А. А. Агеев, Э. Х. Гимади, О. Ю. Цидулко, А. А. Штепа, Задача размещения с ограничениями на объемы производства предприятий на графах древесного вида, *Тр. ИММ УрО РАН*, 2022, том 28, номер 2, 24–44

DOI: 10.21538/0134-4889-2022-28-2-24-44

ТРУДЫ ИНСТИТУТА МАТЕМАТИКИ И МЕХАНИКИ УрО РАН

Том 28 № 2 2022

УДК 519.8

ЗАДАЧА РАЗМЕЩЕНИЯ С ОГРАНИЧЕНИЯМИ НА ОБЪЕМЫ ПРОИЗВОДСТВА ПРЕДПРИЯТИЙ НА ГРАФАХ ДРЕВЕСНОГО ВИДА[1]

А. А. Агеев, Э. Х. Гимади, О. Ю. Цидулко, А. А. Штепа

В данной работе рассматриваются сетевая задача размещения с ограничениями на объемы производства предприятий (CFLP) и ее однородный вариант (UCFLP), в котором объемы производства всех предприятий одинаковые. Мы показываем, что UCFLP на звезде решается за линейное время, если в одной вершине графа не могут одновременно находиться предприятие и клиент, и $\mathcal{NP}$-трудна, если в каждой вершине могут находиться и предприятие, и клиент. Известно, что задача UCFLP на цепи полиномиально разрешима, мы улучшаем известную схему динамического программирования для этой задачи до оценки $\mathcal{O}(m^2n^2)$, где m — количество предприятий, n — количество клиентов. Для CFLP на цепи мы предлагаем псевдополиномиальный алгоритм ее решения на основе подхода из работы Мирчандани и др. (1996) с улучшенной временной сложностью $\mathcal{O}(mB)$, где B — суммарный спрос клиентов.

Ключевые слова: задача размещения с ограничениями на объемы производства, однородная задача размещения с ограничениями на объемы производства, NP-трудная задача, звезда, цепь, полиномиальный алгоритм, псевдополиномиальный алгоритм, динамическое программирование.

A. A. Ageev, E. Kh. Gimadi, O. Yu. Tsidulko, A. A. Shtepa. Capacitated Facility Location Problem on tree-like graphs.

We consider the network Capacitated Facility Location Problem (CFLP) and its special case — the Uniform Capacitated Facility Location Problem (UCFLP), where all facilities have the same capacity. We show that the UCFLP on a star graph is linear-time solvable if every vertex of the star can be either a facility or a client but not both. We further prove that the UCFLP on a star graph is $\mathcal{NP}$-hard if the facilities and clients can be located at each vertex of the graph. The UCFLP on a path graph is known to be polynomially solvable. We give a version of the known dynamic programming algorithm for this problem with the improved time complexity $\mathcal{O}(m^2n^2)$, where m is the number of facilities and n is the number of clients. For the CFLP on a path graph we propose a pseudo-polynomial time algorithm based on the work of Mirchandani et al. (1996) with improved time complexity $\mathcal{O}(mB)$, where B is the total demand of the clients.

Keywords: Capacitated Facility Location Problem, Uniform Capacitated Facility Location Problem, NP-hard problem, star graph, path graph, polynomial time algorithm, pseudo-polynomial time algorithm, dynamic programming.

MSC: 90-02, 90B80

DOI: 10.21538/0134-4889-2022-28-2-24-44

1. Введение

Простейшая задача размещения или задача размещения с неограниченными объемами производства (Uncapacitated Facility Location Problem, UFLP) — это классическая оптимизационная задача для определения оптимального расположения предприятий или складов. Она имеет большое количество применений в логистике, телекоммуникациях, здравоохранении, планировании устойчивых систем, кластеризации, компьютерном зрении и т. д. Для детального обзора рекомендуем заинтересовавшемуся читателю работы [5; 18; 20; 27; 30; 31].

В задаче UFLP дано множество из n клиентов, каждый из которых имеет спрос в производимом продукте, и множество из m предприятий, производящих продукт. Заданы стоимости открытия предприятий и транспортные расходы по доставке единицы продукта от каждого

[1]Работа выполнена в рамках государственного задания ИМ СО РАН (проект FWNF-2022-0019) и при финансовой поддержке РФФИ (проект 20-31-90091).

предприятия к каждому клиенту. Задача состоит в том, чтобы определить, какие предприятия нужно открыть, чтобы суммарные затраты по открытию предприятий и транспортировке продукта были минимальными. Известно, что эта задача $\mathcal{NP}$-трудна в сильном смысле, так к ней сводится задача о покрытии множества [17; 21].

В литературе большое внимание уделено сетевой задаче UFLP, в которой клиенты и предприятия находятся в вершинах реберно-взвешенного графа, и транспортные расходы определяются согласно кратчайшим длинам путей доставки. В случае полных графов эта задача совпадает с метрической. Известно, что сетевая задача UFLP остается $\mathcal{NP}$-трудной в сильном смысле даже на планарной решетке [12]. Тем не менее на дереве UFLP полиномиально разрешима: Трубин [10] первым предложил для нее алгоритм со временем работы $\mathcal{O}(n^3)$ (здесь $m = n$), затем Гимади [8] был разработан подход, использующий технику связных областей обслуживания, с трудоемкостью $\mathcal{O}(mn)$, позже Шах и Фарах-Колтон [31] предложили алгоритм со временем работы $\mathcal{O}(n \log n)$ $(m = n)$. Для задачи UFLP на цепи Хассин и Тамир [26] предложили линейный алгоритм решения. В работе Агеева [1] было показано, что задача на внешнепланарных графах решается за время $\mathcal{O}(mn^3)$, позже в [9] был построен алгоритм со временем работы $\mathcal{O}(mn^{2.5})$. Для задачи UFLP на последовательно-параллельных графах, которые включают в себя внешнепланарные, были предложены алгоритмы с трудоемкостями $\mathcal{O}(n^4)$ в [25] и $\mathcal{O}(m^3n)$ в [2]. Следует отметить, что второй алгоритм построен для более широкого класса задач и использует только существование оптимального решения со связными областями обслуживания. Для UFLP на частичном k-дереве известны полиномиальные алгоритмы со временем работы $\mathcal{O}(m^{k+1}n)$ [11] и $\mathcal{O}(n^{k+2})$ $(m = n)$ [24].

Естественное обобщение классической UFLP — это задача размещения с ограничениями на объемы производства (Capacitated Facility Location Problem, CFLP), в которой каждое предприятие i имеет объем производства a_i, что является ограничением сверху на количество продукта, которое может произвести данное предприятие. Известно, что метрическая задача CFLP с единичными спросами на полном графе принадлежит классу $\mathcal{APX}$ [29] с лучшей текущей оценкой точности приближенного алгоритма, равной 5 [16].

Если спросы клиентов неединичные, можно рассматривать постановку задачи CFLP с делимыми спросами, где спрос клиента может быть обслужен несколькими предприятиями совместно, и постановку с неделимыми спросами, где спрос клиента должен быть обслужен только одним предприятием. Задача CFLP с неделимыми спросами является $\mathcal{NP}$-трудной в сильном смысле даже на графе с одной вершиной, если в ней могут находиться несколько клиентов и предприятий, и даже в случае одинаковых объемов производства предприятий, поскольку к ней сводится задача упаковки в контейнеры. Задача CFLP с делимыми спросами $\mathcal{NP}$-трудна на графе с одной вершиной, поскольку к ней сводится задача о рюкзаке, однако если объемы производства предприятий одинаковы, известно, что задача полиномиально разрешима на цепи [13]. Далее всюду под задачей CFLP мы будем понимать вариант задачи с делимыми спросами.

На многих графах древесного вида задача CFLP с делимыми спросами может быть решена за псевдополиномиальное время. Так, в работе [28] Мирчандани и др. предложили псевдополиномиальный алгоритм решения этой задачи на цепи. При этом задача CFLP легко сводится к задаче без ограничений на объемы производства предприятий, но с ограничениями на пропускные способности ребер: из каждой вершины v_i, в которой находится предприятие с объемом производства a_i, можно перенести предприятие в новую вершину, связанную с v_i ребром нулевой стоимости с пропускной способностью a_i. Для задачи размещения с ограничениями на пропускные способности ребер и делимыми спросами существуют псевдополиномиальные алгоритмы в случаях, когда граф является деревом [6], частичным 2-деревом [7], частичным k-деревом [23] для любого фиксированного k или, что эквивалентно, в графах с древовидной шириной, ограниченной k. Таким образом, на этих типах графов задача CFLP может быть также решена за псевдополиномиальное время.

Если объемы производства предприятий одинаковы, то такая задача называется однород-

ной задачей размещения с ограничениями на объемы производства предприятий (Uniform Capacitated Facility Location Problem, UCFLP). Для метрического случая этой задачи с единичными спросами на данный момент лучшим приближенным алгоритмом является алгоритм локального поиска из работы [15] с оценкой точности, равной 3, а с неединичными спросами эта задача в плане построения эффективных приближенных алгоритмов, насколько нам известно, не изучалась. Для задачи UCFLP на цепи в 2004 году Агеев в работе [13] предложил первый полиномиальный алгоритм со временем работы $\mathcal{O}(m^5n^2 + m^3n^3)$. Позже временная сложность этого алгоритма была улучшена до $\mathcal{O}(m^4n^2)$ в работе [3] и до $\mathcal{O}(m^3n^2)$ в работе [22].

