

2 УНИВЕРСАЛЬНЫЕ АЛГОРИТМЫ ДЛЯ НЕОГРАНИЧЕННОЙ ЗАДАЧИ РАЗМЕЩЕНИЯ

Задача размещения средств обслуживания, как следует из результатов предыдущей главы, является труднорешаемой задачей. Это означает, что, скорее всего, она не может быть решена за полиномиальное время и, следовательно, при построении универсальных алгоритма ее решения необходимо исходить из того, что это будет либо точный переборный алгоритм, либо эффективный приближенный алгоритм с априорной оценкой точности, либо быстрый эвристический алгоритм с апостериорной оценкой точности получаемого приближенного решения.

Одним из наиболее известных эвристических приемов построения приближенного решения как для неограниченной задачи размещения, так и для многих других задач дискретной оптимизации является процедура, называемая в современной литературе «жадным» или «гриди» алгоритмом [49]. Суть такой процедуры, которую удобнее всего применять к задаче оптимизации функций, заданных на подмножествах некоторого конечного множества, состоит в последовательном увеличении текущего решения посредством добавления к нему на каждом шаге некоторого элемента или в последовательном уменьшении текущего решения в результате удаления из него на каждом шаге некоторого элемента. В любом случае изменения текущего решения производятся до тех пор, пока значение целевой функции перестает уменьшаться. Гриди такую процедуру называют потому, что на каждом шаге изменение текущего решения производится с использованием элемента, который дает наибольшее уменьшение целевой функции. Наиболее ранней публикацией, в которой гриди процедура используется для решения задачи размещения средств обслуживания, является, вероятно, работа [226].

Другим универсальным подходом к решению задачи размещения, с учетом того, что множество допустимых решений этой задачи конечно, является процедура перебора. Такой подход может оказаться плодотворным, если различного рода улучшениями привести его к виду, когда в каждом конкретном случае процедуры полного перебора удастся избежать и заменить ее частичным перебором. Улучшенные переборные схемы получили название методов «неявного перебора» [57]. Такой схемой является, например, метод последовательных расчетов [73, 77] или аппроксимационно–комбинаторный метод [76, 77]. Однако наибольшее распространение среди методов неявного перебора получил метод ветвей и границ [17, 57, 198]. Именно этот метод в различных модификациях наиболее успешно применялся для численного решения различных дискретных экстремальных задач. Успешным оказался опыт использования метода ветвей и границ для решения задачи размещения средств обслуживания.

При разработке универсальных алгоритмов решения задачи размещения, как эвристических так и переборных алгоритмов типа ветвей и границ важное значение имеет построение «хорошей» нижней границы для оптимального значения целевой функции задачи. В случае эвристических алгоритмов такие нижние границы необходимы для оценки точности найденного приближенного решения, а в алгоритмах ветвей и границ нижние оценки вообще играют ключевую роль, поскольку от того, насколько быстро и

точно вычисляется нижняя граница, зависит практическая работоспособность алгоритма.

§1 настоящей главы посвящен исследованию вопроса о нижней границе для целевой функции задачи размещения. Применительно к этой задаче рассматриваются два способа построения нижней границы: линейная релаксация и релаксация по Лагранжу. Показано, что оба эти приема приводят к одинаковым результатам и поэтому как основной способ построения нижних оценок используется процедура вычисления «хороших» допустимых решений задачи, двойственной к релаксированной задаче размещения. В качестве таких решений используются так называемые тупиковые решения.

В §2 рассматриваются гриды алгоритмы решения задачи размещения. Предлагаются три такие процедуры. Первая из них — простейшая гриды процедура последовательного увеличения текущего решения без учета каких-либо специфических свойств задачи размещения средств обслуживания. Эти свойства принимаются во внимание только при построении апостериорной оценки точности получаемого приближенного решения. Второй гриды алгоритм использует свойства тупикового решения и строит приближенное решение посредством последовательного уменьшения числа элементов текущего решения, начиная с блокирующего множества. При этом рассматриваемое на начальных шагах текущее решение является так называемым критическим подмножеством, для которого удастся вычислить оценку точности, убывающую с уменьшением числа элементов в критическом множестве. Третий алгоритм является модификацией первого в плане учета свойства элементов блокирующего множества, как перспективных элементов на вхождение в хорошее приближенное решение. Этот алгоритм отличается от первого тем, что на каждом шаге выбор элемента, расширяющего текущее решение, производится не из множества всех возможных элементов, а из некоторого специально сформированного множества. Это множество строится случайным образом, поэтому сам алгоритм является вероятностным. При этом преимущественные шансы попасть в это множество имеют элементы блокирующего множества или «близкие» к ним.

В §3 дается подробное описание метода ветвей и границ и алгоритмов решения задачи размещения, построенных на основе этого метода. Предлагается базовый алгоритм ветвей и границ, вычисляющий оптимальное решение задачи размещения, приближенный алгоритм с априорной оценкой точности, а также приближенный алгоритм, представляющий из себя комбинацию процедуры ветвей и границ с гриды процедурой. Для иллюстрации качества представленных алгоритмов и возможностей их практического использования рассматриваются результаты тестовых расчетов с использованием данных алгоритмов.

1 Нижние оценки для целевой функции неограниченной задачи размещения

Универсальным приемом вычисления нижней границы для целевой функции задачи FL , как и для всякой другой линейной целочисленной задачи, является переход к

рассмотрению линейной релаксации исследуемой задачи, то есть переход к задаче, в которой условия целочисленности переменных заменены на более слабые условия, например, неотрицательности. Оптимальное значение целевой функции релаксированной задачи, которая по построению имеет более широкое множество допустимых решений, дает искомую нижнюю границу. Однако отыскание оптимального решения для релаксированной задачи также может оказаться достаточно трудоемкой процедурой, особенно, если для этого приходится прибегать к универсальным алгоритмам линейного программирования. В этом случае можно перейти к рассмотрению задачи, двойственной к релаксированной. Согласно теории двойственности, всякое допустимое решение двойственной задачи будет давать искомую оценку снизу. Поскольку оценку снизу порождает любое допустимое решение, то можно ограничиться поиском не оптимального решения, а некоторого «хорошего» решения, дающего значение целевой функции, близкое к оптимальному. Отыскание такого решения является уже более простой задачей и открывает возможности для построения малотрудоемкого алгоритма вычисления нижней оценки.

Исторически можно выделить два существенных этапа построения нижних оценок для задачи FL с использованием линейной релаксации. Отличие этих этапов состоит в том, что на первом для релаксации использовалась задача FL в слабой форме, а на втором — в сильной форме. И хотя оба подхода различаются по, казалось бы, незначительному фактору, однако по достигнутым результатам эти подходы существенно различаются. Дело в том, что хотя оптимальные решения задач FL в слабой и сильной формах совпадают, их релаксированные варианты не являются эквивалентными задачами. Первая из релаксированных задач решается достаточно просто, поэтому не удивительно, что получаемая с ее помощью нижняя оценка является грубой. В качестве наиболее известных работ, относящихся к первому этапу, отметим уже упоминавшиеся выше работы [212, 244, 245, 222].

Отличительная особенность второго подхода к построению нижних оценок состоит в переходе к рассмотрению задачи, двойственной к релаксированной задаче FL в сильной форме. Основные идеи этого подхода изложены в работах [213, 13, 17, 51, 159], которые по существу определяют и современное состояние вопроса о нетрудоемком вычислении нижних оценок для задачи FL . В указанных работах, принадлежащих разным авторам, независимо и практически одновременно предложена нетрудоемкая процедура построения «хорошего» допустимого решения двойственной задачи. Такие решения названы в [13, 17] тупиковыми, а сам алгоритм, благодаря [213], получил название «процедура подъема».

Следует отметить, что к вычислению тупикового решения сводится и попытка построения нижней оценки для задачи FL с помощью другого универсального приема — использования двойственной по Лагранжу задачи [217].

Таким образом, процедура подъема далее рассматривается как основной способ вычисления нижних оценок для различных универсальных алгоритмов решения задачи FL . Особенно простой вид эта процедура приобретает, если применять ее к задаче FL , когда матрица затрат на обслуживание C задана в канонической форме.

1.1 Двойственные задачи для релаксированной задачи размещения

Задача FL имеет две формулировки в виде задачи целочисленного линейного программирования, называемые задачей FL в сильной форме и в слабой форме. Рассмотрим релаксированные варианты этих задач, получаемые отбрасыванием из условий задач требования целочисленности переменных.

Релаксированная задача FL в сильной форме имеет вид:

$$\min_{(z_i), (x_{ij})} \left\{ \sum_{i \in I} f_i z_i + \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij} \right\}; \quad (2.1.1)$$

$$\sum_{i \in I} x_{ij} = 1, \quad j \in J; \quad (2.1.2)$$

$$z_i \geq x_{ij}, \quad i \in I, j \in J; \quad (2.1.3)$$

$$z_i, x_{ij} \geq 0, \quad i \in I, j \in J, \quad (2.1.4)$$

а релаксированная задача в слабой форме записывается следующим образом:

$$\min_{(z_i), (x_{ij})} \left\{ \sum_{i \in I} f_i z_i + \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij} \right\}; \quad (2.1.5)$$

$$\sum_{i \in I} x_{ij} = 1, \quad j \in J; \quad (2.1.6)$$

$$\sum_{j \in J} x_{ij} \leq n z_i, \quad i \in I; \quad (2.1.7)$$

$$z_i, x_{ij} \geq 0, \quad i \in I, j \in J. \quad (2.1.8)$$

Пусть $((z_i), (x_{ij}))$ – оптимальное решение последней задачи. Несложно увидеть, что для данного решения выполняются равенства

$$\sum_{j \in J} x_{ij} = n z_i, \quad i \in I;$$

Отсюда получаем, что если $x_{ij} > 0$ для некоторых $i \in I, j \in J$, то

$$\frac{1}{n} f_i + c_{ij} = \min_{k \in I} \left\{ \frac{1}{n} f_k + c_{kj} \right\}$$

и, следовательно, оптимальное значение целевой функции (2.1.5) равняется

$$\min_{(z_i), (x_{ij})} \left\{ \sum_{i \in I} f_i z_i + \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij} \right\};$$

Таким образом, с использованием релаксированной задачи FL в слабой форме получаем достаточно простой способ вычисления нижней границы для значений целевой функции задачи FL . Однако такая нижняя граница является одновременно и достаточно грубой, что видно из сравнения ее с оптимальным значением целевой функции (2.1.1). Дело в том, что, хотя задача FL в сильной форме и задача FL в слабой форме эквивалентны, их релаксации такими не являются. Действительно, если $((z_i), (x_{ij}))$ – до-

пустимое решение задачи (2.1.1)–(2.1.4), то это решение будет допустимым решением и для задачи (2.1.5)–(2.1.8). Следовательно, оптимальное значение целевой функции (2.1.1) не меньше оптимального значения целевой функции (2.1.5). С другой стороны, для оптимального решения $((z_i), (x_{ij}))$ задачи (2.1.5)–(2.1.8) условие (2.1.3), принимающее вид

$$z_i = \frac{1}{n} \sum_{l \in J} x_{il} \geq x_{ij}, \quad i \in I, j \in J,$$

не выполняются за исключением тривиального случая, когда значения всех переменных x_{ij} одинаковые. Следовательно, в нетривиальных случаях оптимальное значение целевой функции (2.1.1) строго больше оптимального значения целевой функции (2.1.5). Это означает, что релаксированная задача FL в сильной форме приводит к более тонким нижним оценкам, чем релаксированная задача FL в слабой форме.

В силу сказанного и нашей заинтересованности в получении достаточно точных нижних оценок, сосредоточим свое внимание на релаксированной задаче FL в сильной форме (2.1.1)–(2.1.4). Поскольку для этой задачи не известно простых алгоритмов построения оптимального решения, то перейдем к рассмотрению задачи, двойственной к исследуемой. Значение целевой функции такой задачи, согласно фундаментальным свойствам пары двойственных задач, также будет давать оценку снизу для значений целевой функции задачи FL .

Задача двойственная к задаче (2.1.1)–(2.1.4), записывается следующим образом:

$$\begin{aligned} \max_{(u_j), (v_{ij})} \quad & \sum_{j \in J} u_j; \\ u_j - v_{ij} \leq & c_{ij}, \quad i \in I, j \in J; \\ \sum_{j \in J} v_{ij} \leq & f_i, \quad i \in I; \\ v_{ij} \geq & 0, \quad i \in I, j \in J. \end{aligned}$$

Любую из двух групп переменных, используемых для записи этой задачи, можно исключить и тем самым переформулировать задачу в более компактном и удобном для исследования виде.

Пусть $((u_j), (v_{ij}))$ – допустимое решение рассматриваемой задачи. Если $u_j \leq c_{ij}$ для некоторых $i \in I, j \in J$, то, положив $v_{ij} = 0$, получим новое допустимое решение с тем же значением целевой функции, что и для исходного решения. Если же $u_j > c_{ij}$, то, положив $v_{ij} = u_j - c_{ij}$, так же получим допустимое решение, не меняя при этом значения целевой функции. Поэтому можно утверждать, что существует оптимальное решение $((u_j), (v_{ij}))$ рассматриваемой задачи такое, что

$$v_{ij} = (u_j - c_{ij})^+ = \begin{cases} u_j - c_{ij}, & \text{если } u_j > c_{ij}, \\ 0, & \text{иначе.} \end{cases}$$

Отсюда следует, что рассматриваемая двойственная задача эквивалентна следующей задаче:

$$\max_{(u_j)} \sum_{j \in J} u_j; \quad (2.1.9)$$

$$\sum_{j \in J} (u_j - c_{ij})^+ \leq f_i, \quad i \in I. \quad (2.1.10)$$

Те же самые соображения приводят к важному замечанию о том, что существует оптимальное решение $((u_j), (v_{ij}))$ двойственной задачи такое, что

$$u_j = \min_{i \in I} \{c_{ij} + v_{ij}\}, \quad j \in J.$$

Это замечание позволяет исключить из рассмотрения переменные $u_j, j \in J$ двойственной задачи и переписать ее следующим образом:

$$\max_{(v_{ij})} \sum_{j \in J} \min_{i \in I} \{c_{ij} + v_{ij}\}; \quad (2.1.11)$$

$$\sum_{j \in J} v_{ij} \leq f_i, \quad i \in I; \quad (2.1.12)$$

$$v_{ij} \geq 0, \quad i \in I, j \in J. \quad (2.1.13)$$

Эту задачу далее будем обозначать через *DFL*. Решением данной задачи является матрица $V=(v_{ij})$ размера $m \times n$, удовлетворяющая условию (2.1.12) и (2.1.13).

