_B
q_A

Для применения метода ветвей и границ стоимость должна быть четко определена для частичных решений. Кроме того, для всех частичных решений $(a_1, a_2, \ldots, a_{k-1})$ и для всех расширений $(a_1, a_2, \ldots, a_{k-1}, a_k)$ должно выполняться соотношение

$$cost\,(a_1, a_2, \ldots, a_{k-1}) \le cost\,(a_1, a_2, \ldots, a_{k-1}, a_k),$$

где $cost\,(a_1, a_2, \ldots, a_k)$ – стоимость решения $(a_1, a_2, \ldots, a_k)$. Когда стоимость обладает указанным свойством, то можно отбросить частичное решение $(a_1, a_2, \ldots, a_k)$, если его стоимость больше или равна стоимости ранее вычисленных решений. Включение этого ограничения в общий алгоритм 3.1 поиска с возвратом дает алгоритм 3.6, где переменная *cost* – это стоимость текущего частичного решения, а *lowcost* – стоимость оптимального на данный момент решения (наименьшая на данный момент стоимость).

Для многих прикладных задач функция стоимости неотрицательная и часто удовлетворяет более сильному требованию

$$cost\,(a_1, a_2, \ldots, a_{k-1}, a_k) = cost\,(a_1, a_2, \ldots, a_{k-1}) + C(a_k),$$

где $C(a_k) \ge 0$ – функция стоимости, определенная для всех a_k. Это условие несколько упрощает общий алгоритм.

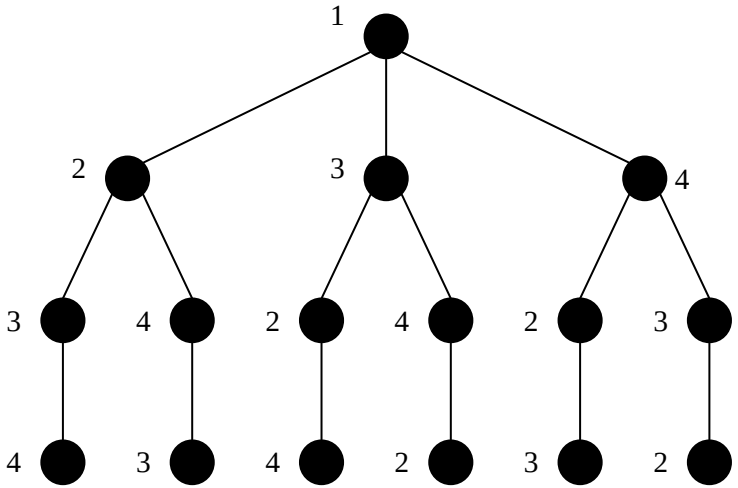
```
lowcost ← ∞
cost ← 0
определить S_1 ⊆ A_1
k ← 1
while k > 0 do
    while S_k ≠ ∅ and cost < lowcost
        do
            a_k ← элемент из S_k
            S_k ← S_k − {a_k}
            cost ← cost(a_1, a_2, …, a_k)
            if (a_1, a_2, …, a_k) – решение
              and cost < lowcost
                then
                    хранить (a_1, a_2, …, a_k)
                      как решение с наи-
                      меньшей стоимостью
                    lowcost ← cost
            k ← k + 1
            определить S_k ⊆ A_k
    k ← k − 1
    cost ← cost(a_1, a_2, …, a_{k−1})
// последнее сохраненное решение есть
// решение с наименьшей стоимостью
```

Алгоритм 3.6 Общий алгоритм метода ветвей и границ

Рассмотрим применение метода ветвей и границ на примере типичной задачи оптимизации – *задачи коммивояжера*. В этой задаче коммивояжер должен посетить n городов, побывав в каждом точно по одному разу, и возвратиться в исходный пункт. Известны стоимости перемещений между всеми парами городов (стоимость перемещения из города i в город j равна C_{ij}). Требуется определить оптимальный маршрут посещения городов, т. е. маршрут с минимальной общей стоимостью. Задача коммивояжера очевидным образом представляется в виде ориентированного взвешенного графа, вершины которого соответствуют городам, а дуги – дорогам между городами. Вес C_{ij} дуги (i, j) есть стоимость перемещения из города i в город j.