В данной работе мы исследуем сетевую задачу CFLP и ее однородный вариант UCFLP на простейших частных случаях дерева — звезде и цепи. Отметим, что задача CFLP на цепи имеет несколько практических приложений, например, размещение понижающих трансформаторов высоковольтной линии электропередач в сельской местности, создание областей отдыха вдоль скоростного шоссе, составление расписаний поставок от ритейлера к домохозяйствам, расположенным по соседству [28]. Мы уточняем сложностной статус задачи UCFLP и доказываем, что уже на звезде задача становится $\mathcal{NP}$-трудной, если в каждой вершине графа могут одновременно размещаться и предприятие, и клиент. С другой стороны, мы показываем, что задача UCFLP, в которой в каждой вершине может находиться либо предприятие, либо клиент, на звезде решается за линейное время, но $\mathcal{NP}$-трудна на дереве. Для известного полиномиально разрешимого случая задачи UCFLP на цепи мы рассматриваем алгоритм динамического программирования из работ [3; 13] и уменьшаем время его работы до $\mathcal{O}(m^2n^2)$, что является лучшим возможным временем работы в рамках текущей схемы динамического программирования [3;13]. Для неоднородной задачи CFLP на цепи мы предлагаем модификацию известного псевдополиномиального алгоритма Мирчандани и др. [28], улучшая таким образом время работы алгоритма с $\mathcal{O}\big(mB\min\{a_{\max}, B\}\big)$ до $\mathcal{O}(mB)$, где $B = \sum_j b_j$ — это суммарный спрос, $a_{\max}$ — максимальный объем производства предприятия.

Статья организована следующим образом. В разд. 2 мы приводим общую постановку задачи, предварительные замечания и определения. Раздел 3 посвящен задаче UCFLP на звезде: в подразд. 3.1 приведено доказательство $\mathcal{NP}$-трудности задачи UCFLP на звезде, если в вершине звезды могут одновременно находиться предприятие и клиент; в подразд. 3.2 предложен жадный точный линейный алгоритм решения UCFLP на звезде для случая, когда в каждой вершине звезды находится либо предприятие, либо клиент. В разд. 4 приводится алгоритм динамического программирования, решающий задачу UCFLP на цепи за время $\mathcal{O}(m^2n^2)$. В разд. 5 показано, как решить задачу CFLP на цепи за время $\mathcal{O}(mB)$.

2. Постановка задачи и базовые обозначения

В сетевой задаче CFLP дан граф $G = (V, E)$, где V — множество вершин графа, а E — множество ребер. Заданы множество предприятий $I = \{1, \ldots, m\}$ и мультимножество вершин графа $\mathcal{F} = \{v_1, \ldots, v_m\} \subseteq V$, указывающее места, где эти предприятия могут быть открыты, а также мультимножество клиентов $J = \{1, \ldots, n\}$ с множеством вершин, где они размещены $\mathcal{C} = \{u_1, \ldots, u_n\} \subseteq V$. Каждое предприятие $i \in I$ имеет объем производства $a_i \in \mathbb{N} \cup \{0\}$ и неотрицательную стоимость открытия f_i. Каждый клиент $j \in J$ имеет спрос $b_j \in \mathbb{N} \cup \{0\}$. Для каждого ребра $e \in E$ известна стоимость c_e транспортировки единицы продукта по этому ребру, а стоимость транспортировки единицы продукта из i-го предприятия j-му клиенту определяется как $g_{ij} = \sum_{e \in P_{ij}} c_e$, где P_{ij} — это кратчайший путь в графе между предприятием i и клиентом j. В задаче требуется открыть подмножество предприятий $S \subseteq I$ и обслужить весь спрос клиентов так, чтобы суммарные затраты на открытие предприятий и транспортировку продукта клиентам были минимальными. Без ограничения общности будем считать, что $\sum_{j=1}^{n} b_j \leq \sum_{i=1}^{m} a_i$, иначе задача, очевидно, не имеет допустимых решений.

В терминах целочисленного линейного программирования сетевая задача CFLP может

быть сформулирована следующим образом:

$$\min\Big(\sum_{i\in S} f_i + \sum_{i\in S}\sum_{j\in J} g_{ij}x_{ij}\Big) \tag{2.1}$$

при ограничениях

$$\sum_{i\in S} x_{ij} = b_j,\ j \in J, \tag{2.2}$$

$$\sum_{j\in J} x_{ij} \le a_i,\ i \in I, \tag{2.3}$$

$$x_{ij} \in \mathbb{N}\cup\{0\},\ i \in I,\ j \in J, \tag{2.4}$$

где x_{ij} равно объему спроса клиента j, который обслуживается из предприятия i. Условия (2.2) гарантируют, что спрос каждого клиента удовлетворен, в то время как уравнения (2.3) говорят о том, что объем производства предприятия i не превышен. Наконец, соотношения (2.4) означают, что спрос может делиться только на целочисленные части. Допустимое решение задачи со множеством открытых предприятий S и планом перевозок $x = (x_{ij})$ будем обозначать через (S, x).

О п р е д е л е н и е 1. Пусть (S, x) — допустимое решение задачи (2.1)–(2.4). Будем говорить, что i-е предприятие *обслуживает* j-го клиента, если $x_{ij} > 0$. Также будем говорить, что i-е предприятие *насыщенное*, если $\sum_{j=1}^{n} x_{ij} = a$, и *ненасыщенное*, если $0 < \sum_{j=1}^{n} x_{ij} < a$. Отметим, что насыщенные и ненасыщенные предприятия обязательно открыты.

З а м е ч а н и е 1. Для сетевой задачи CFLP множество открытых предприятий S полностью определяет стоимость решения, поскольку, зная S, оптимальный план перевозок $x = (x_{ij})$ можно найти за полиномиальное время, например, решив соответствующую транспортную задачу.

Задача с условием $a_i = a$ для любого $i \in I$ называется однородной сетевой задачей размещения с ограничениями на объемы производства предприятий. Эту задачу будем обозначать как UCFLP.

3. Задача UCFLP на звезде

Этот раздел посвящен задаче UCFLP, в которой входной граф является звездой, т. е. частным случаем дерева, где одна (центральная) вершина может иметь любую степень, а все остальные — висячие (имеют степень 1). Мы покажем, что задача UCFLP на звезде $\mathcal{NP}$-трудна в случае, когда в каждой вершине входного графа могут одновременно находиться предприятие и клиент, а в случае, когда в каждой вершине графа находится либо предприятие, либо клиент, $\mathcal{NP}$-трудна на дереве, однако на звезде решается за линейное время.

3.1. Сложностной статус UCFLP на звезде

Теорема 1. *Задача UCFLP, в которой в каждой вершине могут одновременно находиться клиент и предприятие, $\mathcal{NP}$-трудна даже в случае, когда входной граф является звездой.*

Д о к а з а т е л ь с т в о. Покажем, что соответствующая задача распознавания UCFLP на звезде $\mathcal{NP}$-полна, для чего сведем к ней известную $\mathcal{NP}$-полную задачу о рюкзаке [21]. В задаче о рюкзаке дано множество элементов $N = \{1, \dots, n\}$, известны их размеры $f_1, \dots, f_n \in \mathbb{N}\cup\{0\}$, стоимости $a_1, \dots, a_n \in \mathbb{N}\cup\{0\}$, заданы величины $F, A \in \mathbb{N}\cup\{0\}$, и спрашивается, существует ли подмножество $S \subseteq N = \{1, \dots, n\}$ такое, что

$$\sum_{i\in S} f_i \le F \quad \text{и} \quad \sum_{i\in S} a_i \ge A. \tag{3.1}$$

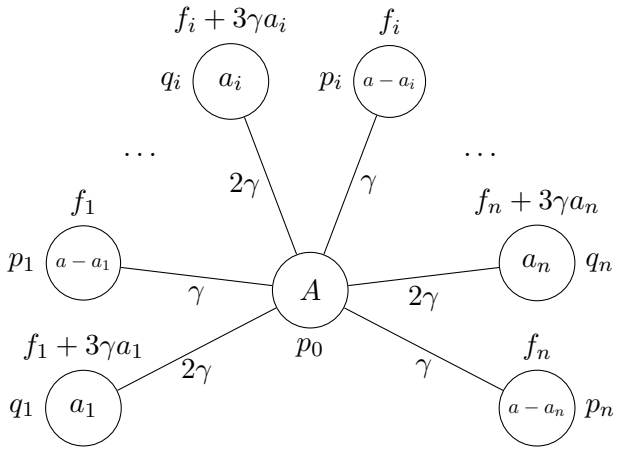

Рис. 1. Внутри вершин указан их клиентский спрос, над вершинами — стоимость открытия предприятия. Объемы производства всех предприятий равны a. У каждого ребра указана стоимость транспортировки единицы продукта.

Без ограничения общности можно считать, что $F < \sum_{i=1}^{n} f_i$.