Если $V=(v_{ij})$ – решение задачи *DFL*, то матрица $W=(w_{ij})$ размера $m \times n$, где $w_{ij} = c_{ij} + v_{ij}, i \in I, j \in J$, обладает тем свойством, что величина

$$H(W) = \sum_{j \in J} \min_{i \in I} w_{ij}$$

является нижней границей для значений целевой функции задачи *FL*. Матрицу W будем называть *оценочной матрицей*, порожденной матрицей V . Матрица W , как легко видеть, получается из матрицы C в результате увеличения элементов этой матрицы. Это увеличение достигается посредством распределения «ресурсов» $f_i, i \in I$, диагональной матрицы F_0 по столбцам матрицы C . При этом элементы матрицы V определяют, какая доля каждого ресурса матрицы F_0 направляется в тот или иной столбец матрицы C в качестве добавки к соответствующему элементу этого столбца.

Таким образом, для вычисления нижней оценки целевой функции задачи *FL* мы намерены использовать оценочную матрицу, порождаемую решением задачи *DFL*, являющейся двойственной в смысле линейного программирования для релаксированной задачи *FL* в сильной форме. Однако прежде попытаемся сравнить данный подход к построению нижней оценки с возможностями другого универсального приема построения нижних оценок, базирующегося на рассмотрении задач, двойственных по Лагранжу.

1.2 Двойственные по Лагранжу задачи для задачи размещения

Согласно [217] двойственная по Лагранжу задача для исходной задачи целочисленного программирования на минимум строится на основе так называемой *Лагранжевой релаксации* исходной задачи относительно некоторой группы ее ограничений. В результате такой релаксации эти ограничения с некоторыми коэффициентами, называе-

мыми *множителями Лагранжа*, переносятся в целевую функцию исследуемой задачи, образуя новую целевую функцию называемую *функцией Лагранжа*. Таким образом, получаем новую задачу с функцией Лагранжа в качестве целевой, коэффициентами Лагранжа в качестве фиксированных параметров целевой функции и ограничениями исходной задачи, не перенесенными в целевую функцию. *Двойственной по Лагранжу* называется задача максимизации по множителям Лагранжа оптимального значения целевой функции этой новой задачи. Фундаментальным свойством двойственности по Лагранжу является то, что при любых значениях множителей Лагранжа оптимальное значение функции Лагранжа не превосходит оптимального значения целевой функции исходной задачи. Это свойство, естественно, может быть использовано для вычисления нижних оценок для целевой функции исследуемой оптимизационной задачи.

Применительно к задаче FL могут быть рассмотрены две Лагранжевы релаксации и построены две функции Лагранжа. Первая получается в результате переноса в целевую функцию ограничений (2.1.2) с множителями $u_j, j \in J$, а вторая — в результате переноса ограничений (2.1.3) с множителями $v_{ij} \geq 0, i \in I, j \in J$. В первом случае приходим к функции Лагранжа вида

$$\sum_{i \in I} f_i z_i + \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij} + \sum_{j \in J} u_j (1 - \sum_{i \in I} x_{ij})$$

и соответствующей Лагранжевой релаксацией задачи FL

$$\begin{aligned} \min_{(z_i), (x_{ij})} \{ & \sum_{i \in I} f_i z_i + \sum_{j \in J} u_j + \sum_{i \in I} \sum_{j \in J} (c_{ij} - u_j) x_{ij} \}; \\ & z_i \geq x_{ij}, \quad i \in I, j \in J; \\ & z_i, x_{ij} \in \{0, 1\}, \quad i \in I, j \in J. \end{aligned}$$

Поскольку существует оптимальное решение $((z_i), (x_{ij}))$ этой задачи такое, что

$$x_{ij} = \begin{cases} z_i, & \text{если } c_{ij} - u_j \leq 0, \\ 0, & \text{иначе,} \end{cases}$$

то данную задачу можно эквивалентным образом переписать в виде

$$\begin{aligned} \min_{(z_i)} \{ & \sum_{i \in I} (f_i - \sum_{j \in J} (u_j - c_{ij})^+) z_i + \sum_{j \in J} u_j \}; \\ & z_i \in \{0, 1\}, \quad i \in I. \end{aligned}$$

В свою очередь оптимальное решение (z_i) последней задачи определяется очевидным образом. Поэтому окончательно получаем следующее представление оптимального значения функции Лагранжа с множителями $u_j, j \in J$, в качестве параметров

$$- \sum_{i \in I} \left(\sum_{j \in J} (u_j - c_{ij})^+ - f_i \right)^+ + \sum_{j \in J} u_j.$$

Соответствующая двойственная по Лагранжу задача записывается следующим образом

$$\max_{(u_j)} \left\{ - \sum_{i \in I} \left(\sum_{j \in J} (u_j - c_{ij})^+ - f_i \right)^+ + \sum_{j \in J} u_j \right\}.$$

Эта задача, как несложно увидеть, и задача (2.1.9) – (2.1.10) эквивалентны.

К аналогичному результату приходим и во втором случае, когда рассматриваем функцию Лагранжа вида

$$\sum_{i \in I} f_i z_i + \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij} + \sum_{i \in I} \sum_{j \in J} v_{ij} (x_{ij} - z_i)$$

и соответствующую Лагранжеву релаксацию задачи FL

$$\begin{aligned} \min_{(z_i), (x_{ij})} \{ & \sum_{i \in I} (f_i - \sum_{j \in J} v_{ij}) z_i + \sum_{i \in I} \sum_{j \in J} (c_{ij} + v_{ij}) x_{ij} \}; \\ & \sum_{i \in I} x_{ij} = 1, \quad j \in J; \\ & z_i, x_{ij} \in \{0, 1\}, \quad i \in I, j \in J. \end{aligned}$$

Поскольку существует оптимальное решение $((z_i), (x_{ij}))$ этой задачи такое, что

$$\begin{aligned} z_i &= \begin{cases} 1, & \text{если } f_i - \sum_{j \in J} v_{ij} < 0, \\ 0, & \text{иначе;} \end{cases} \\ x_{ij} &= \begin{cases} 1, & \text{если } i = \arg \min_{k \in I} \{c_{kj} + v_{kj}\}, \\ 0, & \text{иначе,} \end{cases} \end{aligned}$$

то двойственная по Лагранжу задача записывается следующим образом:

$$\begin{aligned} \max_{(v_{ij})} \{ & - \sum_{i \in I} (\sum_{j \in J} v_{ij} - f_i)^+ + \sum_{j \in J} \min_{i \in I} \{c_{ij} + v_{ij}\} \}; \\ & v_{ij} \geq 0, \quad i \in I, j \in J. \end{aligned}$$

Несложно видеть, что эта задача и задача DFL эквивалентны, а оптимальные значения целевых функций этих задач равны.

Таким образом, в случае задачи FL двойственные по Лагранжу задачи не дают никаких преимуществ для вычисления нижней границы по сравнению с двойственными задачами для релаксированной задачи FL . Поэтому основным способом вычисления нижней оценки остается построение решения $V=(v_{ij})$ задачи DFL .

1.3 Тупиковые решения

Из сказанного выше получаем, что наилучшую нижнюю оценку $H(W^*)$ дает оценочная матрица W^* , порожденная оптимальным решением V^* задачи DFL . Однако при построении алгоритмов вычисления нижней оценки мы откажемся от непременно-го поиска оптимального решения V^* задачи DFL с использованием, например, универсальных алгоритмов линейного программирования. Это связано с достаточной трудоемкостью такой процедуры, что оказывается неоправданным при необходимости многократного вычисления нижних границ. Поэтому мы вынуждены частично поступиться точностью вычисляемой нижней границы и ограничиться построением решений V , приводящих к «хорошим» оценочным матрицам W , дающим значение величины нижней оценки $H(W)$ близкое к максимальной величине $H(W^*)$. В качестве таких решений V рассмотрим так называемые тупиковые матрицы V .

Для решения $V=(v_{ij})$ задачи DFL , порождающего оценочную матрицу $W=(w_{ij})$ положим

$$u_j = \min_{i \in I} w_{ij}, \quad j \in J;$$

$$d_i = f_i - \sum_{j \in J} v_{ij}, i \in I$$

и рассмотрим множества

$$I_0(V) = \{i \in I \mid d_i = 0\};$$

$$I_j(V) = \{i \in I \mid w_{ij} = u_j\}, \quad j \in J;$$

$$I_j^+(V) = \{i \in I \mid v_{ij} > 0\}, \quad j \in J;$$

$$J_0(V) = \{j \in J \mid I_0(V) \cap I_j(V) \neq \emptyset\}.$$

Используя эти множества можно сформулировать необходимые и достаточные условия неуллучшаемости решения V , то есть условия существования покомпонентно большего решения V' , для которого $H(W') > H(W)$. Несложно понять, что такое решение V' существует тогда и только тогда, когда $J_0(V) \neq J$. Действительно, если $J_0(V) = J$, то для всякого $j_0 \in J$ найдется $i_0 \in I$, для которого одновременно $d_{i_0} = 0$ и $u_{j_0} = w_{i_0 j_0}$. Поэтому невозможно увеличить величину $v_{i_0 j_0}$ и, следовательно, величину u_{j_0} , не уменьшая при этом величины $v_{i_0 j}$, $j \neq j_0$. С другой стороны, если $J_0(V) \neq J$, $j_0 \notin J_0(V)$, то для всякого $i \in I_{j_0}(V)$ имеем $d_i > 0$. Поэтому увеличение величин v_{ij_0} , $i \in I_{j_0}(V)$, на некоторую величину Δ , $0 < \Delta \leq \min_{i \in I_{j_0}(V)} d_i$, возможно и это приводит к соответствующему увеличению величины u_{j_0} и, следовательно, величины $H(W)$.

С учетом сказанного, множество $I_0(V)$ будем называть *блокирующим* множеством строк матрицы V , а $J_0(V)$ – множеством *заблокированных* столбцов матрицы V . Подмножество I' блокирующего множества $I_0(V)$ назовем *критическим*, если $I' \cap I_j(V) \neq \emptyset$ для всякого $j \in J$. Отсюда получаем, что решение V – неуллучшаемое тогда и только тогда, когда $I_0(V)$ – критическое множество.

Решение $V=(v_{ij})$ задачи DFL назовем *тупиковым*, если выполняются следующие два условия:

$$1. \quad I_j(V) \cap I_0 \neq \emptyset \quad \text{для всякого } j \in J;$$

$$2. \quad \text{Для любых } i \in I, j \in J, \text{ если } v_{ij} > 0, \text{ то } u_j = w_{ij}.$$

Первое условие означает, что все столбцы матрицы V являются заблокированными и, следовательно, тупиковое решение V – неуллучшаемое. Второе условие реализует идею экономного расходования «ресурсов» f_i , $i \in I$, диагональной матрицы F_0 при построении оценочной матрицы W . Это означает, что всякий элемент c_{ij} матрицы C при построении матрицы W получает «добавку» $v_{ij} > 0$ только в том случае, если это приводит к увеличению величины u_j .

Отметим, что если решение V – тупиковое, то для всякого $j \in J$ имеем $I_j(V) \supset I_j^+(V)$ и, более того, $I_j(V) \setminus I_j^+(V) = \{i \in I \mid c_{ij} = u_j\}$.

Отметим также, что если V – оптимальное решение, то оно является неувлучшаемым и, следовательно, удовлетворяет условию 1. Понятно также, что всякое оптимальное решение V уменьшением некоторых компонент $v_{ij} > 0$ таких, что $w_{ij} > u_j$, легко может быть преобразовано в оптимальное решение V' , для которого выполняется условие 2. Так что ограничиваясь только тупиковыми решениями задачи DFL , мы не теряем ни одного существенного оптимального решения и сосредотачиваемся на рассмотрении только «хороших» допустимых решений, обеспечивающих экономное распределение ресурсов матрицы F_0 по столбцам матрицы C .

Косвенным свидетельством того, что тупиковые решения не приводят к построению плохих оценочных матриц, являются результаты их сравнения с другим классом допустимых решений задачи DFL с так называемыми матрицами, порожденными векторами.

Пусть (α_j) – вектор длины n с компонентами $\alpha_j \geq 0$, $j \in J$ такими, что $\sum_{j \in J} \alpha_j \leq 1$.

Решение (v_{ij}) задачи DFL назовем *порожденным* вектором (α_j) , если $v_{ij} = f_i \alpha_j$, $i \in I, j \in J$. Такое решение действительно является решением задачи DFL , поскольку условие (2.1.12), очевидно, выполняется. Если $\alpha_j = \frac{1}{n}$, $j \in J$, то получаем частный случай рассматриваемого решения, при котором значение целевой функции равняется оптимальному значению целевой функции релаксированной задачи FL в слабой форме. В [17] доказана нижеследующая теорема, показывающая, что всякое, даже самое «плохое» тупиковое решение, не хуже любого решения, порожденного вектором.

Теорема 2.1. Для всякого тупикового решения $V = (v_{ij})$ задачи DFL и для всякого решения задачи DFL , порожденного вектором (α_j) , имеет место неравенство

$$\sum_{j \in J} \min_{i \in I} \{c_{ij} + v_{ij}\} \geq \sum_{j \in J} \min_{i \in I} \{c_{ij} + \alpha_j f_i\}.$$

Доказательство теоремы проведем индукцией по числу элементов множества I . При этом, если $I' \subset I$, то через $V(I')$ будем обозначать решение задачи DFL , в котором множество I заменено на множество I' .