К задаче легко применить общий метод ветвей и границ, получая алгоритм, который возвращается всякий раз, когда стоимость текущего частичного решения равняется или превосходит стоимость лучшего на данный момент решения. Эта проверка устраняет просмотр некоторых частей дерева поиска, но она достаточно слабая и допускает глубокое проникновение внутрь дерева до уровня, когда ветви обрываются. Произвольный фиксированный порядок городов является причиной того, что много времени теряется на исследование путей, которые начинаются с пары городов 1–2, если они отстоят далеко друг от друга.

Дерево поиска для 4-х городов



Для решения подобных задач методом ветвей и границ относительно успешной является перестройка дерева поиска (переупорядочение поиска), позволяющая находить почти оптимальные решения на ранних этапах поиска. Рассмотрим *алгоритм Литтла* (Little), в котором техника перестройки дерева заключается в разбиении на каждом шаге всех оставшихся решений на два подмножества с их точными нижними границами стоимостей. Для разбиения используется некоторая дуга (i, j), выбираемая по определенным критериям так, чтобы отсечь максимальное количество ветвей дерева. Первое подмножество (левое поддерево) включает все решения, в которые эта дуга входит (обозначим его $Y(i, j)$). Второе подмножество (правое поддерево) включает все решения, в которые дуга (i, j) не входит (обозначим его $N(i, j)$).

Таким образом, дерево поиска становится бинарным, каждой вершине сопоставляется нижняя граница стоимостей всех решений, вырастающих из него. Кроме того, каждая вершина (кроме корня) определяет локальное ограничение (входит данная дуга в решение или нет) на всех ее потомков. Поэтому для обозначения вершин будем использовать те же обозначения, что и для соответствующих множеств, т. е. $Y(i, j)$ и $N(i, j)$. Чтобы получить все ограничения, относящиеся к данной вершине, достаточно пройти путь от корня дерева к этой вершине.

По количеству элементов множества $Y(i, j)$ и $N(i, j)$ не одинаковы. Общее число решений в задаче с n городами равно $(n – 1)!$. Если зафиксирована дуга (i, j), т. е. выбраны два города, то имеется $(n – 2)$ способов выбрать третий город, $(n – 3)$ способов – четвертый и т. д. Следовательно, имеется $(n – 2)!$ решений, включающих дугу (i, j), а значит

$$(n – 1)! – (n – 2)! = (n – 2)(n – 2)!$$

решений, не включающих дугу (i, j). Поэтому при $n > 2$ множество $N(i, j)$ будет содержать большее число элементов, чем множество $Y(i, j)$. Поскольку множество с меньшим числом элементов исследовать проще, то в качестве дуги (i, j) следует брать такую, при которой множество $Y(i, j)$ имеет меньшую нижнюю границу, чем множество $N(i, j)$.

Для получения нижних границ L стоимостей решений используется тот факт, что в каждое решение входит только по одному элементу из каждой строки и из каждого столбца матрицы стоимостей. Поэтому уменьшение на какое-либо число элементов любой строки или любого столбца приводит к тому, что на это же число уменьшается стоимость всех решений, но сам путь остается неизменным. На этом свойстве основана операция *приведения* матрицы стоимостей.

Операция приведения матрицы заключается в вычитании из элементов каждой строки i наименьшего элемента этой строки (обозначим h_i), затем из элементов каждого столбца j вычитается наименьший элемент этого столбца (обозначим g_j). В результате приведения стоимости всех решений уменьшатся на сумму вычтенных из каждой строки и каждого столбца чисел, а приведенная матрица будет содержать неотрицательные элементы, причем в каждой строке и каждом столбце будет по крайней мере один элемент с нулевым значением. Таким образом, нижняя граница стоимости любого решения $L = \sum_{i=1}^{n} h_i + \sum_{j=1}^{n} g_j$. Пример приведения матрицы стоимостей для n = 5 представлен на рис. 3.3.