Построим пример задачи распознавания UCFLP на звезде следующим образом (см. рис. 1). Положим объемы производства всех предприятий равными $a = \max_{i \in N} a_i + 1$. Положим $\gamma = 2 \cdot \sum_{i=1}^{n} f_i$. Введем вершину p_0, в которой находится клиент со спросом A, а стоимость открытия предприятия в p_0 равна $4\gamma \cdot (A + \sum_{i=1}^{n} a_i)$. Для каждого элемента $i \in N$ введем вершины p_i и q_i, соединим их ребрами с p_0. Стоимость транспортировки единицы продукта по ребру $\{p_0, p_i\}$ положим равной γ, по ребру $\{p_0, q_i\}$ — равной 2γ. В каждой вершине p_i, $i \in N$, находится клиент со спросом $a - a_i$, и может быть открыто предприятие со стоимостью f_i. В каждой вершине q_i, $i \in N$, находится клиент со спросом a_i, и может быть открыто предприятие стоимости $f_i + 3\gamma a_i$. Ограничение на стоимость допустимого решения положим равным

$$F' = F + \sum_{i=1}^{n} f_i + 3\gamma \cdot \sum_{i=1}^{n} a_i + \gamma \cdot A.$$

Покажем, что в построенном примере задачи UCFLP на звезде есть решение стоимости не более F' тогда и только тогда, когда в задаче о рюкзаке есть решение, удовлетворяющее (3.1).

($\Leftarrow$) Если в задаче о рюкзаке есть решение S, удовлетворяющее (3.1), в примере задачи UCFLP на звезде откроем предприятия в вершинах p_i для всех $i \in N$ и предприятия в вершинах q_i для $i \in S$. Для каждого $i \in N$ спрос клиента в p_i обслуживается из вершины p_i с нулевыми транспортными расходами. Для каждого $i \in N$ спрос клиента в вершине q_i обслуживается из q_i, если $i \in S$, с нулевыми транспортными расходами, и из p_i иначе, с расходами на транспортировку $3\gamma \cdot a_i$. Спрос клиента в вершине p_0 удовлетворяется из вершин p_i, $i \in S$: в каждом из этих предприятий после обслуживания клиента в p_i осталось a_i единиц продукта; согласно (3.1) этого достаточно, чтобы совместно покрыть спрос в вершине p_0 с суммарными транспортными расходами $\gamma \cdot A$. Общая стоимость такого решения с учетом (3.1):

$$\sum_{i \in S} (f_i + 3\gamma a_i) + \sum_{i=1}^{n} f_i + 3\gamma \cdot \sum_{i \notin S} a_i + \gamma \cdot A = \sum_{i \in S} f_i + \sum_{i=1}^{n} f_i + 3\gamma \cdot \sum_{i=1}^{n} a_i + \gamma \cdot A \leq F'.$$

($\Rightarrow$) Заметим, что в любом допустимом решении примера UCFLP на звезде стоимости не более F' предприятие в вершине p_0 не будет открыто, и транспортные затраты на обслуживание клиента в p_0 составят не менее $\gamma \cdot A$. Затраты на клиента в q_i составят не менее $3\gamma \cdot a_i$, поскольку, если клиент в q_i обслуживается из q_i, то величина $3\gamma \cdot a_i$ платится при открытии предприятия в q_i, иначе транспортные расходы на обслуживание спроса в q_i составят не менее $3\gamma \cdot a_i$. Также заметим, что если клиенту в p_i со спросом $a - a_i \geq 1$ хотя бы единица

продукта доставляется не из вершины p_i, транспортные расходы на это составят не менее 2γ, и нижняя оценка на стоимость решения превысит F':

$$\gamma \cdot A + 3\gamma \cdot \sum_{i=1}^{n} a_i + 2\gamma = \gamma \cdot A + 3\gamma \cdot \sum_{i=1}^{n} a_i + 4\sum_{i=1}^{n} f_i > F'.$$

Значит, в решении стоимости не более F' в каждой вершине p_i, $i \in N$, должно быть открыто предприятие, которое полностью обслужит спрос находящегося в ней клиента. Поскольку суммарно объемы производства предприятий из p_i, $i \in N$, покрывают лишь $a \cdot n$ единиц спроса, а общий спрос в примере равен $a \cdot n + A$, в решении должны быть открыты предприятия в вершинах q_i, $i \in S$, для некоторого $S \subseteq N$. Предположим, что из вершины q_i, $i \in S$, доставляется хотя бы единица продукта в вершину p_0 или в вершину q_j, $j \neq i$. В первом случае нижняя оценка на транспортные расходы по обслуживанию спроса в p_0 составит $\gamma(A-1)+2\gamma$. Первое слагаемое означает, что для удовлетворения спроса в p_0 не хватает единицы продукта, а второе есть стоимость ее доставки из q_i. Во втором случае затраты на клиента в q_j составят не менее $3\gamma(a_i-1)+4\gamma$. Здесь аналогично первое слагаемое означает, что для удовлетворения спроса в q_j не хватает единицы продукта, а второе есть стоимость ее доставки из q_i. В любом случае нижняя оценка на стоимость решения с учетом необходимости открытия предприятий во всех p_i, $i \in N$, снова превысит F':

$$\sum_{i=1}^{n} f_i + \gamma \cdot A + 3\gamma \cdot \sum_{i=1}^{n} a_i + \gamma = 3\sum_{i=1}^{n} f_i + \gamma \cdot A + 3\gamma \cdot \sum_{i=1}^{n} a_i > F'.$$

Таким образом, из вершины q_i в допустимом решении стоимости не более F' может обслуживаться только клиент из q_i. Поскольку суммарное используемое предложение должно покрывать суммарный спрос, для решения должно выполняться $a \cdot n + \sum_{i \in S} a_i \geq a \cdot n + A$, и, по крайней мере,

$$\sum_{i \in S} f_i + \sum_{i=1}^{n} f_i + \gamma \cdot A + 3\gamma \cdot \sum_{i=1}^{n} a_i \leq F' = F + \sum_{i=1}^{n} f_i + \gamma \cdot A + 3\gamma \cdot \sum_{i=1}^{n} a_i,$$

откуда следует, что $\sum_{i \in S} a_i \geq A$ и $\sum_{i \in S} f_i \leq F$, а значит, S — допустимое решение задачи о рюкзаке (3.1).

Теорема доказана.

Здесь следует отметить, что вариант задачи UCFLP на звезде, в котором в каждой вершине входного графа может располагаться либо предприятие, либо клиент, решается за линейное время, и мы покажем это в разд. 3.2. При этом легко увидеть, что в такой постановке задача тем не менее будет $\mathcal{NP}$-трудной на дереве: достаточно в примере из теоремы 1 каждую вершину p_i (q_i), $i \in N$, соединить ребром нулевой стоимости с новой вершиной p'_i (q'_i) и перенести в новую вершину предприятие.

Следствие 1. *Задача UCFLP, в которой в каждой вершине может находиться либо клиент, либо предприятие, $\mathcal{NP}$-трудна на дереве.*

3.2. Линейный алгоритм решения задачи UCFLP на звезде, если в каждой вершине может находиться либо предприятие, либо клиент

Введем необходимые обозначения. Пусть граф $G = (V, E)$ — звезда с центральной вершиной u_0, висячими вершинами $\{v_1, v_2, \ldots, v_m\} = \mathcal{F}$, в которых расположены предприятия со стоимостями открытия $f_1, \ldots, f_m$, и висячими вершинами $\{u_1, u_2, \ldots, u_n\} = \mathcal{C}$, в которых находятся клиенты со спросами $b_1, \ldots, b_n$. Без ограничения общности можем считать, что в

центральной вершине нет ни клиентов, ни предприятий, иначе всегда можно вынести предприятие или клиента из u_0 в отдельную висячую вершину, соединенную с u_0 ребром нулевой стоимости. Стоимости транспортировки единицы продукта обозначим через c_i для ребер $\{u_0, v_i\}$, $i = 1, \ldots, m$, и c_{j+m} для ребер $\{u_0, u_j\}$, $j = 1, \ldots, n$. Через $\Phi(S)$ будем обозначать значение целевой функции (2.1) для допустимого решения, определяемого набором открытых предприятий S.

Лемма 1. *Задача UCFLP на звезде, где в каждой вершине графа может находиться либо клиент, либо предприятие, за полиномиальное время сводится к задаче UCFLP на звезде, в которой единственный клиент со спросом $B = \sum_{j=1}^{n} b_j$ находится в центральной вершине, а предприятия могут быть открыты только в висячих вершинах.*

Доказательство. Пусть (S, x) — допустимое решение исходной задачи. Его стоимость равна

$$\sum_{i\in S} f_i + \sum_{i\in S}\sum_{j=1}^{n}(c_i + c_{m+j})x_{ij} = \sum_{i\in S} f_i + \sum_{i\in S} c_i \sum_{j=1}^{n} x_{ij} + \sum_{j=1}^{n} c_{m+j} b_j = \sum_{i\in S} f_i + \sum_{i\in S} c_i y_i + D,$$

где $D = \sum_{j=1}^{n} b_j c_{m+j}$ — константа, а переменные $y_i = \sum_{j=1}^{n} x_{ij}$ соответствуют количеству продукта, доставленного из предприятия в вершине v_i в центральную вершину u_0. При этом из ограничений

$$\sum_{i\in S} x_{ij} = b_j,\ 1 \le j \le n, \quad \text{и} \quad \sum_{j\in J} x_{ij} \le a_i,\ 1 \le i \le m,$$

следует

$$\sum_{i\in S} y_i = B \quad \text{и} \quad y_i \le a_i,\ 1 \le i \le m.$$

Таким образом, (S, y) — допустимое решение задачи UCFLP на звезде, в которой единственный клиент со спросом $B = \sum_{j=1}^{n} b_j$ находится в центральной вершине, а предприятия могут быть открыты только в висячих вершинах. Стоимость решения (S, y) отличается от стоимости решения (S, x) исходной задачи на константу, и оптимум в обоих случаях достигается на одном и том же наборе открытых предприятий S, который согласно замечанию 1 определяет решение обеих задач.