Пусть $I' = \{i_0\}$. Тогда, очевидно, можем написать

$$\sum_{j \in J} (c_{i_0 j} + v_{i_0 j}) = f_{i_0} + \sum_{j \in J} c_{ij} \geq \sum_{j \in J} (c_{i_0 j} + \alpha_j f_{i_0}).$$

Пусть I' – произвольное подмножество множества I и пусть для каждого $i_0 \in I'$ требуемое неравенство выполняется для любого тупикового решения $V(I' \setminus \{i_0\})$. Рассмотрим тупиковое решение $V(I')$ и выберем элемент $i \in I'$ такой, что

$$i_0 = \arg \max_{i \in I'} f_i.$$

Если $i_0 \notin I_0(V(I'))$, то решение $V(I' \setminus \{i_0\})$ – также тупиковое и, с учетом предположения индукции, получаем

$$\sum_{j \in J} \min_{i \in I'} \{c_{ij} + v_{ij}\} = \sum_{j \in J} \min_{i \in I' \setminus \{i_0\}} \{c_{ij} + v_{ij}\} \geq \sum_{j \in J} \min_{i \in I' \setminus \{i_0\}} \{c_{ij} + \alpha_j f_i\} \geq \sum_{j \in J} \min_{i \in I'} \{c_{ij} + \alpha_j f_i\}.$$

Пусть теперь $i_0 \in I_0(V(I'))$ и пусть $J = \{j \in J \mid i_0 \in I_j(V(I'))\}$. Тогда можем написать

$$\begin{aligned} \sum_{j \in J} \min_{i \in I'} \{c_{ij} + v_{ij}\} &= \sum_{j \in J_0} (c_{i_0 j} + v_{i_0 j}) + \sum_{j \notin J_0} \min_{i \in I'} \{c_{ij} + v_{ij}\} \geq f_{i_0} + \sum_{j \in J_0} c_{i_0 j} + \sum_{j \notin J_0} \min_{i \in I'} c_{ij} \geq \\ &\geq \sum_{j \in J} \alpha_j f_{i_0} + \sum_{j \in J} \min_{i \in I'} c_{ij} = \sum_{j \in J} \min_{i \in I'} \{c_{ij} + \alpha_j f_{i_0}\} \geq \sum_{j \in J} \min_{i \in I'} \{c_{ij} + \alpha_j f_i\}. \end{aligned}$$

Таким образом, в обоих случаях получаем требуемое неравенство, что завершает доказательство теоремы.

В заключение, сделаем еще одно замечание о свойствах тупикового решения V . Пусть $V^* = (v_{ij}^*)$ – оптимальное решение задачи DFL . В силу теоремы двойственности [61], для любого оптимального решения $((z_i^*), (x_{ij}^*))$ релаксированной задачи FL выполняются равенства

$$z_i^* (f_i - \sum_{j=1}^n v_{ij}^*) = 0, \quad i \in I,$$

называемые условиями дополняющей нежесткости. Из этих условий вытекает, что если $i \notin I_0(V^*)$, то $z_i^* = 0$.

Таким образом, если предположить, что между нулевыми компонентами оптимальных решений задачи FL и релаксированной задачи FL имеется некоторая взаимосвязь, то оптимальное решение X^* задачи $MINF_0$ и множество $I_0(V^*)$ имеют существенное пересечение. Если, наконец, предположить, что рассматриваемое тупиковое решение V таково, что множества $I_0(V)$ и $I_0(V^*)$ имеют достаточно широкое пересечение, то получаем, что элементы оптимального решения X^* задачи $MINF_0$ следует прежде всего искать среди элементов множества $I_0(V)$.

1.4 Алгоритмы вычисления тупиковых решений

Достаточно простое строение тупиковых решений задачи DFL позволяет разработать несложную вычислительную процедуру построения такого решения по исходной паре матриц (F_0, C) . Эту процедуру, как уже отмечалось, чаще всего называют *процедурой подъема*, поскольку в ходе этой процедуры элементы матрицы V монотонно возрастают от нуля до своих финальных значений.

Алгоритм построения тупикового решения для пары (F_0, C) состоит из конечного числа шагов, на каждом из которых производится увеличение некоторых компонент v_{ij_0} , $i \in I$, текущего решения $V=(v_{ij})$ так, чтобы увеличить величину u_{j_0} для выбранного на данном шаге j_0 -го столбца матрицы W .

На первом шаге имеется начальное решение $V=(v_{ij})$, у которого $v_{ij}=0$, $i \in I, j \in J$.

Пусть к началу очередного шага построено решение V . Шаг начинается с выбора элемента $j_0 \in J \setminus J_0(V)$. Если все столбцы матрицы V заблокированы, то есть если $J = J_0(V)$, то алгоритм заканчивает работу и V – искомое тупиковое решение. В противном случае для выбранного элемента j_0 вычисляется величина

$$\Delta j_0 = \min \left\{ \min_{i \in I_{j_0}(V)} d_i; \min_{i \notin I_{j_0}(V)} \{c_{ij_0} - u_{j_0}\} \right\}.$$

Эта величина Δj_0 есть наименьшая из следующих двух величин. Первая указывает на наименьший остаток тех ресурсов f_i , $i \in I$, которые обязательно должны быть использованы для увеличения величины u_{j_0} . Вторая показывает, на сколько должна быть увеличена величина u_{j_0} , если в качестве ограничения сверху для нового значения величины u_{j_0} берется наименьший элемент j_0 -го столбца матрицы C , превышающий текущее значение величины u_{j_0} . Далее строится новое решение посредством увеличения компонент v_{ij_0} , $i \in I_0(V)$ текущего решения V на величину Δj_0 . После этого начинается следующий шаг.

Отметим, что приведенное описание алгоритма построения тупикового решения V для пары матриц (F_0, C) не является полным, поскольку не определено, каким образом на каждом шаге алгоритма выбирается элемент $j_0 \in J \setminus J_0(V)$. Вместе с тем, именно выбор элемента j_0 приводит к построению той или иной оценочной матрицы с различными значениями нижних оценок. При этом спектр этих значений может быть достаточно широк.

Для примера рассмотрим несколько различных тупиковых решений для пары матриц (F_0, C) вида

$$F_0 = \begin{pmatrix} 5 \\ 6 \\ 6 \\ 6 \\ 5 \end{pmatrix}, \quad C = \begin{pmatrix} 1 & 1 & 6 & 4 \\ 1 & 4 & 2 & 4 \\ 3 & 1 & 5 & 2 \\ 4 & 6 & 2 & 1 \\ 6 & 3 & 3 & 1 \end{pmatrix}.$$

Для данной пары матриц тупиковыми будут, например, решения V_1, V_2, V_3 , приводящие к следующим оценочным матрицам

$$W_1 = \begin{pmatrix} 2 & 5 & 6 & 4 \\ 2 & 5 & 3 & 4 \\ 3 & 5 & 5 & 4 \\ 4 & 6 & 3 & 4 \\ 6 & 5 & 3 & 4 \end{pmatrix}, \quad W_2 = \begin{pmatrix} 3 & 4 & 6 & 4 \\ 3 & 4 & 6 & 4 \\ 3 & 4 & 6 & 2 \\ 4 & 6 & 6 & 2 \\ 6 & 4 & 6 & 2 \end{pmatrix}, \quad W_3 = \begin{pmatrix} 4 & 3 & 6 & 4 \\ 4 & 4 & 5 & 4 \\ 4 & 3 & 5 & 4 \\ 4 & 6 & 5 & 4 \\ 6 & 3 & 5 & 4 \end{pmatrix},$$

для которых $H(W_1)=14$, $H(W_2)=15$, $H(W_3)=16$,

Основное правило выбора элемента j_0 в рассматриваемом алгоритме построения тупикового решения зададим формулой

$$j_0 = \arg \min_{j \in J \setminus J_0(V)} |I_j(V)|.$$

Это правило устанавливает, что выбор номера j_0 производится среди номеров j незаблокированных столбцов, исходя из числа элементов в множестве $I_j(V)$, и в качестве j_0 выбирается такой номер j , для которого число элементов в множестве $I_j(V)$ наименьшее.

Такая стратегия выбора номера j_0 , разумеется, не является наилучшей и не всегда приводит к оптимальному решению. Однако эта стратегия представляется разумной, во-первых, потому что процедура ее осуществления не трудоемкая и, во-вторых, поскольку она отвечает основной цели — построить тупиковое решение, дающее хорошую нижнюю границу. Действительно, на каждом шаге алгоритма увеличение величины нижней оценки производится счет уменьшения остаточных ресурсов матрицы F_0 . Причем, увеличивая нижнюю оценку на единицу, мы уменьшаем суммарный ресурс матрицы F_0 на величину $|I_{j_0}(V)|$ единиц. Таким образом, выбранное правило определяет стратегию экономного расходования ресурса матрицы F_0 , при которой на каждом шаге алгоритма увеличение нижней оценки производится при минимальных суммарных затратах ресурсов.

Кроме указанного основного правила выбора элемента j_0 имеет смысл рассматривать и некоторые другие правила. Отметим, что одним из недостатков рассмотренного способа выбора элемента j_0 является то, что он не учитывает «дефицитности» остаточных ресурсов матрицы F_0 . В результате на очередном шаге алгоритма при небольшом суммарном расходе ресурсов, тем не менее, может быть израсходовано относительно много дефицитных ресурсов, то есть ресурсов, остатки которых малы и без которых не возможно «поднятие» минимума в других столбцах. Это, в конечном итоге, приводит к блокировке многих столбцов и невозможности увеличения нижней оценки на последующих шагах алгоритма. Количественной характеристикой дефицитности i -го ресурса матрицы F_0 можно считать величину d_i / f_i . Поэтому правило выбора номера j_0 с учетом дефицитности остаточных ресурсов матрицы F_0 определяется формулой

$$j_0 = \arg \min_{j \in J \setminus J_0(V)} \sum_{i \in I_j(V)} f_i \setminus d_i.$$

Можно также сформулировать правило выбора элемента j_0 , обобщающее рассмотренные выше правила. Оно задается формулой

$$j_0 = \arg \min_{j \in J \setminus J_0(V)} \sum_{i \in I_j(V)} (1 - R_1(1 - f_i \setminus d_i)),$$

где R_1 , $0 \leq R_1 \leq 1$, – некоторый параметр. Легко видеть, что если $R_1 = 0$, то это правило совпадает с основным, а если $R_1 = 1$, то с правилом, учитывающим дефицитность остаточных ресурсов. Меняя значение параметра R_1 можно подобрать наилучшую стратегию выбора элемента j_0 для того или иного класса пар матриц (F_0, C) .

Несложно оценить трудоемкость рассмотренной процедуры вычисления тупикового решения. Если используется основное правило выбора элемента j_0 , то число шагов, на которых величина $\min_{j \in J_0(V)} |I_j(V)|$ остается неизменной, очевидно, не превосхо-

дит n , а оценка общей трудоемкости этих шагов равняется величине $O(mn)$. Но поскольку в ходе работы алгоритма величина $\min_{j \in J_0(V)} |I_j(V)|$ может изменить свое значе-

ние не более m раз, то трудоемкость алгоритма в целом равняется величине $O(m^2n)$.

Применение обобщающего правила выбора элемента j_0 при значении параметра $R_1 > 0$ увеличивает трудоемкость алгоритма построения тупикового решения. Наиболее трудоемкая часть каждого шага алгоритма в этом случае связана с определением номера j_0 и требует $O(mn)$ действий. Поскольку общее число шагов алгоритма ограничено величиной mn , то временная сложность алгоритма в целом оценивается величиной $O(m^2n^2)$.

1.5 Двойственная задача в случае матрицы затрат на обслуживание в канонической форме

В заключение, приведем еще одну двойственную задачу, получаемую из релаксированной задачи FL , в которой исходная матрица C заменяется на каноническую форму \tilde{C} этой матрицы. Такая двойственная задача не дает улучшения нижней границы, но полезна, поскольку позволяет в более простом виде представить описанную выше процедуру построения тупикового решения.

Рассмотрим задачу FL с парой (F_0, C) . Пусть матрица C имеет характеристическую матрицу $H=(h_{is})$ размера $m \times s$ с весами столбцов b_s , $s=1, \dots, S$. Как известно, задача FL с исходной парой (F_0, C) эквивалентна задаче FL с парой (F_0, \tilde{C}) , где $\tilde{C} = (b_s h_{is})$ – каноническая форма и при этом оптимальные значения целевых функций этих задач отличаются на величину $\sum_{j \in J} \min_{i \in I} c_{ij}$.

Применительно к задаче FL с парой (F_0, \tilde{C}) задача DFL записывается следующим образом:

$$\begin{aligned} \max_{(v_{is})} \sum_{s=1}^S \min_{i \in I} \{b_s h_{is} + v_{is}\}; \\ \sum_{s=1}^S v_{is} \leq f_i, \quad i \in I; \\ v_{is} \geq 0, \quad i \in I, s = 1, \dots, S. \end{aligned}$$

Тупиковое решение $V = (v_{is})$ этой задачи, очевидно, имеет вид

$$v_{is} = \begin{cases} u_s, & \text{если } h_{is} = 0, \\ 0, & \text{иначе,} \end{cases}$$

где $u_s \leq b_s, s = 1, \dots, S$.

В силу этого, рассматриваемая двойственная задача может быть переписана следующим образом

$$\begin{aligned} \max_{(u_s)} \sum_{s=1}^S u_s; \\ \sum_{s=1}^S (1 - h_{is}) u_s \leq f_i, \quad i \in I; \\ 0 \leq u_s \leq b_s, \quad s = 1, \dots, S. \end{aligned}$$

Если (u_s) – допустимое решение этой задачи, то величина

$$\sum_{j \in J} \min_{i \in I} c_{ij} + \sum_{s=1}^S u_s$$

будет нижней оценкой для целевой функции задачи FL с исходной парой (F_0, C) .

С учетом приведенной записи задачи DFL и с учетом того, что столбцы характеристической матрицы \tilde{C} могут быть упорядочены по возрастанию числа нулевых элементов в столбце, приведенный выше алгоритм построения тупикового решения с основным правилом выбора элемента j_0 может быть переписан следующим образом.

Алгоритм состоит из S шагов, на каждом из которых вычисляется соответствующая величина u_s .

На s -м, $s = 1, \dots, S$, шаге величина u_s определяется по формуле

$$u_s = \min \{b_s; \min_{i|h_{is}=0} \{f_i - \sum_{t=1}^{s-1} (1 - h_{it}) u_t\}\}.$$

После этого, если $s < S$, начинается следующий шаг, иначе (u_s) – искомое тупиковое решение и работа алгоритма заканчивается.