$i \setminus j$	1	2	3	4	5	h_i
1	∞	25	30	5	22	5
2	12	∞	8	16	19	8
3	17	24	∞	21	14	14
4	23	11	20	∞	9	9
5	27	13	7	26	∞	7

$\sum h_i = 43$

а

$i \setminus j$	1	2	3	4	5
1	∞	20	25	0	17
2	4	∞	0	8	11
3	3	10	∞	7	0
4	14	2	11	∞	0
5	20	6	0	19	∞
g_j	3	2	0	0	0

$\sum g_j = 5$

б

$i \setminus j$	1	2	3	4	5
1	∞	18	25	0	17
2	1	∞	0	8	11
3	0	8	∞	7	0
4	11	0	11	∞	0
5	17	4	0	19	∞

$L = \sum h_i + \sum g_j = 48$

в

Рис. 3.3. Приведение матрицы стоимостей:
а – исходная матрица стоимостей; *б* – матрица, приведенная по строкам;
в – приведенная матрица с нижней границей $L = 48$

Чтобы получить матрицу стоимостей для множества $N(i, j)$, следует в исходной (приведенной) матрице запретить движение по дуге (i, j), положив $C_{ij} = \infty$. Для множества $Y(i, j)$ матрица стоимостей получается из исходной удалением i-й строки и j-го столбца, так как включение в маршрут дуги (i, j) исключает возможность включения других дуг, идущих из i и приходящих в j. В результате размер матрицы уменьшается на единицу. Кроме того, необходимо следить за тем, чтобы предотвратить посещение города, который уже пройден, пока по самой последней дуге не будет реализован возврат в исходный город. В простейшем случае для этого достаточно запретить дугу (j, i), положив $C_{ji} = \infty$. В общем случае, если дуга (i_u, j_1) добавляется к частичному маршруту, состоящему из путей $i_1 - i_2 - ... - i_u$ и $j_1 - j_2 - ... - j_v$, то необходимо предотвратить использование дуги (j_v, i_1). Для построенных матриц вновь выполняется приведение, и на вычисленные значения увеличиваются нижние границы.

Осталось решить вопрос, какую дугу необходимо выбирать для разбиения. Если в качестве разбивающей дуги (i, j) взять дугу, у которой в приведенной матрице $C_{ij} \neq 0$, то для множества $Y(i, j)$ нижняя граница возрастет на C_{ij}, поскольку заранее известно, что эта дуга войдет во все решения. Она может еще возрасти, если матрица при удалении i-й строки и j-го столбца будет допускать дальнейшее приведение. В то же время при замене элемента C_{ij} на бесконечность матрица останется приведенной, т. е. нижняя граница для $N(i, j)$ не возрастет. Это нежелательно. Поэтому в качестве дуги (i, j) надо брать такую, у которой в приведенной матрице $C_{ij} = 0$. Поскольку таких дуг может быть несколько, надо выбрать ту, для которой увеличение нижней границы для множества $N(i, j)$ будет наибольшим, так как в этом случае получится наибольшая разница в нижних границах для множеств $Y(i, j)$ и $N(i, j)$. Другими словами, следует выбрать нуль, который при замене на бесконечность разрешает вычитать наибольшее число из его строки и столбца. Увеличение нижней границы множества $N(i, j)$ обозначим через $\Delta(i, j)$, а его значение получается при сложении наименьших чисел в i-й строке и j-м столбце (при $C_{ij} = \infty$).

Рассмотрим этот процесс для приведенной матрицы (рис. 3.3, *в*). Значения $\Delta(i, j)$ для нее следующие:

$$\Delta(1, 4) = 17 + 7 = 24;\ \Delta(2, 3) = 1;\ \Delta(3, 1) = 1;\ \Delta(3, 5) = 0;$$
$$\Delta(4, 2) = 4;\ \Delta(4, 5) = 0;\ \Delta(5, 3) = 4.$$

Наибольшее увеличение нижней границы правого поддерева дает выбор дуги (1, 4), т. е. для вершины $N(1, 4)$ нижняя граница $L = 48 + 24 = 72$, а соответствующая матрица стоимостей получается из исходной заменой значения $C_{1,4}$ на бесконечность. Для получения матрицы стоимостей для левого поддерева (вершины $Y(1, 4)$) необходимо удалить первую строку и четвертый столбец и наложить запрет на дугу (4, 1), положив $C_{4,1} = \infty$ (рис. 3.4, *а*). Поскольку полученная матрица является приведенной (каждая строка и каждый столбец содержат по крайней мере один элемент с нулевым значением), нижняя граница для $Y(1, 4)$ не изменяется и остается равной 48.