Лемма доказана.

Лемма 2. *Для задачи UCFLP на звезде, в которой в каждой вершине может находиться либо клиент, либо предприятие, существует оптимальное решение, в котором открыто не более одного ненасыщенного предприятия.*

Доказательство. Предположим, найдутся предприятия i и k такие, что $\sum_{j=1}^{n} x_{ij} < a$ и $\sum_{j=1}^{n} x_{kj} < a$, тогда их вклад в стоимость решения составит $f_i + c_i \sum_{j=1}^{n} x_{ij} + f_k + c_k \sum_{j=1}^{n} x_{kj}$. Не ограничивая общности, полагаем, что $c_k \ge c_i$, тогда решение, в котором i насыщенное, а k обслуживает $(\sum_{j=1}^{n} x_{kj} - (a - \sum_{j=1}^{n} x_{ij}))$ единиц спроса, не хуже:

$$f_i + c_i a + f_k + c_k(\sum_{j=1}^{n} x_{kj} - (a - \sum_{j=1}^{n} x_{ij})) = f_i + c_i \sum_{j=1}^{n} x_{ij} + c_i(a - \sum_{j=1}^{n} x_{ij}) + f_k + c_k \sum_{j=1}^{n} x_{kj}$$

$$- c_k(a - \sum_{j=1}^{n} x_{ij}) \le f_i + c_i \sum_{j=1}^{n} x_{ij} + f_k + c_k \sum_{j=1}^{n} x_{kj}.$$

Таким образом, можно считать, что в оптимальном решении содержится не более одного ненасыщенного предприятия, а остальные открытые предприятия насыщенные.

Лемма доказана.

Далее мы будем использовать следующий факт. В произвольном массиве из m элементов k-й по неубыванию элемент может быть найден за время $\mathcal{O}(m)$ [19]. Описание соответствующего алгоритма из работы [19] представлено также в монографии [4, гл. 3.6, алгоритм 3.6].

Теперь приведем описание линейного по времени жадного алгоритма $\mathcal{A}$ для задачи UCFLP на звезде, в которой в каждой вершине может находиться либо клиент, либо предприятие.

Описание алгоритма $\mathcal{A}$

Ш а г 0. Положим $B = \sum_{j=1}^{n} b_j$. Всех клиентов из висячих вершин $\{u_1, \ldots, u_n\}$ переместим в центральную вершину u_0 и будем рассматривать их как единственного клиента со спросом B.

Ш а г 1. Положим $k = \lceil B/a \rceil$. Построим множество S, состоящее из первых k предприятий с наименьшими значениями величины $h_i = f_i + c_i a$, $i = 1, \ldots, m$. Для этого алгоритмом 3.6 из [4] найдем k-е по неубыванию величины h_i предприятие i_k и добавим его в S. Внесем в S все предприятия i такие, что $h_i < h_{i_k}$, и затем, пока в S не наберется k элементов, добавим в S предприятия i, для которых $h_i = h_{i_k}$. Положим $b = ak - B$. Если $b = 0$, считаем все предприятия в S насыщенными и перейдем к шагу 3, иначе перейдем к шагу 2.

Ш а г 2. Найдем оптимальный набор открытых предприятий с единственным ненасыщенным предприятием i^*, обслуживающим оставшийся спрос $a - b$, следующим образом. Для каждого $i = 1, \ldots, m$ рассмотрим набор предприятий S_i с ненасыщенным предприятием i. Если $i \notin S$, в качестве насыщенных добавим в S_i все предприятия из $S \setminus \{i_k\}$, иначе в качестве насыщенных добавим предприятия $S \setminus \{i\}$. Обозначим $H = \sum_{i \in S} h_i$, тогда затраты, связанные с использованием предприятий S_i, вычисляются как

$$\Phi_i = \begin{cases} H - c_i b, & \text{если } i \in S, \\ H - h_{i_k} + f_i + c_i(a - b), & \text{если } i \notin S, \end{cases}$$

Найдем $i^* = \arg\min_{1 \le i \le m} \Phi_i$ и положим $S = S_{i^*}$.

Ш а г 3. По найденному набору открытых предприятий S для каждого клиента j с ненулевым спросом вычислим положительные значения плана перевозки x_{ij}. А именно, переупорядочим предприятия так, чтобы первыми k предприятиями были предприятия из S и ненасыщенное предприятие, если оно есть в S, было k-м. Для каждого $j = 1, \ldots, n$, вычислим $B_j = B_{j-1} + b_j$ при $B_0 = 0$, $i'_j = \lfloor B_{j-1}/a \rfloor + 1$, $i''_j = \lceil B_j/a \rceil$ и найдем положительные значения переменных x_{ij}:

$$x_{ij} = \begin{cases} b_j, & \text{если } i = i'_j = i''_j, \\ i'_j a - B_{j-1}, & \text{если } i = i'_j < i''_j, \\ B_j - (i''_j - 1)a, & \text{если } i'_j < i''_j = i, \\ a, & \text{если } i'_j < i < i''_j. \end{cases}$$

Вернем решение (S, x).

Утверждение 1. *Трудоемкость алгоритма $\mathcal{A}$ равна $\mathcal{O}(m + n)$.*

Д о к а з а т е л ь с т в о. Шаг 0 алгоритма выполняется за время $\mathcal{O}(n)$. На шаге 1 строится множество S, состоящее из первых k предприятий с наименьшими значениями величины h_i. Для этого, чтобы не тратить $\mathcal{O}(m \log m)$ времени на сортировку, используется линейный алгоритм 3.6 из [4] для нахождения k-го по неубыванию величины h_i предприятия i_k, после чего за время $\mathcal{O}(m)$ в S добавляются оставшиеся необходимые элементы. Шаг 2 выполняется суммарно за время $\mathcal{O}(m)$. На шаге 3 для каждого $j = 1, \ldots, n$ такого, что $b_j > 0$, положительные

значения принимают переменные x_{ij} при $i \in [i'_j, i''_j]$, каждое значение x_{ij} вычисляется за $\mathcal{O}(1)$, а общее число положительных переменных не превосходит

$$\sum_{j=1}^{n}(i''_j - i'_j + 1) = \sum_{j=1}^{n}(\lceil B_j/a\rceil - (\lfloor B_{j-1}/a\rfloor + 1) + 1) = \lceil B_n/a\rceil + \sum_{j=1}^{n-1}(\lceil B_j/a\rceil - \lfloor B_j/a\rfloor) \le m + n,$$

поскольку $B_n = B \le ma$. Таким образом, шаг 3 реализуется за время $\mathcal{O}(m+n)$, и суммарная трудоемкость алгоритма составляет $\mathcal{O}(m+n)$.

Утверждение доказано.

Докажем корректность предложенного алгоритма.

Теорема 2. *Алгоритм $\mathcal{A}$ находит оптимальное решение для UCFLP на звезде, если в каждой вершине может находиться либо клиент, либо предприятие.*

Доказательство. Согласно лемме 1 нам достаточно найти множество открытых предприятий, считая, что в центральной вершине находится единственный клиент со спросом $B = \sum_{j=1}^{n} b_j$. Согласно лемме 2 существует оптимальное решение задачи, в котором среди открытых предприятий S^* содержится не более одного ненасыщенного. Если $b = 0$, то S^* должно состоять из $k = B/a$ насыщенных предприятий. В этом случае построенное алгоритмом $\mathcal{A}$ на шаге 1 множество открытых предприятий S оптимально, поскольку оно состоит из первых k предприятий с наименьшими затратами $h_i = f_i + c_i a$.

Если $b \neq 0$, то S^* должно содержать $k-1$ насыщенное предприятие и одно ненасыщенное. Пусть в оптимальном решении предприятие $t \in S^*$ единственное ненасыщенное. Тогда для множества открытых предприятий S, построенного на шаге 2, верно

$$\Phi(S) = \Phi(S_{i^*}) \le \Phi(S_t) = D + \sum_{i \in S_t \setminus \{t\}} h_i + f_t + c_t(a-b) \le D + \sum_{i \in S^* \setminus \{t\}} h_i + f_t + c_t(a-b) = \Phi(S^*),$$

где $D = \sum_{j=1}^{n} c_{m+j} b_j$ — константа из леммы 1, а второе неравенство выполняется, поскольку по построению в качестве насыщенных предприятий в S_t взяты первые $k-1$ предприятий из $I \setminus \{t\}$ с наименьшими затратами h_i. Таким образом, решение, в котором открыты предприятия из S, оптимально.