В качестве примера использования данного алгоритма для вычисления оценочной матрицы построим такую матрицу \tilde{W} для рассмотренной выше пары матриц (F_0, C) ,

$$F_0 = \begin{pmatrix} 5 \\ 6 \\ 6 \\ 6 \\ 5 \end{pmatrix}, \quad C = \begin{pmatrix} 1 & 1 & 6 & 4 \\ 1 & 4 & 2 & 4 \\ 3 & 1 & 5 & 2 \\ 4 & 6 & 2 & 1 \\ 6 & 3 & 3 & 1 \end{pmatrix}.$$

Каноническая форма \tilde{C} данной матрицы C , у которой столбцы упорядочены по возрастанию числа нулей, выглядит следующим образом:

$$\tilde{C} = \begin{bmatrix} 0 & 0 & 1 & 1 & 0 & 0 & 2 & 2 & 0 & 1 \\ 0 & 2 & 0 & 1 & 0 & 1 & 0 & 2 & 0 & 0 \\ 2 & 0 & 1 & 1 & 0 & 0 & 2 & 0 & 0 & 0 \\ 2 & 2 & 0 & 0 & 1 & 1 & 0 & 0 & 2 & 0 \\ 2 & 2 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Тупиковая матрица \tilde{W} , получаемая в результате работы рассмотренной выше процедуры, как несложно понять, имеет вид

$$\tilde{W} = \begin{bmatrix} 2 & 2 & 1 & 1 & 1 & 0 & 2 & 0 & 1 \\ 2 & 2 & 1 & 1 & 1 & 1 & 2 & 0 & 0 \\ 2 & 2 & 1 & 1 & 1 & 0 & 2 & 0 & 0 \\ 2 & 2 & 1 & 1 & 1 & 1 & 2 & 2 & 0 \\ 2 & 2 & 1 & 1 & 1 & 0 & 2 & 0 & 0 \end{bmatrix}.$$

Следовательно, значение нижней границы для целевой функции задачи FL с исходной парой (F_0, C) равняется $5 + H(\tilde{W}) = 5 + 11 = 16$.

2 Эвристические гриды алгоритмы

Гриды процедура — одна из простейших и хорошо известных универсальных процедур построения приближенного решения задачи минимизации целевой функции, заданной на подмножествах некоторого конечного множества I . Основная идея этой процедуры, включающей конечное число шагов, состоит либо в добавлении на каждом шаге к текущему подмножеству еще одного элемента, либо в удалении на каждом шаге из текущего подмножества некоторого элемента. В первом случае процесс начинается с подмножества относительно малого размера, например, с пустого множества, а во втором — с некоторого достаточно обширного множества, например, всего множества I . При этом всякий раз для изменения текущего решения выбирается такой элемент, при котором либо значение целевой функции уменьшается максимально, либо максимально изменяется значение некоторого другого параметра, связанного с целевой функцией. Понятно, что построенные на основе такой вычислительной схемы алгоритмы являются достаточно простыми и малотрудоемкими. Понятно также, что в подавляющем большинстве интересных целевых функций такие алгоритмы не гарантируют получение оптимальных решений. Вместе с тем, для так называемых субмодулярных функций

[] алгоритмы, построенные на базе гриди процедуры, дают оптимальные решения. Существуют также примеры гриди алгоритмов решения задач *SC* и *FL* [179,], которые имеют априорные оценки точности, слабо растущие с ростом числа элементов множества *I*. Однако основным достоинством гриди процедуры является ее простота и универсальность. Поэтому на базе такой процедуры в сочетании с другими приемами могут быть построены нетрудоемкие эвристические алгоритмы для быстрого поиска приближенных решений. Такие решения, в частности, могут быть начальными допустимыми решениями для алгоритмов отыскания точных решений.

Задача *FL*, как показано ранее, может быть сформулирована в виде задачи $MINF_0$, то есть задачи минимизации функции $F_0(X)$, заданной на подмножествах *X* множества *I*. Поэтому схема гриди алгоритма в полной мере применима к задаче *FL* и, вероятно, впервые рассмотрена в [226]. В этой же работе рассмотрены возможные процедуры улучшения решения, полученного в результате применения основной гриди эвристики. Такой дополнительной процедурой, которая также работает по принципу гриди алгоритма, является, например, процедура одновременного исключения и добавления элементов. Эта процедура представляет собой последовательное преобразование текущего множества, состоящее в одновременном удалении из множества и добавлении к нему по одному элементу. При этом выбор такой пары элементов также производится, исходя из наибольшего уменьшения значения целевой функции.

Возвращаясь к основной гриди эвристики построения приближенного решения задачи *FL*, укажем на следующие три рассматриваемых ниже варианта такого алгоритма. Первый — простой гриди алгоритм, который фактически реализует общую схему гриди процедуры последовательного увеличения текущего решения *X* для задачи минимизации функции $F_0(X)$. Второй алгоритм использует тупиковое решение *V* задачи *DFL* для построения нижней оценки $H(W)$ и множества $I_0(V)$, элементы которого рассматриваются как перспективные на вхождение в «хорошее» приближенное решение. Этот алгоритм представляет собой процедуру последовательного уменьшения текущего приближенного решения, начиная с множества $I_0(V)$. Третий алгоритм — вероятностная модификация первого алгоритма. Вероятностными называют алгоритмы, содержащие процедуры, часть параметров которых не являются детерминированными, а определяются случайным образом. Такие алгоритмы широко используются при исследовании дискретных экстремальных задач и, в частности, задачи *FL*. В качестве примера таких исследований можно привести работу [199], содержащую описание вероятностных гриди процедур для решения различных задач и, в том числе, задачи *FL*. Исследованию вероятностных алгоритмов с помощью вычислительных экспериментов посвящена работа [46]. В рассматриваемом гриди алгоритме вероятностной является процедура формирования на каждом шаге алгоритма множества I' , из которого выбирается элемент, увеличивающий текущее решение. В этом множестве может оказаться любой элемент множества *I*, не входящий в текущее решение, но преимущественные шансы имеют элементы множества $I_0(V)$.

Каждый из трех представленных алгоритмов может быть дополнен разного рода процедурами улучшения полученного решения, аналогичными упомянутой выше про-

цедуре их [226]. И хотя с их помощью в некоторых случаях можно добиться существенного улучшения приближенного решения, далее такие процедуры рассматриваться не будут.

2.1 Простейший гриди алгоритм

Описание рассматриваемых гриди эвристик, как отмечено выше, удобней давать применительно к задаче $MINF_0$, которая, как известно, состоит в отыскании среди подмножеств $X \subset I$ такого множества X^* , которое дает минимум функции

$$F_0(X) = \sum_{i \in X} f_i + \sum_{j \in J} \min_{i \in X} c_{ij}.$$

Предлагаемый *простейший гриди алгоритм* решения задачи $MINF_0$ состоит из конечного числа шагов, на каждом из которых текущее решение X расширяется за счет некоторого выбираемого на данном шаге элемента $i_0 \in I \setminus X$. Этот процесс продолжается до тех пор, пока получаемое на шаге новое решение лучше, чем имеющееся.

На первом шаге $X \neq \emptyset$.

Пусть к очередному шагу найдено некоторое множество X . Шаг начинается с вычисления для всякого $i \in I \setminus X$ величины

$$F_i(X) = F_0(X) - F_0(X \cup \{i\}).$$

Далее определяется элемент i_0 такой, что

$$i_0 = \arg \max_{i \in I \setminus X} F_i(X).$$

Если $F_{i_0}(X) \leq 0$, то алгоритм заканчивает работу и X – искомое приближенное решение. В противном случае, множество X заменяется на множество $X \cup \{i_0\}$ и начинается следующий шаг.

Для решения X , найденного в результате работы рассмотренного алгоритма, можно вычислить апостериорную оценку точности.

Положим для всякого $j \in J$

$$u_j(X) = \min_{i \in X} c_{ij}$$

и покажем, что $(u_j(X))$ – допустимое решение двойственной задачи (2.1.9) – (2.1.10). Для этого заметим, что при любом $i \in I$ выполняется неравенство

$$\sum_{j \in J} (u_j(X) - c_{ij})^+ \leq f_i.$$

Действительно, если $i \in I$, то, поскольку для всякого $j \in J$ имеем $u_j(X) \leq c_{ij}$, требуемое неравенство, очевидно, выполняется. Если же $i \notin X$, то, поскольку $F_i(X) \leq 0$, получаем

$$F_i(X) = F_0(X) - F_0(X \cup \{i\}) = \sum_{j \in J} \min_{k \in X} c_{kj} -$$

$$-\sum_{j \in J} \min_{k \in X \cup \{i\}} c_{kj} - f_i = \sum_{j \in J} (u_j(X) - c_{ij})^+ - f_i \leq 0.$$

Таким образом, величина $\sum_{j \in J} u_j(X)$ является оценкой снизу для оптимального

значения $F_0(X^*)$ целевой функции задачи $MINF_0$. Это позволяет получить оценку точности для найденного в результате работы алгоритма решения X

$$F(X) - F(X^*) \leq F(X) - \sum_{j \in J} u_j(X) = \sum_{i \in X} f_i. \quad (2.2.1)$$

Отсюда видно, что за исключением малоинтересного случая, когда величины f_i , $i \in I$, относительно малы, полученная оценка точности является достаточно грубой. Это означает, что либо само решение X , вычисленное с помощью рассмотренного гриди алгоритма, является плохим, либо используемая оценка для величины $F_0(X^*)$ не является удовлетворительной. Относительно оценки данное утверждение действительно справедливо, поскольку вектор $(u_j(X))$ порождается решением (v_{ij}) задачи DFL , для которого $v_{ij} = 0$, $j \in J$, при любом $i \in X$. Такое решение (v_{ij}) нельзя считать «хорошим» решением задачи DFL . Что касается самого решения X , то его также нельзя считать «хорошим». Дело в том, что поскольку в ходе работы алгоритма на выбор элементов, пополняющих текущее решение, не накладывается никаких ограничений, то возможна ситуация, когда на первых шагах гриди процедуры в текущее решение попадают «плохие» элементы, блокирующие возможность дальнейшего улучшения текущего решения. В любом случае неудовлетворительное качество решения, получаемого в результате работы рассмотренного алгоритма, не является неожиданными, поскольку в ходе его работы практически не используется специфика задачи FL . Эта специфика может учитываться, в частности, посредством ограничений на выбор элементов, формирующих в ходе работы алгоритма текущее решение.

2.2 Гриди алгоритм с использованием блокирующего множества

Как уже неоднократно отмечалось, использование тупикового решения V , построенного по паре матриц (F_0, C) , позволяет не только вычислить достаточно хорошую нижнюю границу $H(W)$, но и дает возможность получить с помощью блокирующего множества $I_0(V)$ дополнительную информацию для построения хорошего приближенного решения X . Эта информация состоит в том, что предположительно элементы оптимального решения X^* прежде всего следует искать среди элементов блокирующего множества $I_0(V)$.

Пусть $V = (v_{ij})$ – тупиковое решение, а $X \subset I_0(V)$ – некоторое критическое множество. Напомним, что множество X называется критическим, если $X \cap I_j(V) \neq \emptyset$ при любом $j \in J$. При фиксированном X для всякого $j \in J$ положим

$$c_j = \min_{i \in X} c_{ij},$$

$$i(j) = \arg \min_{i \in X} c_{ij}.$$

Понятно, что поскольку V – тупиковое решение, то для любого $j \in J$ имеем

$$v_{i(j)j} = \max_{i \in X} v_{ij}.$$

Следующая лемма устанавливает равенство, позволяющее получить оценку точности для приближенного решения, являющегося критическим множеством.

Лемма 2.1. Если $V = (v_{ij})$ – тупиковое решение, $W = (w_{ij})$ – оценочная матрица, порожденная V , а X – критическое множество, то

$$F_0(X) = H(W) + \sum_{j \in J} \sum_{i \in X \setminus \{i(j)\}} v_{ij}.$$

Доказательство. Справедливость утверждения вытекает из следующей цепочки равенств:

$$\begin{aligned} H(W) &= \sum_{j \in J} \min_{i \in I} w_{ij} = \sum_{j \in J} \min_{i \in X} w_{ij} = \\ &= \sum_{i \in X} \sum_{j | i(j)=i} w_{ij} = \sum_{i \in X} \left\{ \sum_{j \in J} v_{ij} + \sum_{j | i(j)=i} c_{ij} - \sum_{j | i(j) \neq i} v_{ij} \right\} = \\ &= \sum_{i \in X} f_i + \sum_{j \in J} c_{i(j)j} - \sum_{j \in J} \sum_{i \in X \setminus \{i(j)\}} v_{ij} = F_0(X) - \sum_{j \in J} \sum_{i \in X \setminus \{i(j)\}} v_{ij}. \end{aligned}$$

Присутствующую в доказанном равенстве двойную сумму обозначим через $D(V, X)$ и назовем *дефектом критического множества X относительно тупикового решения V* . В качестве следствия из доказанного равенства получаем следующую оценку точности приближенного решения X , являющегося критическим множеством

$$F_0(X) - F_0(X^*) \leq H(W) + D(V, X) - H(W) = D(V, X).$$

Отсюда видно, что для критического множества оценка точности равняется дефекту этого множества и, следовательно, эта оценка существенно лучше полученной ранее оценки (2.2.1) для приближенного решения, даваемого первым алгоритмом. Кроме того, ясно, что чем меньше число элементов в критическом множестве, тем меньше величина дефекта и тем меньше значение целевой функции. Поэтому в качестве приближенного решения следует использовать минимальное критическое множество, то есть такое критическое множество, удаление из которого любого элемента превращает его в множество не являющееся критическим.

Для приближенного решения, представляющего собой минимальное критическое множество, доказанная лемма позволяет сформулировать *критерий оптимальности* этого решения. Если критическое множество X такое, что $|X \cap I_j^+(V)| \leq 1$ для всякого $j \in J$, то данное множество является оптимальным решением. Действительно, в этом случае имеем $(X \setminus \{i(j)\}) \cap I_j^+ = \emptyset$ при любом $j \in J$ и, следовательно, дефект $D(V, X)$ равен нулю. Из сказанного получаем, в частности, что если X – одноточечное критическое множество, то X – оптимальное решение.

Суммируя сказанное выше, можно заключить, что есть все основания для рассмотрения гриди алгоритма построения приближенного решения в виде минимального критического множества. Такой алгоритм естественно представлять как процедуру по-

следовательного уменьшения текущего критического множества, начиная с множества $I_0(V)$, до тех пор пока, не будет найдено наименьшее множество. Выбор элемента i_0 , уменьшающего текущее критическое множество, производимый по правилу наибольшего уменьшения значения целевой функции, будет соответствовать максимальному уменьшению на каждом шаге дефекта критического множества. Поэтому такой алгоритм будет ориентирован на построение минимального критического множества с наименьшим дефектом. При этом отметим, что, вообще говоря, возможно уменьшение минимального критического множества, приводящее к уменьшению значения целевой функции. Поэтому рассматриваемая гриди процедура не обязательно должна останавливаться на минимальном критическом множестве, а может быть продолжена до тех пор, пока не будет найдено подмножество минимального критического множества, не улучшаемое по значению целевой функции.