Продолжим рассмотрение левого поддерева. Имеем следующие значения $\Delta(i, j)$:

$$\Delta(2, 3) = 1;\ \Delta(3, 1) = 1;\ \Delta(3, 5) = 0;$$
$$\Delta(4, 2) = 4;\ \Delta(4, 5) = 0;\ \Delta(5, 3) = 4.$$

Наибольшее увеличение нижней границы правого поддерева дают дуги (4, 2) и (5, 3). Выбираем любую из них, например, (4, 2), получая цепочку 1 – 4 – 2. Тогда для $N(4, 2)$ имеем нижнюю границу $L = 48 + 4 = 52$. Матрица для $Y(4, 2)$ получается удалением четвертой строки и второго столбца. Для предотвращения посещения города 1 из города 2 (получается замкнутый контур, но еще не все города пройдены) необходимо положить $C_{2,1} = \infty$ (рис. 3.4, *б*). Полученная матрица является приведенной, поэтому нижняя граница для $Y(4, 2)$ не меняется и остается равной 48.

$i \setminus j$	1	2	3	5
2	1	∞	0	11
3	0	8	∞	0
4	∞	0	11	0
5	17	4	0	∞

$L = 48$

а

$i \setminus j$	1	3	5
2	∞	0	11
3	0	∞	0
5	17	0	∞

$L = 48$

б

$i \setminus j$	1	5
2	∞	11
3	0	∞

$L = 48$

в

$i \setminus j$	1	5
2	∞	0
3	0	∞

$L = 48 + 11 = 59$

г

Рис. 3.4 Процесс получения маршрута 1 – 4 – 2 – 5 – 3 – 1:
а – после выбора дуги (1, 4); *б* – после выбора дуги (4, 2);
в – после выбора дуги (5, 3); *г* – приведение матрицы *в*

Для полученной матрицы имеем:

$$\Delta(2, 3) = 11; \Delta(3, 1) = 17; \Delta(3, 5) = 11; \Delta(5, 3) = 17.$$

Наибольшее увеличение нижней границы правого поддерева дают дуги (3, 1) и (5, 3). Чтобы показать, что частичное решение может содержать произвольное число непересекающихся цепочек, выберем для разбиения дугу (5, 3), получая цепочки 1 – 4 – 2 и 5 – 3. Тогда для $N(5, 3)$ имеем $L = 48 + 17 = 65$. Матрица стоимостей для $Y(5, 3)$ получается удалением пятой строки и третьего столбца и заменой $C_{3,5}$ на бесконечность (рис. 3.4, *в*). Полученная матрица не является приведенной. Для ее приведения надо вычесть 11 из второй строки (рис. 3.4, *г*), что увеличивает нижнюю границу стоимостей решений в левом поддереве, т. е. для $Y(5, 3)$ нижняя граница $L = 48 + 11 = 59$. Дальнейшее разбиение решений не имеет смысла, поскольку правые поддеревья будут пустыми (при наличии цепочек 1 – 4 – 2 и 5 – 3 не существует решений, в которые не входят дуги (2, 5) или (3, 1)), т. е. остается единственная возможность – добавить в решение обе оставшиеся дуги (2, 5) и (3, 1). В результате получается замкнутый путь (маршрут) 1 – 4 – 2 – 5 – 3 – 1, стоимость которого равна 59.

Чтобы выяснить, является ли полученное решение оптимальным, необходимо просмотреть значения нижних границ для всех правых поддеревьев. Если значение нижней границы какого-либо правого поддерева меньше стоимости полученного решения, то в этом поддереве может находиться лучшее решение, т. е. необходимо продолжить решение задачи, применяя метод разбиений решений для этого поддерева.

В данном примере имеются следующие значения нижних границ правых поддеревьев: $L = 65 > 59$ для $N(5, 3)$, $L = 52 < 59$ для $N(4, 2)$ и $L = 72 > 59$ для $N(1, 4)$. Поскольку оценка $L = 52$ для вершины $N(4, 2)$ меньше стоимости 59 лучшего на данный момент маршрута, более оптимальное решение возможно в поддереве с корнем $N(4, 2)$, т. е. в подмножестве решений, маршруты которых включают в себя дугу (1, 4) и не содержат дугу (4, 2). Необходимо повторить процесс разбиения решений. Матрица стоимостей для этого поддерева (рис. 3.5, *а*) отличается от матрицы на рис. 3.4, *а* тем, что значение $C_{4,2} = \infty$ для запрета дуги (4, 2). Приведение этой матрицы вычитанием 4 из второго столбца (рис. 3.5, *б*) и дает значение нижней границы рассматриваемого множества, равное 48 + 4 = 52. Последующее применение метода разбиений дает такой порядок выбора дуг: (4, 5), (3, 1), (2, 3) и (5, 2). Соответствующие приведенные матрицы для левых поддеревьев представлены на рис. 3.5, *в* для дуги (4, 5) и рис. 3.5, *г* для дуги (3, 1).