Поскольку можно считать, что все клиенты находятся в центральной вершине, стоимость решения не зависит от того, какое из открытых предприятий обслуживает какого клиента. Поэтому на шаге 3 при вычислении плана перевозки (x_{ij}) достаточно действовать "жадно", назначая значения неотрицательных x_{ij} согласно разбиению интервала длины $B = \sum_{i=1}^{n} b_j$, составленного из интервалов с длинами, равными клиентским спросам $b_j > 0$, на интервалы, соответствующие предложению предприятий из S, длины a каждый, кроме, возможно, последнего длины $a - b$.

Теорема доказана.

4. Точный полиномиальный алгоритм с улучшенной трудоемкостью для задачи UCFLP на цепи

В этом разделе рассматривается задача UCFLP, когда входной граф является цепью, т. е. связным графом, в котором каждая вершина, кроме концевых, имеет степень 2, а концевые — степень 1.

4.1. Свойства оптимальных решений UCFLP на цепи

Приведем ключевые свойства оптимальных решений UCFLP на цепи, доказанные в работах [3; 13], на которые опирается алгоритм динамического программирования для рассматриваемой задачи.

Будем считать, что в заданном графе-цепи $G = (V, E)$ вершины, множество клиентов и множество предприятий нумеруются слева направо вдоль цепи, так что клиент $j \in J$ является j-м клиентом по счету слева и находится в вершине u_j, а предприятие $i \in I$ является i-м по счету слева и находится в вершине v_i.

О п р е д е л е н и е 2. *Сегментом предприятий* $[i', i'']$ будем называть подмножество $\{i \in I \mid i' \le i \le i''\}$ для любых $1 \le i' \le i'' \le m$, а *сегментом клиентов* $[j', j'']$ будем называть подмножество $\{j \in J \mid j' \le j \le j''\}$ для любых $1 \le j' \le j'' \le n$.

Первое полезное свойство состоит в том, что существуют оптимальные решения для UCFLP на цепи, в которых множество клиентов разбивается на непрерывные сегменты, и для каждого сегмента $[j', j'']$ существует одно открытое предприятие в решении, которое полностью обслуживает клиентов в этом сегменте за исключением, возможно, только клиентов j' и j''.

Лемма 3 [3, лемма 1; 13]. *Для задачи UCFLP на цепи существует оптимальное решение* (S, x), *удовлетворяющее следующему свойству*:

(1^0) *для любых* $i', i'' \in S$, *и* $j', j'' \in J$, *таких, что* $i' < i''$ *и* $j' < j''$, *либо* $x_{i'j''} = 0$, *либо* $x_{i''j'} = 0$.

Отметим, что свойство (1^0) верно и в случае произвольных объемов производства. Следующее важное свойство оптимальных решений UCFLP на цепи касается ненасыщенных предприятий.

Лемма 4 [3, лемма 3; 13]. *Задача UCFLP на цепи имеет оптимальное решение* (S, x), *удовлетворяющее свойству* (1^0) *и следующему свойству*:

(2^0) *для любых ненасыщенных предприятий* i' *и* i'' *сегмент предприятий* $[i', i'']$ *содержит открытые предприятия* i^* *и* i^{**} *такие, что они не делят клиента, и не существует открытых предприятий в сегменте* $[i^* + 1, i^{**} - 1]$.

4.2. Полиномиальный алгоритм решения UCFLP на цепи

Теперь опишем полиномиальный алгоритм динамического программирования [3; 13] для задачи UCFLP на цепи, основанный на структурных свойствах (1^0) и (2^0), и покажем, как улучшить его трудоемкость до $\mathcal{O}(m^2n^2)$.

Для любых $i \in \{0, 1, \ldots, m\}$ и $j \in \{0, 1, \ldots, n\}$ пусть $R(i, j)$ обозначает стоимость оптимального решения для подзадачи с первыми i предприятиями и первыми j клиентами. Следовательно, значение $R(m, n)$ — это стоимость оптимального решения изначальной UCFLP на цепи. Оптимальное решение, удовлетворяющее свойствам (1^0) и (2^0), может быть найдено с помощью следующей схемы динамического программирования:

$$R(i, 0) := 0, \;\; 0 \le i \le m, \quad R(0, j) := +\infty, \;\; 1 \le j \le n, \tag{4.1}$$

для всех $1 \le i \le m$, $1 \le j \le n$

$$R(i, j) = \min_{1 \le i' \le i,\, 1 \le j' \le j} \Big\{ R(i' - 1, j' - 1) + Q_{i',i}(j', j) \Big\}, \tag{4.2}$$

где $Q_{i',i}(j', j)$ — стоимость оптимального решения для примера с сегментом предприятий $[i', i]$, $1 \le i' \le i \le m$ и сегментом клиентов $[j', j]$, $1 \le j' \le j \le n$, в котором не более одного предприятия из $[i', i]$ ненасыщенно. В работе [3, разд. 3–4] было показано, как вычислить все величины $Q_{i',i}(j', j)$ суммарно за время $\mathcal{O}(m^4n^2)$. Далее мы приведем схему динамического программирования для вычисления $Q_{i',i}(j', j)$ из [3] и покажем, как можно более эффективно вычислять величины $Q_{i',i}(j', j)$ и составляющие их компоненты.

Утверждение 2 [3, разд. 3].

$$Q_{i',i''}(j',j'') = \min_{1\le t\le r,\ i'\le i\le i''}\{\widetilde{G}^L_{i-1,t-1}(j') + S_{it}(j',j'') + \widetilde{G}^R_{i+1,r-t}(j'')\}, \tag{4.3}$$

где
$r = \lceil (b_{j'} + \ldots + b_{j''})/a \rceil$ — *оптимальное число предприятий для обслуживания спроса в сегменте* $[j', j'']$,
$\widetilde{G}^L_{i-1,t-1}(j')$ — *оптимальные затраты, связанные с открытием* $t-1$ *насыщенных предприятий в сегменте* $[i', i-1]$ *и обслуживанием спроса в размере* $a(t-1)$ *для первых клиентов сегмента* $[j', j'']$,
$\widetilde{G}^R_{i+1,r-t}(j'')$ — *оптимальные затраты, связанные с открытием* $r-t$ *насыщенных предприятий в сегменте* $[i+1, i'']$ *и обслуживанием клиентского спроса в размере* $a(r-t)$ *для последних клиентов сегмента* $[j', j'']$,
$S_{it}(j', j'')$ — *затраты, связанные с выбором предприятия* i *в качестве единственного, возможно, ненасыщенного и* t*-го по счету открытого предприятия в* $[i', i'']$ *и соответствующими транспортными затратами на обслуживание оставшегося спроса в сегменте* $[j', j'']$.

Д о к а з а т е л ь с т в о. Очевидно, что оптимально открыть $r = \lceil (b_{j'} + \ldots + b_{j''})/a \rceil$ предприятий в сегменте $[i', i'']$ для обслуживания спроса всех клиентов из $[j', j'']$. Таким образом, идея вычисления $Q_{i',i}(j', j)$ состоит в том чтобы определить, какие r предприятий в сегменте $[i', i'']$ открыть, и найти порядковый номер t одного ненасыщенного открытого предприятия. Для наглядности удобно рассматривать отрезок прямой линии длины $\sum_{j=j'}^{j''} b_j$, который соответствует суммарному спросу сегмента клиентов $[j', j'']$. Наша задача — покрыть этот отрезок множеством из $(r-1)$ отрезков длины a и одного блока длины $\sum_{j=j'}^{j''} b_j - (r-1)a$. Тогда, двигая по отрезку суммарного спроса блок длины $\sum_{j=j'}^{j''} b_j - (r-1)a$, соответствующий, возможно, ненасыщенному предприятию, мы сможем эффективно вычислить значение $Q_{i',i''}(j', j'')$.

Введем дополнительные обозначения: $B_j = \sum_{k=1}^{j} b_k$ — общий спрос клиентов из сегмента $[1, j]$, $1 \le j \le n$; $C_{ij} = \sum_{k=1}^{j} g_{ik} b_k$ — транспортные затраты, связанные с удовлетворением суммарного спроса клиентов из сегмента $[1, j]$ с помощью i-го предприятия, где $C_{i0} = 0$; $r = r(i', i'', j', j'') = \lceil (B_{j''} - B_{j'-1})/a \rceil$ — количество предприятий, которые необходимо открыть в сегменте $[i', i'']$ для удовлетворения суммарного спроса сегмента клиентов $[j', j'']$, при условии, что $[i', i'']$ является сегментом с не более одним ненасыщенным предприятием. Для простоты записи мы будем опускать зависимость r от индексов i', i'', j', j'', когда из контекста понятно, какие индексы имеются в виду.

Сначала покажем, как вычислить величины $\widetilde{G}^L_{i,t}(j')$, $1 \le t \le i \le m$. Найдем величины $\xi_{i,t,j'}$, равные сумме транспортных затрат и стоимости открытия насыщенного предприятия $i \in [i', i'']$, при условии, что среди предприятий из $[i', i-1]$ в точности $(t-1)$ открытых и все они являются насыщенными (мы предполагаем, что $t < r$). Пересечение областей обслуживания предприятия i с соседними открытыми предприятиями согласно лемме 4 происходит для некоторых клиентов j^*_{t-1} и j^*_t. Возможны следующие два случая.