Дадим более подробное описание этого алгоритма.

Гриди алгоритм с использованием блокирующего множества состоит из предварительного шага и конечного числа однотипных основных шагов, на каждом из которых текущее решение X уменьшается за счет некоторого выбираемого на данном шаге элемента $i_0 \in X$. При этом на первых шагах до тех пор, пока не будет построено минимальное критическое множество, выбор элемента i_0 ограничивается тем требованием, что множество $X \setminus \{i_0\}$ должно быть критическим множеством. Процесс продолжается до тех пор, пока уменьшение текущего множества не приведет к возрастанию значения целевой функции.

На предварительном шаге по исходной паре матриц (F_0, C) вычисляется тупиковое решение V , строится блокирующее множество $I_0(V)$ и множества $I_j(V)$, $j \in J$.

На первом основном шаге $X = I_0(V)$.

Пусть к очередному основному шагу построено множество X . Шаг начинается с построения множества $X' = \{i \in I \mid (X \setminus \{i\}) \cap I_j(V) \neq \emptyset, j \in J\}$. Если $X' \neq \emptyset$, то полагается $X' = X$. Далее, для всякого $i \in X'$ вычисляется величина

$$F_i^-(X) = F_0(X) - F_0(X \setminus \{i\})$$

и после этого определяется номер i_0 такой, что

$$i_0 = \arg \max_{i \in X'} F_i^-(X).$$

Если $F_{i_0}^-(X) < 0$, то алгоритм заканчивает работу и X – искомое приближенное решение. В противном случае множество X заменяется на множество $X \setminus \{i_0\}$ и начинается следующий шаг.

Отметим, что данный алгоритм является корректным в том смысле, что на некотором его шаге будет построено минимальное критическое множество. Покажем, что если $X \setminus \{i_0\}$ – критическое множество, то $F_{i_0}^-(X) \geq 0$. Действительно, для всякого $j \in J$ положим

$$c_j = \min_{i \in X \setminus \{i_0\}} c_{ij}.$$

Поскольку $X \setminus \{i_0\}$ – критическое множество, то $c_j \leq u_j$, $j \in J$. Обозначим через $J(i_0)$ множество $\{j \in J \mid c_j > c_{i_0j}\}$. Справедливы следующие соотношения

$$\begin{aligned} F_{i_0}^-(X) &= F_0(X) - F_0(X \setminus \{i_0\}) = f_{i_0} + \sum_{j \in J(i_0)} c_{i_0j} - \sum_{j \in J(i_0)} c_j \geq \\ &\geq \sum_{j \in J(i_0)} (c_{i_0j} + v_{i_0j}) - \sum_{j \in J(i_0)} c_j = \sum_{j \in J(i_0)} (u_j - c_j) \geq 0, \end{aligned}$$

доказывающее требуемое.

Таким образом, в результате работы алгоритма будет построено приближенное решение X , являющееся подмножеством некоторого минимального критического множества.

2.3 Вероятностный гриди алгоритм

Вероятностными, как уже отмечалось, называют алгоритмы, включающие процедуру, некоторые параметры которой не являются детерминированными, а принимают значения случайным образом. Конкретные значения таких параметров в ходе работы алгоритма определяются, например, с помощью датчика случайных чисел.

Предлагаемый вероятностный гриди алгоритм можно рассматривать как некоторую модификацию простейшей гриди процедуры. Основное отличие этого алгоритма состоит в том, что на шаге алгоритма выбор элемента, увеличивающего текущее решение, производится не из всего множества I , а некоторого его подмножества I' . При этом процедура формирования множества I' является вероятностной, а «вероятности» попадания в множество I' определяются таким образом, чтобы в него в принципе могли войти любые элементы множества I , но определенное преимущество имели бы элементы из множества $I_0(V)$. С помощью вероятностной процедуры формирования множества I' создается ситуация, когда в построении приближенного решения будут участвовать, в основном, элементы множества $I_0(V)$, но не только они.

Вероятностная процедура формирования множества I' предполагает задание для всякого $i \in I$ величины p_i , $0 \leq p_i \leq 1$, определяющей «вероятность» попадания элемента i в множество I' . В рассматриваемом случае при заданном тупиковом решении V эти величины имеют вид

$$P_i = R_0(1 - d_i / f_i),$$

где R_0 , $0 \leq R_0 \leq 1$ – некоторый параметр.

Если величины p_i определены, то вероятностная процедура формирования множества I' работает следующим образом. Для всякого $i \in I$ с помощью датчика случайных чисел выбирается случайная величина q_i , $0 \leq q_i \leq 1$. Если $q_i \leq p_i$, то элемент i зачисляется в множество I' .

Дадим теперь описание самого алгоритма.

Вероятностный гриди алгоритм состоит из предварительного шага и конечного числа однотипных основных шагов, аналогичных шагам простейшего гриди алгоритма.

На предварительном шаге по исходной паре матриц (F_0, C) вычисляется тупиковое решение V и с его помощью определяются величины $p_i, i \in I$.

На первом основном шаге $X = \emptyset$.

Пусть к очередному основному шагу построено множество X . Шаг начинается с формирования посредством вероятностной процедуры множества I' . Если $I' \setminus X = \emptyset$, то алгоритм заканчивает работу и X – искомое приближенное решение. В противном случае для всякого $i \in I' \setminus X$ вычисляется величина

$$F_i^+(X) = F_0(X) - F_0(X \cup \{i\})$$

и после этого определяется номер i_0 такой, что

$$i_0 = \arg \max F_i^+(X).$$

Если $F_{i_0}^+(X) \leq 0$, то алгоритм заканчивает работу и X – искомое решение. В противном случае множество X заменяется на множество $X \cup \{i_0\}$ и начинается следующий шаг.

В результате работы вероятностного гриди алгоритма получаем приближенное решение X , которое является случайным множеством. Поэтому в результате применения нескольких применений алгоритма будут получены, вообще говоря, различные приближенные решения. Если в результате k применений алгоритма найдены приближенные решения X_1, \dots, X_k , то лучшее из этих решений $X(k)$ считаем результатом k применений алгоритма.

3 Алгоритмы неявного перебора

Поскольку множества допустимых решений рассматриваемых задач FL , $MINF_0$ и $MINP_0$, как и других задач дискретной оптимизации, суть конечные множества, то для их решения может быть использован простой метод – метод прямого перебора. С помощью прямого перебора всегда можно, по крайней мере теоретически, найти оптимальное решение. Поэтому простой перебор можно считать универсальным способом решения дискретных экстремальных задач. Конечно, если число допустимых решений задачи достаточно велико, то такой метод оказывается практически нереализуемым. Однако некоторая привлекательность в силу своей простоты и универсальности в таком подходе все же имеется и метод перебора может быть востребован, если различного рода улучшениями привести его к виду, приемлемому для практического использования.

Среди таких модифицированных схем перебора наибольшее распространение получил так называемый метод ветвей и границ [17, 57, 198]. Именно этот метод, особенно в сочетании с другими менее трудоемкими процедурами оказался полезным при решении «классически трудных» экстремальных задач.

В методе ветвей и границ, в отличие от простого перебора, в котором наилучшее текущее решение сравнивается с очередным допустимым решением, такое решение сравнивается с целым множеством допустимых решений. При этом устанавливается признак, позволяющий в некоторых случаях убедиться в том, что данное подмножество не содержит решений лучше, чем наилучшее текущее решение. Таким образом, метод ветвей и границ можно коротко охарактеризовать как метод, осуществляющий поиск оптимального решения посредством последовательного разбиения множества допустимых решений на все более мелкие подмножества и последующего сравнения этих подмножеств с наилучшим текущим решением.

Использование метода ветвей и границ для построения алгоритмов решения задачи FL можно считать успешным. Описанию таких алгоритмов посвящено значительное число работ. Наиболее известными из них являются [212, 222], в которых сделаны, вероятно, первые попытки построения алгоритмов решения задачи FL на основе метода ветвей и границ. Достаточно проработанные варианты таких алгоритмов описаны в [17, 51]. Эти алгоритмы построены независимо разными группами авторов и достаточно близки по своему строению и работоспособности.

3.1 Метод ветвей и границ

В основе метода ветвей и границ, как отмечено выше, лежит идея последовательного разбиения множества допустимых решений на подмножества и последующее сравнение таких подмножеств с наилучшим текущим решением. Такое наилучшее решение называется *рекордом*, а значение целевой функции на этом решении – *рекордным*. Сравнение подмножества с рекордом производится с целью проверки, содержит ли данное множество решение лучше рекорда. Если рассматривается задача минимизации, то проверка осуществляется посредством вычисления оценки снизу для множества значений целевой функции на данном подмножестве. Если оценка снизу не меньше рекордного значения целевой функции, то, очевидно, рассматриваемое подмножество не содержит решения лучше рекорда и, следовательно, это множество решений может быть отброшено. Проверяемое множество отбрасывается не только в указанном случае, но и тогда, когда в нем удастся найти наилучшее решение. При этом, если найденное решение лучше рекорда, то происходит смена рекорда.

Если удастся отбросить все элементы разбиения множества допустимых решений, то поиск оптимального решения заканчивается и таким решением является рекорд. В противном случае, из неотброшенных элементов разбиения выбирается множество, являющееся наиболее *перспективным* в смысле содержания оптимального решения, которое подвергается дальнейшему разбиению (ветвлению) на более мелкие подмножества. После этого элементы нового разбиения множества неотброшенных допустимых решений подвергаются проверке и т.д.

Дадим теперь более формальное описание алгоритма ветвей и границ. Изложение будем вести применительно к задаче минимизации целевой функции $f(x)$ на множестве D , то есть задаче вида

$$\min \{f(x) \mid x \in D\}.$$

При этом будем дополнительно предполагать, что задано некоторое разбиение множества D на конечное число так называемых *базисных* множеств. В случае, когда множество D – конечное множество, роль базисных могут играть одноточечные множества. При описании алгоритма ветвей и границ будем рассматривать только такие множества $d \subset D$, которые есть объединение базисных множеств. Неформально, базисные множества можно рассматривать как подмножества максимальной мощности, на которых рассматриваемая задача существенно более легкая по сравнению с исходной. Будем считать, что существует нетрудоемкая (эффективная) процедура, позволяющая отыскивать оптимальное решение задачи минимизации функции $f(x)$ на любом базисном множестве.

Оптимальное решение исследуемой задачи обозначим через x^* и будем считать, что $f(x^*) > 0$.

Функцию $b(d)$, определенную на подмножествах множества D и ставящую в соответствие множеству $d \subset D$ определенное разбиение его на несобственные подмножества d_1, d_2, \dots, d_N будем называть *функцией ветвления*. Вещественную функцию $H(d)$, определенную на подмножествах множества D и такую, что

$$0 < H(d) < \min_{x \in d} f(x)$$

назовем *функцией вычисления нижней границы* или коротко *нижней границей*. Эту функцию будем считать невозрастающей, то есть такой, что $H(d_1) \geq H(d_2)$, если $d_1 \subset d_2$. Функцию $x(d)$, определенную на подмножествах множества D , включая все базисные множества и определяющую по множеству d оптимальное решение задачи минимизации функции $f(x)$ на множестве d , назовем *функцией выбора наилучшего решения*.

Алгоритм, реализующий метод ветвей и границ, состоит из конечной последовательности однотипных шагов. На каждом шаге рассматривается некоторое разбиение t_1, \dots, t_L множества еще неотброшенных допустимых решений и некоторый рекорд x^0 .

На первом шаге имеем $t_1 = D$, x^0 – произвольный элемент множества D .

Пусть к очередному шагу имеется разбиение t_1, \dots, t_L и рекорд x^0 . Шаг начинается с проверки элементов разбиения (не обязательно всех) на предмет выяснения, во-первых, содержит ли оно решение лучше рекорда и, во-вторых, какое решение в подмножестве является наилучшим. Предположим, что проверяется множество t_l . Это множество считается проверенным и отбрасывается, если выполняется одно из двух условий:

- 1) $H(t_l) \geq f(x^0)$;
- 2) функция $x(t)$ определена на множестве t_l .

При этом, если реализуется второй случай и $f(x(t_l)) < f(x^0)$, то устанавливается новое значение рекорда $x^0 = x(t_l)$.

Пусть $t'_1, \dots, t'_{L'}, L' \leq L$, – множества не отброшенные в результате проверок. Если $L' = 0$, то есть отброшенными оказываются все элементы разбиения, алгоритм заканчивает работу и x^0 – решение, найденное в результате его работы. Если $L' > 0$, то среди неотброшенных элементов разбиения отыскивается некоторое «перспективное» под-

множество. Пусть таковым является множество t_{l_0} . Тогда к множеству t_{l_0} применяется функция ветвления $b(d)$, в результате чего получается разбиение d_1, \dots, d_N множества t_{l_0} и в целом новое разбиение $t'_1, \dots, t'_{l_0-1}, d_1, \dots, d_N, t'_{l_0+1}, \dots, t'_{L'}$ множества неотброшенных решений. После этого начинается следующий шаг.

Удостоверимся, что приведенный алгоритм заканчивает работу через конечное число шагов, а результатом его работы является оптимальное решение. Действительно, конечность алгоритма вытекает из конечности числа базисных множеств, того, что на каждом шаге алгоритма хотя бы один элемент разбиения либо отбрасывается, либо разбивается на подмножества, каждое из которых содержит меньшее число базисных множеств и, наконец, из того, что базисные множества всегда отбрасываются.

Покажем теперь, что полученный в результате работы алгоритма рекорд x^0 является оптимальным решением. Предположим, это не так и пусть $f(x^0) > f(x^*)$. Тогда на некотором шаге алгоритма элемент x^* был отброшен вместе с некоторым множеством t . Но множество t не может быть отброшено по первому условию, поскольку в этом случае будут выполняться противоречащие исходному предположению неравенства

$$f(x^0) \leq f(x_1^0) \leq H(t) \leq f(x^*),$$

где x^0 – рекорд, действующий на момент проверки множества t . Значит на множестве t определена функция выбора наилучшего решения $x(d)$ и это множество отброшено по второму условию. Но тогда рекорд x_1^0 должен быть заменен на решение $x(t)$ такое, что $f(x(t)) = f(x^*)$. Отсюда получаем что $f(x^0) = f(x^*)$. Это противоречит исходному предположению и доказывает оптимальность решения x^0 .