В результате получается замкнутый путь 1 – 4 – 5 – 2 – 3 – 1, стоимость которого равна 52. Поскольку нет правых поддеревьев, значения нижних границ которых меньше 52, второе найденное решение является оптимальным.

$i \setminus j$	1	2	3	5
2	1	∞	0	11
3	0	8	∞	0
4	∞	∞	11	0
5	17	4	0	∞

$L = 48$

а

$i \setminus j$	1	2	3	5
2	1	∞	0	11
3	0	4	∞	0
4	∞	∞	11	0
5	17	0	0	∞

$L = 48 + 4 = 52$

б

$i \setminus j$	1	2	3
2	1	∞	0
3	0	4	∞
5	∞	0	0

$L = 52$

в

$i \setminus j$	2	3
2	∞	0
5	0	∞

$L = 52$

г

Рис. 3.5.Процесс получения маршрута 1 – 4 – 5 – 2 – 3 – 1
а – запрещена дуга (4, 2); *б* – приведение матрицы *а*;
в – после выбора дуги (4, 5); *г* – после выбора дуги (3, 1)

Бинарное дерево поиска для рассмотренного примера решения задачи коммивояжера представлено на рис. 3.6.

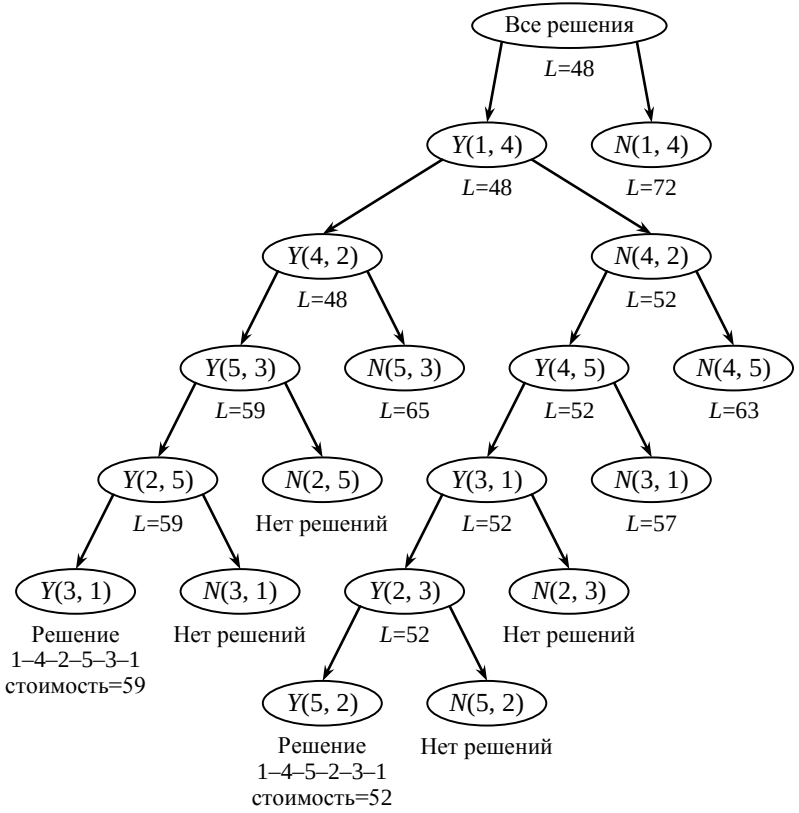


Рис. 3.6. Бинарное дерево поиска для задачи коммивояжера

В данном примере исследовано 19 вершин. Если использовать очевидную специализацию общего метода ветвей и границ, то потребуется исследовать гораздо больше вершин. На основе экспериментальных данных предполагается, что для случайной $n \times n$ матрицы стоимостей число исследуемых вершин при применении изложенного метода переупорядочения поиска равно $O(1{,}26^n)$ [21].