1. $j^*_{t-1} = j^*_t$. Тогда насыщенное предприятие i обслуживает только одного клиента j^*_t и

$$\xi_{i,t,j'} = f_i + g_{ij^*_t} a.$$

2. $j^*_{t-1} < j^*_t$. В этом случае насыщенное предприятие i обслуживает весь спрос всех клиентов из интервала (j^*_{t-1}, j^*_t). Транспортные затраты на обслуживание этого интервала клиентов равны

$$(C_{ij^*_t - 1} - C_{ij^*_{t-1}}).$$

Транспортные затраты на обслуживание недостающей части спроса в j^*_{t-1} и возможной части спроса j^*_t, соответственно, равны

$$g_{ij^*_{t-1}}\big(B_{j^*_{t-1}} - B_{j'-1} - (t-1)a\big) \quad \text{и} \quad g_{ij^*_t}\big(ta - (B_{j^*_t - 1} - B_{j'-1})\big).$$

Таким образом, общие затраты составят

$$\xi_{i,t,j'} = f_i + \big(C_{ij^*_t-1} - C_{ij^*_{t-1}}\big) + g_{ij^*_{t-1}}\big(B_{j^*_{t-1}} - B_{j'-1} - (t-1)a\big) + g_{ij^*_t}\big(ta - (B_{j^*_t-1} - B_{j'-1})\big).$$

Введем вспомогательное множество клиентов $T = \{j^*_1, \ldots, j^*_{r-1}\}$ следующим образом:

$$B_{j^*_t-1} - B_{j'-1} < ta \le B_{j^*_t} - B_{j'-1}, \quad 1 \le t < r.$$

Если первые k открытых предприятий $\{i_1, i_2, \ldots, i_k\}$, $k \le r$, являются насыщенными, тогда по определению T для любых $s = 1, \ldots, k-1$ либо пересечение областей обслуживания соседних предприятий i_s и i_{s+1} происходит на клиенте j^*_s, либо, если области обслуживания не пересекаются, предприятие i_s обслуживает клиента j^*_s, а предприятие i_{s+1} обслуживает клиента $j^*_s + 1$ (клиент j^*_s принадлежит сегменту $[j', j'']$).

Обозначим через $G^L_{i,t}(j')$ оптимальные транспортные затраты плюс затраты на открытие t насыщенных предприятий из сегмента $[i', i]$, в ситуации, когда предприятие i открыто (мы предполагаем, что $|\{i', \ldots, i\}| = i - i' + 1 \ge t$). Тогда для величин $G^L_{i,t}(j')$ и $\widetilde{G}^L_{i,t}(j')$ выполняются следующие рекуррентные соотношения:

$$G^L_{i,t}(j') = \begin{cases} \xi_{i,1,j'}, & \text{если } t = 1, \\ \xi_{i,t,j'} + \widetilde{G}^L_{i-1,t-1}(j'), & \text{если } 2 \le t \le i - i' + 1, \end{cases}$$

$$\widetilde{G}^L_{i,t}(j') = \begin{cases} \min\limits_{i' \le k \le i} G^L_{k,1}(j') = \min\limits_{i' \le k \le i} \xi_{k,1,j'}, & \text{если } t = 1, \\ \min\limits_{i' \le k \le i,\ (k-i'+1) > t} G^L_{k,t}(j') = \min\big\{G^L_{i,t}(j'), \widetilde{G}^L_{i-1,t}(j')\big\}, & \text{если } 2 \le t \le i - i' + 1. \end{cases}$$

Используя аналогичную схему, мы можем найти значения величин $\widetilde{G}^R_{i,t}(j'')$ и $G^R_{i,t}(j'')$, которые соответствуют сумме транспортных затрат и затрат на открытие t насыщенных предприятий в сегменте $[i, i'']$. Разница состоит в том, что в предыдущих рассуждениях открывались насыщенные предприятия с левого конца сегмента $[i', i'']$ и эти предприятия обслуживали некоторый сегмент клиентов $[j', j_s] \subseteq [j', j'']$. Теперь мы рассматриваем открытые предприятия, начиная с правого конца сегмента $[i', i'']$, которые обслуживают сегмент клиентов $[j_s, j''] \subseteq [j', j'']$. Обозначим через $\xi'_{i,t,j''}$ транспортные затраты плюс затраты на открытие насыщенного предприятия $i \in [i', i'']$ при условии, что среди предприятий $[i+1, i'']$ в точности $(t-1)$ открытых и все они насыщенные. Введем величины B'_j, $T' = \{j^{**}_1, \ldots, j^{**}_{r-1}\}$, аналогичные величинам B_j, $T = \{j^*_1, \ldots, j^*_{r-1}\}$. Легко увидеть, что их можно вычислить аналогичным способом:

$$B'_j = \sum_{k=j}^{n} b_k,\ 1 \le j \le n, \quad r = \big\lceil (B'_{j'} - B'_{j''+1})/a \big\rceil,$$

$$T' = \{j^{**}_1, \ldots, j^{**}_{r-1}\}\colon\ B'_{j^{**}_t+1} - B'_{j''+1} < ta \le B'_{j^{**}_t} - B'_{j''+1},\ \ t = 1, \ldots, r-1,$$

$$\xi'_{i,t,j''} = f_i + \big(C_{ij^{**}_t+1} - C_{ij^{**}_{t-1}}\big) + g_{ij^{**}_{t-1}}\big(B'_{j^{**}_{t-1}} - B'_{j''+1} - (t-1)a\big) + g_{ij^{**}_t}\big(ta - (B'_{j^{**}_t+1} - B'_{j''+1})\big).$$

Тогда величины $G^R_{i,t}(j'')$ и $\widetilde{G}^R_{i,t}(j'')$ находятся согласно рекуррентным соотношениям

$$G^R_{i,t}(j'') = \begin{cases} \xi'_{i,1,j''}, & \text{если } t = 1, \\ \xi'_{i,t,j''} + \widetilde{G}^R_{i+1,t-1}(j''), & \text{если } 2 \le t \le i'' - i + 1, \end{cases}$$

$$\widetilde{G}^R_{i,t}(j'') = \begin{cases} \min\limits_{i \le k \le i''} G^R_{k,1}(j'') = \min\limits_{i \le k \le i''} \xi'_{k,1,j''}, & \text{если } t = 1, \\ \min\limits_{i \le k \le i'',\ (i''-k+1) > t} G^R_{k,t}(j'') = \min\big\{G^R_{i,t}(j''), \widetilde{G}^R_{i+1,t}(j'')\big\}, & \text{если } 2 \le t \le i'' - i + 1. \end{cases}$$

Оставшиеся величины $S_{it}(j', j'')$ вычисляются с помощью следующей формулы, при условии что $i - i' + 1 \geq t$:

$$S_{it}(j', j'') = f_i + (C_{ij^{**}_{r-t}-1} - C_{ij^{*}_{t-1}}) + g_{ij^{*}_{t-1}}(B_{j^{*}_{t-1}} - B_{j'-1} - (t-1)a) + g_{ij^{**}_{r-t}}(B'_{j^{**}_{r-t}} - B'_{j''+1} - (r-t)a).$$

Теорема доказана.

Лемма 5. *Вычисление всех значений $\widetilde{G}^L_{i,t}(j')$, $\widetilde{G}^R_{i,t}(j'')$ и $S_{it}(j', j'')$ может быть выполнено за время $\mathcal{O}(m^2n)$, $\mathcal{O}(m^2n)$ и $\mathcal{O}(m^2n^2)$, соответственно.*

Доказательство. Вычисление всех значений B_j и B'_j требует $\mathcal{O}(n)$ времени, нахождение всех C_{ij} может быть выполнено за время $\mathcal{O}(mn)$. Построение множеств T и T' для каждого фиксированного j' требует не более $\mathcal{O}(m)$ действий, следовательно, суммарно необходимо $\mathcal{O}(mn)$ операций. Величины $\xi_{i,t,j'}$ и $\xi'_{i,t,j''}$ для всех индексов могут быть найдены за общее время $\mathcal{O}(m^2n)$. Зная значения $\xi_{i,t,j'}$ и $\xi'_{i,t,j''}$, величины $\widetilde{G}^L_{i,t}(j')$ и $G^L_{i,t}(j')$, $\widetilde{G}^R_{i,t}(j'')$ и $G^R_{i,t}(j'')$ можно последовательно вычислить за суммарное время $\mathcal{O}(m^2n)$. Наконец, определение величин $S_{it}(j', j'')$ для всех индексов i, t, j', j'', займет $\mathcal{O}(m^2n^2)$ времени.

Лемма доказана.

Теперь, на основе этих утверждений, мы можем доказать основной результат этого раздела.