Заметим, возвращаясь к описанию алгоритма ветвей и границ, что оно не является полным, поскольку одно место в нем нуждается в уточнении. Действительно, не конкретизирован способ выбора «перспективного» элемента разбиения, то есть подмножества, к которому применяется функция ветвления. Имеется два основных правила выбора такого подмножества: правило *одновременного ветвления* и правило *одностороннего ветвления*. При одновременном ветвлении функция ветвления может быть применена к любому элементу разбиения. Чаще всего выбор элемента t_{l_0} производится по правилу

$$l_0 = \arg \min_{1 \leq l \leq L} H(t_l).$$

При одностороннем ветвлении выбор разбиваемого подмножества предписан от начала до конца. В таком случае считается, что перспективным является, например, первый или последний элемент разбиения.

Представленный выше алгоритм ветвей и границ будем называть еще *основной вычислительной схемой* метода ветвей и границ соответственно с одновременной или односторонней схемами ветвления. Отметим, что поскольку при одновременной схеме ветвления перспективным может оказаться любой элемент разбиения, такая схема требует значительных ресурсов для хранения информации об элементах разбиения. Напротив, в случае одностороннего ветвления, когда выбор перспективного множества предписан, можно использовать существенно более экономные способы задания ин-

формации о разбиении множества неотброшенных решений. Разумеется, возможна комбинация указанных способов ветвления и она оказывается полезной в модифицированном алгоритме ветвей и границ, называемом *алгоритмом с ограниченным числом элементов разбиения*.

Пусть имеется целочисленный параметр R_2 , ограничивающий число элементов в разбиении, полученном в результате одновременного ветвления. Если на очередном шаге алгоритма ветвей и границ с ограниченным числом элементов разбиения после применения функции ветвления число элементов разбиения превышает значение параметра R_2 , то, начиная со следующего шага, перспективный элемент разбиения подвергается так называемой *глобальной проверке*. Это происходит до тех пор, пока число элементов разбиения не будет превышать величину R_2 . Глобальная проверка означает, что на последующих шагах алгоритма до тех пор, пока глобально проверяемое множество не будет отброшено, будут рассматриваться и проверяться только подмножества данного множества. При этом может быть использована схема одностороннего ветвления или какой-либо иной способ поиска наилучшего решения глобально проверяемого множества.

Это замечание о способах ветвления завершает описание общей схемы алгоритма ветвей и границ в том числе и с ограниченным числом элементов разбиения. Разумеется при доведении этой общей схемы до конкретного вычислительного алгоритма решения той или иной экстремальной задачи необходима детальная конкретизация данной общей схемы, учитывающая особенности решаемой экстремальной задачи. В частности, исходя из специфики рассматриваемой задачи, необходимо конкретизировать и определить следующие составные элементы общей схемы:

- базисные множества решений;
- способ задания подмножеств решений;
- функцию ветвления;
- способ вычисления нижней границы;
- функцию выбора наилучшего решения;
- правило выбора перспективного элемента разбиения.

Ниже будет дана конкретизация указанных элементов общей схемы применительно к задаче *FL*. Однако прежде, оставаясь на уровне общей схемы, рассмотрим вопрос о трудоемкости алгоритмов ветвей и границ.

3.2 Трудоемкость алгоритмов ветвей и границ

Говоря о трудоемкости или временной сложности алгоритмов ветвей и границ, следует, прежде всего, подчеркнуть, что полиномиальную оценку для числа шагов таких алгоритмов получить не удастся. Это связано с трудностями принципиального характера, поскольку из свойств функции нижней границы и функции выбора наилучшего решения никак не следует, что в ходе работы алгоритма хотя бы одно подмножество, отличное от базисного, будет отброшено. Поэтому теоретически алгоритм ветвей и границ может в итоге свестись к просмотру всех базисных множеств, что в случае задачи дискретной оптимизации соответствует простому перебору множества решений.

Однако в действительности дело обстоит не совсем так плохо, как может показаться из сказанного. Хотя априори и нельзя установить, какая часть просматриваемых подмножеств будет отброшена в ходе работы алгоритма, тем не менее построить нижнюю границу, с помощью которой оказывается возможным в случае каждой конкретной задачи отбрасывать достаточно большое число подмножеств, часто удается. В результате алгоритм может оказаться практически работоспособным. При этом выводы о работоспособности алгоритма должны быть основаны на анализе апостериорных показателей качества алгоритма, полученных в результате проведения тестовых расчетов.

Каковы же пути построения работоспособного алгоритма? Вероятно, алгоритм будет приближаться к таковому, если в ходе работы алгоритма относительно большое число проверяемых множеств отбрасывается. Эта цель может быть достигнута, если выполняются следующие два условия. Прежде всего, необходимо, чтобы значение нижней границы, вычисляемое на подмножестве решений, было близко к наименьшему значению целевой функции на этом подмножестве. Кроме того, удачный исход проверки зависит и от того, насколько хорошим, то есть близким к оптимальному решению, является текущий рекорд. В силу этого, для алгоритма существенно как можно быстрее получить хороший рекорд и продолжать работу с этим рекордом. Важное значение имеет также использование разного рода признаков оптимальности, позволяющих определить функцию выбора наилучшего решения для относительно большого числа проверяемых множеств и, тем самым, отбросить эти множества. Однако решающее значение имеет точность используемой нижней границы. Хотя при этом следует иметь в виду, что стремление к использованию «хорошей» нижней границы может не привести к желаемому результату. Дело в том, что такая нижняя граница может потребовать неоправданно большого времени для вычисления своих значений. В результате, алгоритм с такой нижней границей будет требовать для своей реализации гораздо большего времени, чем алгоритм с более грубой нижней границей. Поэтому при построении нижней границы возникает задача соизмерения ее точности и скорости вычисления. Этот вопрос можно считать центральным вопросом, встающим при построении алгоритма ветвей и границ. От того, насколько правильно определен для нижней границы баланс скорости вычисления и точности, зависит трудоемкость алгоритма.

Возможность реализации указанных рекомендаций, по построению работоспособного алгоритма ветвей и границ, разумеется, зависит от специфики исследуемой оптимизационной задачи. Именно использование в полной мере особенностей задачи дает возможность построить работоспособный алгоритм ветвей и границ.

3.3 Приближенные алгоритмы ветвей и границ

Алгоритмы ветвей и границ, работающие по описанной выше схеме, являются точными алгоритмами и позволяют в принципе получать оптимальное решение задачи. Более того, часто отыскание такого решения осуществляется на относительно ранних шагах алгоритма, а остальные шаги необходимы лишь для того, чтобы убедиться в оптимальности найденного решения. При этом даже в случае достаточно работоспособного алгоритма с хорошей нижней границей реализация всех шагов алгоритма может потребовать значительных затрат времени. Поэтому имеет смысл рассмотреть модифи-

кации точных алгоритмов ветвей и границ, требующие для своей реализации существенно меньших затрат времени, но не гарантирующих получение оптимального решения и превращающихся, тем самым, в приближенные алгоритмы.

Рассмотрим некоторые такие модификации. Первая из них уменьшает время работы алгоритма ветвей и границ за счет задания ограничения сверху для числа шагов алгоритма или для числа полученных элементов разбиения и дает, тем самым, приближенное решение с апостериорной оценкой точности. Вторая не ограничивает общего числа шагов, но уменьшает это путем использования вместо функции $x(d)$ выбора наилучшего решения функцию $x'(d)$ выбора приближенного решения, определенную на достаточно широком множестве подмножеств и дающую некоторое «хорошее» решение из множества d . Если известна априорная оценка точности функции $x'(d)$, то данная модификация алгоритма ветвей и границ имеет априорную оценку точности, в противном случае получаем приближенное решение с апостериорной оценкой. Наконец, третья модификация также уменьшает число шагов алгоритма путем «искусственного» увеличения величины нижней границы. Эта модификация приводит к алгоритму с априорной оценкой точности. Приведем более подробное описание указанных вариантов приближенных алгоритмов ветвей и границ.

Приближенный алгоритм ветвей и границ с ограниченным числом шагов отличается от рассмотренного ранее основного алгоритма ветвей и границ только правилом остановки. Если в конце текущего шага перед применением функции ветвления выясняется, что номер N данного шага равняется заданному ограничению N_0 на число шагов или что число элементов разбиения L превышает заданную величину L_0 , то работа алгоритма прекращается, не смотря на то, что элементы t_1, \dots, t_L текущего разбиения остались неотброшенными. Вычисленный к концу шага рекорд x^0 считается результатом работы алгоритма. Для полученного приближенного решения x^0 можно вычислить апостериорную оценку точности. Действительно, величина

$$H = \min \{H(t_1), \dots, H(t_L)\},$$

очевидно, является нижней оценкой для оптимального значения $f(x^*)$ целевой функции. Поэтому можно написать

$$f(x^0) \leq \frac{f(x^0)}{H} f(x^*).$$

Отсюда получаем, что величина $f(x^0)/H$ есть апостериорная оценка точности найденного приближенного решения x^0 . Значение этой оценки, конечно, зависит от числа шагов, проделанных алгоритмом, и будет уточняться с ростом числа шагов.

Приближенный алгоритм ветвей и границ с функцией выбора приближенного решения. Данный алгоритм отличается от основной вычислительной схемы метода ветвей и границ только тем, что вместо функции $x(t)$ выбора наилучшего решения используется функция $x'(t)$ выбора приближенного решения. При этом точность алгоритма гарантируется точностью функции $x'(t)$. Если для всякого множества t , для которого определена функция $x'(t)$, выполняется неравенство $f(x'(t)) \leq (1 + \varepsilon)f(x^*(t))$, где $x^*(t)$ – наилучшее решение множества t , то $f(x^0) \leq (1 + \varepsilon)f(x^*)$. Действительно, пред-

положим противное и пусть $f(x^0) > (1 + \varepsilon)f(x^*)$. Тогда $f(x^0) \neq f(x^*)$ и, следовательно, решение x^* на некотором шаге алгоритма было отброшено вместе с множеством t . Поскольку $f(x^0) \neq f(x^*)$, то множество t не могло быть отброшено по первому условию. Если оно отброшено по второму условию, то можем написать

$$f(x^0) \leq f(x'(t)) \leq (1 + \varepsilon)f(x^*(t)) = (1 + \varepsilon)f(x^*).$$

Полученное неравенство противоречит предположению, что доказывает справедливость требуемого.

Помимо априорной оценки точности рассмотренного приближенного алгоритма для найденного в результате его работы решения x^0 может быть вычислена также апостериорная оценка точности. Для этого в ходе работы алгоритма должна быть вычислена величина H_0 , равная наименьшему значению функции $H(d)$ по всем множествам t , отброшенным в процессе работы алгоритма по второму условию, то есть по всем рассмотренным в ходе работы алгоритма множествам, для которых определена функция $x'(d)$.

Если для полученного решения x^0 выполняется неравенство $f(x^0) \leq H_0$, то x^0 – оптимальное решение, поскольку все отброшенные по второму признаку множества не содержат решения лучше, чем x^0 .

Если же $f(x^0) > H_0$, то справедливо неравенство

$$f(x^0) \leq \frac{f(x^0)}{H_0} f(x^*),$$

то есть $f(x^0)/H_0$ есть апостериорная оценка точности приближенного решения x^0 .

Действительно, предположим противное и пусть $H_0 > f(x^*)$. Тогда $f(x^0) \neq f(x^*)$ и, следовательно, решение x^* на некотором шаге алгоритма было отброшено вместе с множеством t по второму признаку. В этом случае можем написать

$$f(x^*) \geq H(t) \geq H_0,$$

что противоречит сделанному предположению и доказывает требуемое неравенство.

Приближенный алгоритм ветвей и границ с усиленной нижней границей отличается от основного алгоритма только тем, что на каждом шаге этого алгоритма при проверке множества t_l рекордное значение $f(x^0)$ целевой функции сравнивается не с величиной $H(t_l)$, а с величиной $(1+R_3)H(t_l)$ и множество t_l считается отброшенным по первому условию, если выполняется неравенство

$$(1 + R_3)H(t_l) \geq f(x_0),$$

где $R_3 \geq 0$ – некоторый параметр, называемый *коэффициентом усиления нижней границы*.

В результате работы данной модификации алгоритма ветвей и границ получается решение x^0 , для которого справедливо неравенство

$$f(x_0) \leq (1 + R_3)f(x^*).$$

Это означает, что рассмотренный приближенный алгоритм имеет априорную оценку точности равную $(1+R_3)$. В справедливости приведенного неравенства можно убедиться точно также, как и в справедливости рассмотренных выше аналогичных неравенств.

3.4 Задание подмножеств множества булевых векторов

Уже отмечалось, что для построения алгоритма ветвей и границ необходимо, в соответствии со спецификой исследуемой задачи, конкретизировать способ задания подмножеств решений и функцию ветвления. Можно считать, что интересующие нас задачи в качестве множества допустимых решений имеют множество булевых векторов. Поэтому в соответствии со спецификой множества булевых векторов, определим способ задания подмножеств этого множества и функцию ветвления так, чтобы по возможности более компактно записывать информацию о рассматриваемых в ходе работы алгоритма ветвей и границ подмножествах. Важную роль при этом играет понятие частичного вектора или частичного решения

Пусть исследуемая задача имеет m булевых переменных и пусть множеством допустимых решений задачи является множество B^m булевых векторов длины m . Вектор (y_1, \dots, y_m) , элементы которого принимают три значения: 0, 1, *, назовем *частичным решением (вектором)*. Для частичного решения (y_1, \dots, y_m) положим $I_0 = \{i \in I \mid y_i = 0\}$, $I_1 = \{i \in I \mid y_i = 1\}$, $I' = \{i \in I \mid y_i = *\}$. Таким образом, частичное решение делит переменные исследуемой задачи на две группы: переменные, значения которых фиксированы и свободные переменные. Булевый вектор (z_1, \dots, z_m) назовем *продолжением* частичного решения (y_1, \dots, y_m) , если $\{i \in I \mid z_i = 0\} \supset I_0$ и $\{i \in I \mid z_i = 1\} \supset I_1$. Частичное решение (y_1, \dots, y_m) определяет некоторое подмножество множества B^m . Таким подмножеством является совокупность всех продолжений этого частичного решения. Это подмножество будем обозначать $P(y_1, \dots, y_m)$. Понятно, что если у частичного решения $I' = I$, то $P(y_1, \dots, y_m)$ совпадает с множеством B^m всех булевых векторов. Если же $I' = \emptyset$, то $P(y_1, \dots, y_m)$ есть одноточечное множество $\{(y_1, \dots, y_m)\}$.