Теорема 3. *Задача UCFLP на цепи может быть решена за время $\mathcal{O}(m^2n^2)$.*

Доказательство. Из леммы 5 следует, что все значения $\widetilde{G}^L_{i,t}(j')$, $\widetilde{G}^R_{i,t}(j'')$ и $S_{it}(j', j'')$ могут быть найдены суммарно за время $T_1 = \mathcal{O}(m^2n^2)$. Вычислим эти величины и введем следующие обозначения для всех $1 \leq i \leq m$, $1 \leq j' \leq j'' \leq n$:

$$\widetilde{Q}_i(j', j'') = \min_{1 \leq t \leq r} \{\widetilde{G}^L_{i-1,t-1}(j') + S_{it}(j', j'') + \widetilde{G}^R_{i+1,r-t}(j'')\}. \tag{4.4}$$

Отметим, что теперь требуется $T_2 = \mathcal{O}(m^2n^2)$ времени для нахождения всех значений $\widetilde{Q}_i(j', j'')$. С учетом соотношений (4.3) и (4.4) мы получаем следующую формулу для вычисления $Q_{i',i''}(j', j'')$:

$$Q_{i',i''}(j', j'') = \min_{i' \leq i \leq i''} \widetilde{Q}_i(j', j'') = \min\{Q_{i',i''-1}(j', j''); \widetilde{Q}_i(j', j'')\}.$$

Таким образом, зная все величины $\widetilde{Q}_i(j', j'')$ и предыдущее значение $Q_{i',i''-1}(j', j'')$, можно найти следующее значение $Q_{i',i''}(j', j'')$ за константное время. Следовательно, вычисление $Q_{i',i''}(j', j'')$ для всех индексов $1 \leq i' \leq i'' \leq m$, $1 \leq j' \leq j'' \leq n$ может быть выполнено за время $T_3 = \mathcal{O}(m^2n^2)$.

Окончательно из (4.1), (4.2) следует, что, зная все значения $Q_{i',i''}(j', j'')$, нахождение всех $R(i, j)$ требует $T_4 = \mathcal{O}(m^2n^2)$ времени. Таким образом, общая трудоемкость алгоритма оценивается величиной $T_1 + T_2 + T_3 + T_4 = \mathcal{O}(m^2n^2)$.

Теорема доказана.

5. Точный псевдополиномиальный алгоритм с улучшенной трудоемкостью для задачи CFLP на цепи

Из работы Мирчандани и др. [28] следует, что для задачи CFLP на цепи существует точный псевдополиномиальный алгоритм динамического программирования с трудоемкостью $\mathcal{O}(mB \min\{a_{max}, B\})$, где $a_{max} = \max_{i \in I} a_i$ — максимальный объем производства предприятия, $B = \sum_{j \in J} b_j$ — суммарный клиентский спрос. В этом разделе мы покажем, как можно улучшить время работы этого алгоритма до $\mathcal{O}(mB)$.

5.1. Алгоритм динамического программирования

Будем считать, что в заданном графе-цепи $G = (V, E)$ множество вершин $V = \{1, \dots, n_V\}$, и вершины нумеруются слева направо вдоль цепи. Также считаем, что предприятие $i \in I = \{1, \dots, m\}$ является i-м по счету слева направо и находится в вершине $v_i \in \mathcal{F}$, а клиент $j \in J = \{1, \dots, n\}$ является j-м клиентом по счету слева направо и находится в вершине $u_j \in \mathcal{C}$. Если такая нумерация не задана, ее всегда можно найти за время $\mathcal{O}(|V| \log |V|)$.

На предварительном шаге задача CFLP на цепи $G = (V, E)$ с n клиентами за время $\mathcal{O}(B + m)$ сводится к задаче CFLP на цепи $G' = (V', E')$ с $B = \sum_{j \in \mathcal{C}} b_j$ клиентами, имеющими единичные спросы. Для этого естественным образом каждая вершина u_t, содержащая клиента t со спросом $b_t > 1$, заменяется подцепью из b_t вершин, в каждой из которых находится клиент с единичным спросом. Стоимости транспортировки единицы продукта по ребру между последовательными клиентами такой подцепи полагаются равными нулю. В новой цепи $G' = (V', E')$ число вершин $|V'| = n' \leq m + B$.

Пусть $S(i, j)$ — оптимальное значение целевой функции подзадачи CFLP с единичными спросами клиентов на цепи, в которой первые j клиентов цепи должны быть обслужены с помощью первых i предприятий. Для каждой пары индексов k, j, $1 \leq k < j \leq B$, обозначим через $w_i(k, j)$ суммарные транспортные затраты, необходимые для обслуживания спросов всех клиентов из интервала $(k, j]$ предприятием i, т. е. $w_i(k, j) = \sum_{t=k+1}^{j} g_{it}$. Тогда задача решается с помощью схемы динамического программирования

$$S(i, 0) := 0, \quad 0 \leq i \leq m, \quad S(0, j) := H, \quad 1 \leq j \leq B, \tag{5.1}$$

для всех $i = 1, \dots, m$, $j = 1, \dots, B$

$$S(i, j) = \min \Big\{ H,\ S(i-1, j),\ f_i + \min_{\max\{j - a_i, 1\} \leq k < j} \{ S(i-1, k) + w_i(k, j) \} \Big\}, \tag{5.2}$$

где $H = \sum_{i=1}^{m} f_i + B \cdot \sum_{e \in E} c_e$ — достаточно большое число, отвечающее отсутствию допустимого решения в соответствующей подзадаче. Если известны значения $w_i(k, j)$, таблица S вычисляется за время $\mathcal{O}(mB \min\{a_{max}, B\})$, где a_{max} — максимальный объем производства предприятия [28]. Оптимум целевой функции задачи равен $S(m, B)$, а само решение восстанавливается обратным ходом алгоритма за время $\mathcal{O}(m)$, если хранить для каждого элемента (i, j) таблицы индекс k, на котором достигается минимум в (5.2).

Покажем, как эффективно вычислять значения $w_i(k, j)$.

Лемма 6. *После предварительных вычислений, требующих $\mathcal{O}(B + m)$ времени, значение $w_i(k, j)$ может быть найдено за время $\mathcal{O}(1)$ для любых $1 \leq i \leq m$, $1 \leq k < j \leq B$.*

Д о к а з а т е л ь с т в о. Рассмотрим частичные суммы

$$d(t) = \sum_{s=1}^{t} c_{(s-1,s)}, \quad D(t) = \sum_{j=1}^{t} d(j) b'_j \quad \text{и} \quad B(t) = \sum_{j=1}^{t} b'_j$$

для всех $t = 1, \dots, n' \leq B + m$, где $c_{(0,1)} = 0$ и

$$b'_t = \begin{cases} 1, & \text{если в вершине } v_t \text{ находится клиент,} \\ 0 & \text{иначе.} \end{cases}$$

Эти суммы можно вычислить рекурсивно за время $\mathcal{O}(B + m)$. Тогда значения $w_i(k, j)$ вычисляются следующим образом. Если $v_i < u_k < u_j$, то

$$w_i(k, j) = \sum_{t=k+1}^{j} g_{it} = \sum_{t=u_k+1}^{u_j} (d(t) - d(v_i)) b'_t = \sum_{t=u_k+1}^{u_j} d(t) b'_t - \sum_{t=u_k+1}^{u_j} d(v_i) b'_t$$

$$= D(u_j) - D(u_k) - d(v_i) \sum_{t=u_k+1}^{u_j} b'_t = D(u_j) - D(u_k) - d(v_i)(B(u_j) - B(u_k)).$$

Если $u_k < u_j < v_i$, аналогично получим

$$w_i(k,j) = D(u_k) - D(u_j) + d(v_i)(B(u_j) - B(u_k)).$$

Если $u_k < v_i < u_j$, то

$$w_i(k,j) = \sum_{t=u_k+1}^{v_i} (d(v_i) - d(t))b'_t + \sum_{t=v_i+1}^{u_j} (d(t) - d(v_i))b'_t = -(D(v_i) - D(u_k)) + (D(u_j) - D(v_i))$$
$$+ d(v_i)\Big(\sum_{t=u_k+1}^{v_i} b'_t - \sum_{t=v_i+1}^{u_j} b'_t\Big) = D(u_k) + D(u_j) - 2D(v_i) - d(v_i)(B(u_k) + B(u_j) - 2B(v_i)).$$

Таким образом, зная значения $d(t)$, $D(t)$ и $B(t)$ для каждого $t = 1, \ldots, n'$, можно найти $w_i(k,j)$ за время $\mathcal{O}(1)$.

Лемма доказана.

5.2. Улучшенный алгоритм для задачи CFLP на цепи

В этом разделе мы улучшим время работы алгоритма динамического программирования (5.1), (5.2) до $\mathcal{O}(mB)$. Будем последовательно вычислять строки таблицы S и покажем, как, зная значения элементов в строке $i-1$, вычислить строку i за время $\mathcal{O}(B)$. Для этого нам понадобится алгоритм из [14, разд. 4], который находит минимальный элемент в каждом столбце полностью монотонной $\alpha \times \beta$-матрицы суммарно за время $\mathcal{O}(\alpha + \beta)$.

О п р е д е л е н и е 3. $\alpha \times \beta$-матрица A с действительными элементами называется *монотонной по столбцам*, если для каждой пары столбцов с индексами $j_0 < j_1$ верно

$$i(j_0) \le i(j_1),$$

где $i(j)$ — наименьший индекс строки i такой, что элемент $A(i,j)$ равен минимальному значению в j-м столбце матрицы A. Матрица A называется *полностью монотонной по столбцам*, если любая 2×2 подматрица матрицы A монотонна по столбцам.

Отметим, что в работе [14] монотонность матрицы определялась в терминах позиции максимального элемента в каждой строке, но все результаты можно легко адаптировать для нашего случая, транспонировав матрицу и поменяв знаки соответствующих неравенств в алгоритме.