Нашей ближайшей целью будет показать, что функция ветвления может быть определена таким образом, что всякое рассматриваемое в ходе работы алгоритма ветвей и границ множество есть множество продолжений некоторого частичного решения.

Определение функции ветвления в рассматриваемом случае не вызывает трудностей. Пусть (y_1, \dots, y_m) — частичное решение для некоторого $I' \neq \emptyset$. Выберем, пока не конкретизируя по какому правилу, некоторый элемент $i_0 \in I'$, который будем называть *ведущим элементом функции ветвления*. Результатом применения функции ветвления к множеству $P(y_1, \dots, y_m)$ считаем пару множеств, задаваемых следующим частичным решением

$$(y_1, \dots, y_{i_0-1}, 0, y_{i_0+1}, \dots, y_m), (y_1, \dots, y_{i_0-1}, 1, y_{i_0+1}, \dots, y_m).$$

Таким образом, функция ветвления, применяемая к множеству продолжений частичного решения (y_1, \dots, y_m) с ведущим элементом $i_0 \in I'$ делит это множество на два

подмножества, каждое из которых также является множеством продолжений частичного решения. При этом, первое подмножество содержит все решения (z_1, \dots, z_m) исходного множества, для которых $z_{i_0} = 0$, а второе — все решения, для которых $z_{i_0} = 1$.

Отсюда следует, что на каждом шаге алгоритма ветвей и границ информация о разбиении множества неотброшенных решений может быть задана последовательностью частичных решений. При этом в качестве перспективного может быть множество, заданное любым частичным решением, что соответствует схеме одновременного ветвления.

Однако число частичных решений, задающих разбиение множества неотброшенных решений при одновременной схеме ветвления, может быть достаточно большим и сравнимым с числом всех решений. Поэтому рассмотрим более компактный способ задания разбиения. Этот способ предполагает одностороннюю схему ветвления, при которой в качестве перспективного подмножества выступает не произвольный элемент разбиения, а всегда, например, последний.

Покажем, что частичное решение (y_1, \dots, y_m) может быть использовано не только для задания множества $P(y_1, \dots, y_m)$ его продолжений, но и некоторой последовательности множеств, в которой каждый элемент есть множество продолжений некоторого частичного решения. Для этого рассмотрим так называемую компактную форму задания частичного решения. *Компактным заданием частичного решения* (y_1, \dots, y_m) , для которого $|I_0 \cup I_1| = q$, $0 \leq q \leq m$, назовем пару векторов (y'_1, \dots, y'_q) и (i_1, \dots, i_q) такую, что $\{i_1, \dots, i_q\} = I_0 \cup I_1$ и $y'_k = y_{i_k}$, $k = 1, \dots, q$. При этом булевый вектор (y'_1, \dots, y'_q) будем называть *компактным частичным решением длины q* , а вектор (i_1, \dots, i_q) — вектором номеров компактного частичного решения.

Пусть имеется компактное частичное решение (y_1, \dots, y_q) с вектором номеров (i_1, \dots, i_q) . Рассмотрим порождаемую данным решением следующую последовательность компактных частичных решений

$$(0), (y_1, 0), (y_1, y_2, 0), \dots, (y_1, \dots, y_{q-1}, 0), (y_1, \dots, y_{q-1}, 1)$$

с соответствующими векторами номеров

$$(i_1), (i_1, i_2), (i_1, i_2, i_3), \dots, (i_1, \dots, i_q), (i_1, \dots, i_q).$$

Далее рассмотрим последовательность множеств, каждый элемент в которой есть множество продолжений соответствующего компактного частичного решения

$$P(0), P(y_1, 0), P(y_1, y_2, 0), \dots, P(y_1, \dots, y_{q-1}, 0), P(y_1, \dots, y_{q-1}, 1).$$

Исключим из этой последовательности, во-первых, все множества $P(y_1, \dots, y_{p-1}, 0)$, $1 \leq p \leq q$, если $y_p = 0$, и, во-вторых, множество $P(y_1, \dots, y_{q-1}, 1)$, если $y_q = 0$. Полученную таким образом последовательность множеств считаем по определению *последовательностью, задаваемой компактным частичным решением* (y_1, \dots, y_q) с вектором номеров (i_1, \dots, i_q) .

Покажем, что если в алгоритме ветвей и границ используется определенная выше функция ветвления, а на каждом шаге алгоритма проверяется только последний элемент разбиения и при этом в качестве перспективного также используется послед-

ний элемент разбиения, то на каждом шаге алгоритма ветвей и границ информация о разбиении множества неотброшенных решений может быть задана частичным решением в компактной форме.

Действительно, если на первом шаге алгоритма множество решений, заданное пустым частичным решением с $I' = I$, не отбрасывается и, тем самым, алгоритм не заканчивает работу, то к этому множеству применяется функция ветвления с ведущим элементом i_0 . В результате получается следующее разбиение $P(0)$, $P(1)$, заданное компактным частичным решением (1) с вектором номеров (i_1) , где $i_1 = i_0$. Таким образом, на втором шаге алгоритма рассматривается разбиение, заданное компактным частичным решением.

Пусть на очередном шаге алгоритма ветвей и границ разбиение множества неотброшенных решений задается компактным частичным решением (y_1, \dots, y_q) с вектором номеров (i_1, \dots, i_q) , $1 \leq q \leq m$. Пусть p – наибольший номер i , $1 \leq i \leq q$, для которого $y_i = 1$. Возможны следующие три случая: либо $p = q$, либо $1 \leq p < q$, либо $p = 0$, когда номера i с указанным выше свойством не существует. Если $p = 0$, то рассматриваемое разбиение состоит из единственного элемента $P(y_1, \dots, y_q)$. Когда $p < q$ и $p = q$, то последние два элемента разбиения имеют соответственно вид:

$$P(y_1, \dots, y_{p-1}, 0), P(y_1, \dots, y_{q-1}, 0)$$

и

$$P(y_1, \dots, y_{q-1}, 0), P(y_1, \dots, y_{q-1}, 1).$$

Предположим, что в результате проверки последнего элемента рассматриваемого разбиения это множество решений должно быть отброшено. Если $p = 0$, то алгоритм заканчивает работу. Если $p < q$, то получаемое разбиение задается компактным частичным решением $(y_1, \dots, y_{p-1}, 0)$ с вектором номеров (i_1, \dots, i_p) , а если $p = q$, то компактным частичным решением $(y_1, \dots, y_{q-1}, 0)$ с вектором номеров (i_1, \dots, i_q) .

Пусть последний элемент рассматриваемого разбиения множество $P(y_1, \dots, y_q)$ отбросить не удастся и к нему применяется функция ветвления с ведущим элементом i_0 . В каждом из трех рассмотренных случаев последний элемент разбиения заменяется на пару множеств $P(y_1, \dots, y_q, 0)$, $P(y_1, \dots, y_q, 1)$. При этом непосредственно проверяется, что в любом из трех случаев вновь полученное разбиение задается компактным частичным решением $(y_1, \dots, y_q, 1)$ с вектором номеров $(i_1, \dots, i_q, i_{q+1})$, где $i_{q+1} = i_0$.

Из сказанного следует, что на каждом шаге алгоритма ветвей и границ при односторонней схеме ветвления информация о разбиении множества неотброшенных решений может быть задана частичным решением в компактной форме.

Отсюда ясно, что способ одностороннего ветвления будет полезным при глобальной проверке элементов разбиения в алгоритме с ограниченным числом элементов разбиения. Его полезность обусловлена тем, что при своей реализации он не требует дополнительных ресурсов памяти для хранения информации о разбиении глобально проверяемого множества, поскольку вся информация о таком разбиении может быть задана компактным частичным решением.

Таким образом, применительно к задаче минимизации функции, заданной на множестве булевых векторов, определены способы задания подмножеств решений и функция ветвления. Далее, строя алгоритмы ветвей и границ для задач FL , $MINF_0$, $MINP_0$, будем иметь в виду указанные способы. При этом останется только уточнить правило выбора ведущего элемента i_0 у функции ветвления.

3.5 Функции вычисления нижней границы и наилучшего решения для задачи размещения средств обслуживания

Помимо способа задания подмножеств решений и функции ветвления центральным моментом алгоритма ветвей и границ является способ вычисления нижней границы на подмножествах решений. При этом, в отличие от задания подмножеств решений, построение нижней границы предполагает конкретный вид целевой функции исследуемой задачи. Имея в виду в дальнейшем построение алгоритма ветвей и границ для задачи FL , рассмотрим способы вычисления нижней границы и функции выбора применительно к задаче FL . При этом данную задачу будем представлять как задачу минимизации функции

$$f_0(z_1, \dots, z_m) = \sum_{i \in I} f_i z_i + \sum_{j \in J} \min_{i|z_i=1} c_{ij}$$

на множестве булевых векторов. Эта задача есть задача $MINF_0$, в которой решения задаются не подмножествами, а булевыми векторами.

Согласно определенному выше способу задания подмножеств множества булевых векторов, нижняя граница H для рассматриваемой задачи должна быть определена для всякого частичного решения (y_1, \dots, y_m) , то есть для множества $P(y_1, \dots, y_m)$ продолжений любого частичного решения (y_1, \dots, y_m) .

Заметим, что способ построения нижней оценки для множества значений целевой функции задачи FL и, следовательно, рассматриваемой задачи определен ранее. Напомним, что такая оценка вычисляется по оценочной матрице W , порожденной тупиковым решением V для пары матриц (F_0, C) . Поэтому для частичного решения (y_1, \dots, y_m) , у которого $I_0 = \emptyset$, $I_1 = \emptyset$, интересующая нас функция $H(y_1, \dots, y_m)$ определена. Определим эту функцию для частичного решения (y_1, \dots, y_m) с любыми множествами I_0 , I_1 и I' , $I' \neq \emptyset$.

Для этого рассмотрим задачу минимизации исходной функции $f_0(z_1, \dots, z_m)$ на множестве $P(y_1, \dots, y_m)$, то есть задачу

$$\min \{f_0(z_1, \dots, z_m) \mid (z_1, \dots, z_m) \in P(y_1, \dots, y_m)\}$$

и задачу

$$\min \{f'_0(z_1, \dots, z_m) \mid (z_1, \dots, z_m) \in B^m\}$$

для функции

$$f'_0(z_1, \dots, z_m) = \sum_{i \in I_1} f_i + \sum_{i \in I} f'_i z_i + \sum_{j \in J} \min_{i|z_i=1} c'_{ij},$$

задаваемой парой матриц (F'_0, C') , где $F'_0 = (f'_i)$ – вектор-столбец длины m , а $C' = (c'_{ij})$ – матрица размера $m \times n$. Элементы указанных матриц определяются следующим образом:

$$f'_i = \begin{cases} 0, & \text{если } i \in I_1, \\ f_i, & \text{иначе;} \end{cases}$$

$$c'_{ij} = \begin{cases} \max_{i \in I} c_{ij}, & \text{если } i \in I_0, \\ c_{ij}, & \text{иначе.} \end{cases}$$

Покажем, что существует оптимальное решение второй задачи, которое будет оптимальным решением первой, а оптимальные значения целевых функций рассматриваемых задач равны. Действительно, пусть (z_1^*, \dots, z_m^*) – оптимальное решение второй задачи. Заметим, что в силу оптимальности этого решения, имеем $z_i^* = 0$, если $i \in I_0$. Кроме того, можно считать, что $z_i^* = 1$, если $i \in I_1$. Действительно, если $z_{i_0}^* = 0$ для некоторого $i_0 \in I_1$, то решение (z'_1, \dots, z'_m) , которое отличается от первого только тем, что $z'_{i_0} = 1$, также будет оптимальным. Таким образом, оптимальное решение (z_1^*, \dots, z_m^*) является допустимым решением первой задачи и для этого решения справедливы равенства

$$\sum_{i \in I_1} f_i + \sum_{i \in I} f'_i z_i^* = \sum_{i \in I_1} f_i + \sum_{i \in I'} f_i z_i^* = \sum_{i \in I} f_i z_i^*.$$

Чтобы убедиться в том, что это решение является оптимальным решением первой задачи, рассмотрим некоторое оптимальное решение (z_1, \dots, z_m) первой задачи. Это решение будет, очевидно, допустимым решением второй задачи, а значения целевых функций обеих задач на этом решении будут одинаковыми. Отсюда, в силу леммы 1.1, следует, что исходное оптимальное решение (z_1^*, \dots, z_m^*) второй задачи есть оптимальное решение первой задачи, а оптимальные значения целевых функций равны.

Отсюда получаем, что если W' – оценочная матрица, порожденная тупиковым решением V' для пары матриц (F'_0, C') , то для всякого продолжения (z_1, \dots, z_m) частичного решения (y_1, \dots, y_m) можем написать

$$f_0(z_1, \dots, z_m) \geq f_0(z_1^*, \dots, z_m^*) = f'_0(z_1^*, \dots, z_m^*) \geq \sum_{i \in I_1} f_i + H(W').$$

Следовательно,

$$H(y_1, \dots, y_m) = \sum_{i \in I_1} f_i + H(W')$$

есть искомое значение нижней границы.

Таким образом, для произвольного частичного решения вычисление значения нижней границы на множестве продолжений этого частичного решения также сводится к построению тупикового решения, но не для исходной пары (F_0, C) , а для пары (F'_0, C') , построенной по паре (F_0, C) указанным выше образом.

Поскольку при вычислении значения нижней границы $H(y_1, \dots, y_m)$ на множестве $P(y_1, \dots, y_m)$ строится, как сказанного выше, тупиковое решение V' для пары матриц (F'_0, C') , то свойствами этого тупикового решения и, в частности, свойствами блокирующего множества $I'_0(V')$ можно воспользоваться, чтобы найти наилучшее решение этого множества и тем самым определить значение на множестве $P(y_1, \dots, y_m)$ функции выбора $z(y_1, \dots, y_m)$ наилучшего решения.

Напомним, что согласно лемме 2.1, если $I'_0(V') \cap I' = \emptyset$, то наилучшим решением множества $P(y_1, \dots, y_m)$ является вектор (z_1, \dots, z_m) такой, что

$$z_i = \begin{cases} y_i, & \text{если } i \in I_0 \cup I_1, \\ 0, & \text{если } i \in I'; \end{cases}$$

а если $I'_0(V') \cap I' = \{i_0\}$, то наилучшим будет вектор (z_1, \dots, z_m) , где

$$z_i = \begin{cases} y_i, & \text{если } i \in I_0 \cup I_1, \\ 1, & \text{если } i = i_0, \\ 0, & \text{если } i \in I' \setminus \{i_0\}; \end{cases}$$

Таким образом, функцию выбора наилучшего решения $z(y_1, \dots, y_m)$ будем считать определенной на таких частичных решениях, для которых соответствующая тупиковая матрица V' имеет блокирующее множество $I'_0(V')$ такое, что множество $I'_0(V') \cap I'$ содержит не более одного элемента.

При определении ведущего элемента функции ветвления в случае задачи FL также используем свойства блокирующего множества $I'_0(V')$. Напомним, что элементы этого множества рассматриваются как вероятные номера единичных компонент наилучшего решения. Поэтому при применении к множеству $P(y_1, \dots, y_m)$ функции ветвления ведущий элемент $i_0(y_1, \dots, y_m)$ естественно выбирать из множества $I'_0(V') \cap I'$. При этом можно руководствоваться несколькими правилами.

В качестве основного и наиболее простого используем правило выбора в качестве $i_0(y_1, \dots, y_m)$ наименьшего элемента множества $I'_0(V') \cap I'$.

Два других правила близки к основному. Первое из них состоит в выборе такого элемента $I'_0(V') \cap I'$ для которого в ходе построения тупикового решения $V' = (v'_{ij})$ первым реализовывалось равенство $f_i = \sum_{j \in J} v'_{ij}$. По второму правилу выбирается такой элемент $i \in I'_0(V') \cap I'$, который блокирует наибольшее число столбцов матрицы V' , то есть такой элемент, для которого мощность множества $J(i) = \{j \in J \mid i \in I_j^+(V')\}$ наибольшая.

3.6 Алгоритмы ветвей и границ для задачи размещения средств обслуживания

Уже неоднократно подчеркивалось, что для построения конкретного алгоритма ветвей и границ, предназначенного для решения определенной задачи, необходимо, учитывая специфику этой задачи, конкретизировать основные элементы общей схемы ветвей и границ.

В соответствии с этим, выше для задачи FL , представленной как задача минимизации функции от булевых переменных на множестве булевых векторов, конкретизированы основные элементы вычислительной схемы ветвей и границ. В частности, указано, как задаются рассматриваемые в ходе работы алгоритма множества решений, как устроена функция ветвления, как вычисляется нижняя граница и в каких случаях считается определенной функция выбора наилучшего решения.

Суммируя сделанные выше «заготовки», дадим описание двух «базовых» алгоритмов ветвей и границ для решения задачи FL . Первый алгоритм, дающий решение произвольной, наперед заданной точности, представляет собой алгоритм с ограниченным числом элементов разбиения, в котором при глобальной проверке элементов разбиения используется схема одностороннего ветвления. Второй алгоритм – приближенный, отличающийся от первого лишь тем, что для глобальной проверки применяется гриди алгоритм, использующий блокирующее множество.

Работа каждого такого алгоритма регулируется заданием значений определенных выше параметров R_2 и R_3 . Напомним смысл каждого из них.

Целочисленный параметр R_2 , $R_2 \geq 0$, ограничивает число одновременно рассматриваемых элементов разбиения множества неотброшенных решений или, другими словами, число одновременно рассматриваемых частичных решений. Если $R_2 = 0$, то реализуется односторонняя схема ветвления и на каждом шаге рассматривается только одно компактное частичное решение.

Параметр R_3 , $R_3 \geq 0$, называемый коэффициентом усиления нижней границы, реализует величину «искусственного» увеличения нижней границы. Если $R_3 = 0$, то увеличения нижней границы не производится.

Точный алгоритм ветвей и границ для решения задачи FL включает конечное число однотипных шагов, на каждом из которых имеется рекордное решение (z_1^0, \dots, z_m^0) и последовательность частичных решений

$$(y_1^1, \dots, y_m^1), (y_1^2, \dots, y_m^2), \dots, (y_1^L, \dots, y_m^L),$$

где $L \leq R_2 + 1$, задающих разбиение множества неотброшенных решений. Если $L = R_2 + 1$, то последнее частичное решение является компактным решением (y_1^L, \dots, y_q^L) , для которого задан также вектор номеров (i_1, \dots, i_q) .

В начале первого шага имеется пустое частичное решение (y_1^1, \dots, y_m^1) , для которого $I' = I$. Шаг начинается с построения для исходной пары (F_0, C) тупикового решения V , вычисления нижней границы $H(y_1^1, \dots, y_m^1)$ и получения блокирующего множест-

ва $I_0(V)$. Если $|I_0(V)| = 1$, то формируется соответствующее рекордное решение (z_1^0, \dots, z_m^0) и алгоритм заканчивает работу. Если $|I_0(V)| > 1$, то посредством гриди алгоритма, использующего блокирующее множество, строится приближенное решение (z_1^0, \dots, z_m^0) , которое принимается в качестве первого рекордного решения. Если для полученного решения выполняется неравенство

$$H(y_1^1, \dots, y_m^1) \geq f(z_1^0, \dots, z_m^0),$$

то работа алгоритма заканчивается, в противном случае в множестве $I_0(V)$ выбирается ведущий элемент i_0 и к множеству $P(y_1^1, \dots, y_m^1)$ применяется функция ветвления.

Если $R_2 \neq 0$, то формируется последовательность из двух частичных решений

$$(y_1^1, \dots, y_{i_0-1}^1, 0, y_{i_0+1}^1, \dots, y_m^1), (y_1^1, \dots, y_{i_0-1}^1, 1, y_{i_0+1}^1, \dots, y_m^1),$$

определяющих новое разбиение множества неотброшенных решений, и начинается второй шаг алгоритма.

Если $R_2 = 0$, то строится компактное частичное решение (1) с вектором номеров (i_1) , где $i_1 = i_0$ и начинается второй шаг.

Пусть к началу очередного шага имеется текущий рекорд (z_1^0, \dots, z_m^0) и последовательность частичных решений

$$(y_1^1, \dots, y_m^1), \dots, (y_1^L, \dots, y_m^L), 1 \leq L \leq R_2 + 1.$$

Считается, что для всякого частичного решения (y_1^l, \dots, y_m^l) , $l \leq L'$, известно значение нижней границы $H(y_1^l, \dots, y_m^l)$, а также полученный в результате вычисления нижней границы ведущий элемент $i_0(y_1^l, \dots, y_m^l)$ функции ветвления. Отметим, что $L' \geq L-2$, поскольку значения нижней границы могут быть еще не известны только для двух или одного частичного решения, которые получены на предыдущем шаге в результате соответственно одновременного или одностороннего ветвления.

Если $L \geq R_2$, то шаг начинается с построения для каждого частичного решения (y_1^l, \dots, y_m^l) , $l = L' + 1, \dots, L$ соответствующей пары матриц (F_0', C') и тупикового решения V' , а также вычисления значения нижней границы $H(y_1^l, \dots, y_m^l)$, и ведущего элемента $i_0(y_1^l, \dots, y_m^l)$. Одновременно проверяется справедливость неравенства $|I_0'(V') \cap I'| \leq 1$. Если неравенство выполняется, то строится наилучшее решение $z(y_1^l, \dots, y_m^l)$. Если при этом найденное решение $z(y_1^l, \dots, y_m^l)$ лучше рекорда, то производится замена рекорда на это решение. Далее для всех частичных решений (y_1^l, \dots, y_m^l) , $l \leq L$, проверяется справедливость неравенства

$$H(y_1^l, \dots, y_m^l) \geq f_0(z_1^0, \dots, z_m^0)$$

и отбрасываются те частичные решения, для которых это неравенство выполняется. В результате, после перенумерации не отброшенных частичных решений получается новая последовательность

$$(y_1^1, \dots, y_m^1), \dots, (y_1^L, \dots, y_m^L), 1 \leq L \leq R_2,$$

элементы которой считаем упорядоченными по убыванию значения нижней границы. Если $L = 0$, то алгоритм заканчивает работу, в противном случае полагается $i_0 = i_0(y_1^L, \dots, y_m^L)$ и частичное решение (y_1^L, \dots, y_m^L) заменяется на два новых частичных решения (y_1^L, \dots, y_m^L) и $(y_1^{L+1}, \dots, y_m^{L+1})$, в которых компонента с номером i_0 равняется соответственно 0 и 1. Далее, если $L < R_2$, начинается следующий шаг. Если же $L = R_2$, то по частичному решению $(y_1^{L+1}, \dots, y_m^{L+1})$ строится компактное частичное решение $(y_1^{L+1}, \dots, y_r^{L+1})$ и вектор номеров (i_1, \dots, i_r) . После этого начинается следующий шаг.

Если в начале рассматриваемого шага $L = R_2 + 1$, то дальнейший шаг начинается с построения для компактного частичного решения (y_1^L, \dots, y_q^L) соответствующей пары матриц (F'_0, C') и тупикового решения V' , а также вычисления значения нижней границы $H(y_1^L, \dots, y_q^L)$ и ведущего элемента $i_0(y_1^L, \dots, y_q^L)$. Одновременно проверяется справедливость неравенства $|I'_0(V') \cap I'| \leq 1$, то есть выясняется, определена ли для исследуемого частичного решения функция выбора наилучшего решения.

Пусть выполняется одно из двух: либо $H(y_1^L, \dots, y_q^L) \geq f_0(z_1^0, \dots, z_m^0)$, либо $|I'_0(V') \cap I'| \leq 1$ и наилучшее решение $z(y_1^L, \dots, y_q^L)$ определено. Если при этом найденное решение $z(y_1^L, \dots, y_q^L)$ лучше рекорда, то производится замена рекорда на это решение. Далее в любом случае определяется наибольший номер p , $r < p \leq q$, для которого $y_p^L = 1$. Напомним, что r – длина начального компактного частичного решения при переходе от одновременного к одностороннему ветвлению. Если номера p не существует, то рассматриваемое компактное частичное решение отбрасывается. Если при этом $R_1 = 0$, то алгоритм заканчивает работу, в противном случае начинается следующий шаг. Если же номер p с указанными свойствами найден, то строится новое компактное частичное решение $(y_1^L, \dots, y_{p-1}^L, y_p^L)$, где $y_p^L = 0$, и начинается следующий шаг.

Пусть не выполняется ни одно из указанных выше условий, то есть пусть $H(y_1^L, \dots, y_q^L) < f_0(z_1^0, \dots, z_m^0)$ и $|I'_0(V') \cap I'| > 1$. Тогда строится новое компактное частичное решение $(y_1^L, \dots, y_q^L, y_{q+1}^L)$, где $y_{q+1}^L = 1$, с вектором номеров $(i_1, \dots, i_q, i_{q+1})$, где $i_{q+1} = i_0(y_1^L, \dots, y_q^L)$, и начинается следующий шаг.

В результате работы данного алгоритма получается решение (z_1^0, \dots, z_m^0) , для которого справедливо неравенство

$$f(z_0^1, \dots, z_m^1) \leq (1 + R_3) f(z_1^*, \dots, z_m^*),$$

где (z_1^*, \dots, z_m^*) – оптимальное решение. Понятно, что если используется нулевое значение параметра R_3 , то полученное решение (z_1^0, \dots, z_m^0) является оптимальным.

Приближенный алгоритм ветвей и границ для решения задачи FL работает при $F_2 \neq 0$ и в основной своей части ничем не отличается от первого алгоритма.

На каждом шаге алгоритма имеется рекордное решение (z_1^0, \dots, z_m^0) и последовательность частичных решений

$$(y_1^1, \dots, y_m^1), (y_1^2, \dots, y_m^2), \dots, (y_1^L, \dots, y_m^L),$$

где $L \leq R_2 + 1$. Кроме того, имеется величина H_0 равная наименьшему значению нижней границы $H(y_1^L, \dots, y_m^L)$ по всем рассмотренным на предыдущих шагах частичным решениям (y_1^L, \dots, y_m^L) , для которых $L = R_2 + 1$.

Действия на каждом шаге данного алгоритма отличаются от действий предыдущего алгоритма только, когда в начале шага выполняется равенство $L = R_2 + 1$. В этом случае по частичному решению (y_1^L, \dots, y_m^L) строится пара матриц (F'_0, C') , тупиковое решение V' и нижняя граница $H(y_1^L, \dots, y_m^L)$. Если $H(y_1^L, \dots, y_m^L) < H_0$, то полагается $H_0 = H(y_1^L, \dots, y_m^L)$. Далее гриди алгоритмом с использованием блокирующего множества $I'_0(V')$ решается задача $MINF_0$ с парой матриц (F'_0, C') . Если для полученного решения (z_1, \dots, z_m) выполняется неравенство $f(z_1, \dots, z_m) < f(z_1^0, \dots, z_m^0)$, то производится замена рекорда. После этого частичное решение (y_1^L, \dots, y_m^L) отбрасывается и начинается следующий шаг алгоритма.

В результате работы данного приближенного алгоритма получается решение (z_1^0, \dots, z_m^0) с апостериорной оценкой точности равной

$$\max \{ (1 + R_3); f(z_1^0, \dots, z_m^0) / H_0 \}.$$

В заключение отметим, что приведенные алгоритмы ветвей и границ для решения задачи FL , которые названы базовыми, допускают различные модификации и усовершенствования по многим направлениям. Изменениям могут быть подвергнуты различные элементы вычислительной схемы, в частности, процедуры глобальной проверки подмножеств, правила выбора перспективных подмножеств, а также некоторые процедуры алгоритма вычисления нижней границы. Полезность той или иной модификации базисного алгоритма невозможно оценить в общем случае. От ее целесообразности применительно к тому или иному классу задач FL можно судить по результатам вычислительных экспериментов. Некоторые такие результаты, дающие общее представление о работоспособности рассмотренных алгоритмов, приводятся ниже.