Утверждение 3 [14]. *Пусть $\alpha \times \beta$-матрица A полностью монотонна по столбцам. Существует алгоритм* [14, разд. 4]*, который находит минимальный элемент в каждом столбце матрицы A суммарно за время $\mathcal{O}(\alpha + \beta)$.*

Рассмотрим i-ю строку S, заданную соотношением (5.2). Для вычисления каждого элемента $S(i,j)$, $j = 1, \ldots, B$, необходимо найти $\min\limits_{1 \le k \le B} A_i(k,j)$, где $B \times B$-матрица A_i определяется следующим образом:

$$A_i(k,j) = \begin{cases} S(i-1,k) + w_i(k,j), & \text{если } \max\{j - a_i,\ 1\} \le k < j \text{ и } S(i-1,k) < H, \\ H + B - k & \text{если } 1 \le k < j - a_i \text{ и } S(i-1,k) < H, \\ H + B, & \text{иначе.} \end{cases} \tag{5.3}$$

Здесь $H = \sum_{i=1}^{m} f_i + B \cdot \sum_{e \in E} c_e$ — достаточно большое число, и $H > S(i,k)$ для любых $i = 1, \ldots, m$, $k = 1, \ldots, B$. Элементы $A_i(k,j) \ge H$ отвечают отсутствию допустимого решения соответствующей подзадачи, и их значения выбраны так, чтобы впоследствии матрица A_i оказалась строго монотонной по столбцам. Отметим, что матрица A_i определена корректно: элементы $S(i-1,k)$ известны после вычисления значений $(i-1)$-й строки таблицы S, а каждое значение $w_i(k,j)$ можно найти за время $\mathcal{O}(1)$ ввиду леммы 6. Зная минимальные элементы

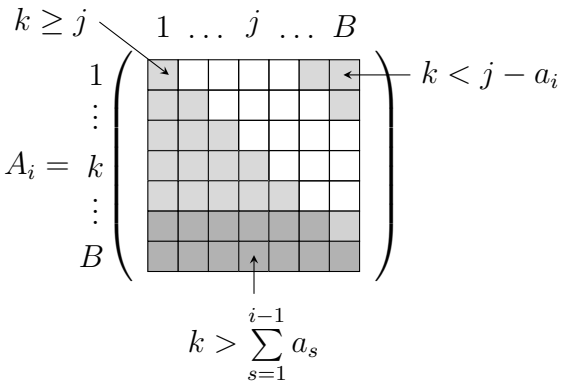

Рис. 2. Схематичный рисунок матрицы A_i с выделенными серыми и белыми элементами.

в каждом столбце матрицы A_i, можно вычислить i-ю строку таблицы S за время $\mathcal{O}(B)$ согласно (5.2). Покажем, что матрица A_i полностью монотонна по столбцам, тогда согласно утверждению 3 алгоритм из [14] найдет минимальный элемент в каждом ее столбце суммарно за время $\mathcal{O}(B)$.

Лемма 7. *Для каждого* $1 \leq i \leq m$ $B \times B$*-матрица* A_i*, определенная согласно* (5.3)*, полностью монотонна по столбцам.*

Д о к а з а т е л ь с т в о. Сначала покажем, что функции w_i вогнуты, т. е. для каждого $i = 1, \ldots, m$ и любых $1 \leq k_0 < k_1 \leq j_0 < j_1 \leq B$

$$w_i(k_0, j_0) + w_i(k_1, j_1) \leq w_i(k_0, j_1) + w_i(k_1, j_0). \tag{5.4}$$

Действительно, по определению $w_i(k, j) = \sum_{t=k+1}^{j} g_{it} b_t$, и тогда

$$w_i(k_0, j_1) - w_i(k_0, j_0) = \sum_{t=k_0+1}^{j_1} g_{it} b_t - \sum_{t=k_0+1}^{j_0} g_{it} b_t = \sum_{t=j_0+1}^{j_1} g_{it} b_t$$

$$= \sum_{t=k_1+1}^{j_1} g_{it} b_t - \sum_{t=k_1+1}^{j_0} g_{it} b_t = w_i(k_1, j_1) - w_i(k_1, j_0).$$

Назовем элемент матрицы A_i *серым*, если его значение не меньше H, и *белым* иначе. Таким образом, серыми являются (см. рис. 2) элементы, лежащие не выше главной диагонали матрицы, элементы верхнего правого угла ($k < j - a_i$) и элементы, для которых $S(i-1, k) \geq H$, соответствующие случаю, когда для обслуживания первых k единиц спроса недостаточно объемов производства первых $(i-1)$ предприятий, что эквивалентно $k > \sum_{s=1}^{i-1} a_s$.

Предположим, что матрица A_i, заданная (5.3), не полностью монотонна по столбцам, т. е. существуют индексы $k_0 < k_1$ и $j_0 < j_1$ такие, что

$$A_i(k_0, j_0) > A_i(k_1, j_0) \;\text{ и }\; A_i(k_1, j_1) \geq A_i(k_0, j_1). \tag{5.5}$$

Рассмотрим четыре возможных случая.

С л у ч а й 1: $k_1 \geq j_0$. Заметим, что $H + B \geq A_i(k, j)$ для любых $1 \leq k, j \leq B$ и любых $i = 1, \ldots, m$. Тогда $A_i(k_1, j_0) = H + B \geq A_i(k_0, j_0)$, что противоречит (5.5).

С л у ч а й 2: $k_0 < k_1 < j_0 < j_1$ и $S(i-1, k_1) \geq H$. Снова получим $A_i(k_1, j_0) = H + B \geq A_i(k_0, j_0)$, что противоречит (5.5).

С л у ч а й 3: $k_0 < k_1 < j_0 < j_1$, $S(i-1, k_1) < H$, и элемент $A_i(k_0, j_1)$ серый. В этом случае по определению (5.3) $A_i(k_0, j_1) = H + (B - k_0) > H + (B - k_1) \geq A_i(k_1, j_1)$, что противоречит (5.5).

С л у ч а й 4: $k_0 < k_1 < j_0 < j_1$, $S(i-1,k_1) < H$, и элемент $A_i(k_0, j_1)$ белый. Последнее означает, что $j_1 - a_i \le k_0 < j_1$, и тогда $j_0 - a_i \le j_1 - a_i \le k_0 < k_1 < j_0 < j_1$. Также заметим, что из $S(i-1,k_1) < H$ следует $S(i-1,k_0) < H$. Тогда согласно (5.3) элементы $A_i(k_0, j_0)$, $A_i(k_1, j_0)$, $A_i(k_1, j_1)$ тоже белые. А значит, сложив два неравенства из (5.5), мы получим

$$S(i-1,j_0) + w_i(k_0,j_0) + S(i-1,k_1) + w_i(k_1,j_1) = A_i(k_0,j_0) + A_i(k_1,j_1) > A_i(k_1,j_0) + A_i(k_0,j_1)$$

$$= S(i-1,j_0) + w_i(k_1,j_0) + S(i-1,j_1) + w_i(k_0,j_1),$$

откуда следует

$$w_i(k_0,j_0) + w_i(k_1,j_1) > w_i(k_1,j_0) + w_i(k_0,j_1),$$

что противоречит свойству вогнутости w_i (5.4).

Следовательно, матрица A_i полностью монотонна по столбцам.

Лемма доказана.

Теорема 4. *Задача CFLP на цепи может быть решена за время $\mathcal{O}(mB)$.*

Д о к а з а т е л ь с т в о. Если суммировать вышесказанное, получим ускоренный алгоритм для задачи CFLP на цепи следующим образом. Сначала за время $\mathcal{O}(B+m)$ сведем исходную задачу к задаче с единичными спросами. Затем вычислим суммы из леммы 6 за время $\mathcal{O}(B+m)$, что позднее позволит находить любое значение $w_i(k,j)$ за $\mathcal{O}(1)$.

Далее, последовательно для каждого $i = 1, \ldots, m$ находим i-ю строку таблицы S динамического программирования (5.2). Для этого рассматриваем $B \times B$-матрицу A_i, заданную согласно (5.3), которая по лемме 7 полностью монотонна по столбцам, а значит, применив к ней алгоритм из работы [14], за время $\mathcal{O}(B)$ найдем минимальные элементы в каждом ее столбце. Отметим, что в алгоритме из [14] не требуется непосредственно создавать и хранить $B \times B$-массив для матрицы A_i, достаточно лишь иметь доступ к любому элементу A_i за время $\mathcal{O}(1)$, что может быть сделано, поскольку проверка того, является элемент A_i серым или белым требует $\mathcal{O}(1)$ времени, и вычисление значения белого элемента с учетом леммы 6 также требует $\mathcal{O}(1)$ времени. Наконец, зная значения минимальных элементов каждого столбца матрицы A_i, можно вычислить все элементы i-й строки S за время $\mathcal{O}(B)$ ввиду (5.2).

Поскольку в таблице S всего m строк, суммарная трудоемкость предлагаемого алгоритма равна $\mathcal{O}(mB) + \mathcal{O}(B+m) = \mathcal{O}(mB)$.

Теорема доказана.

Заключение

В данной работе мы доказали, что задача UCFLP в случае, когда в каждой вершине могут одновременно находиться предприятие и клиент, $\mathcal{NP}$-трудна даже на звезде, а в случае, когда в каждой вершине графа может находиться либо предприятие, либо клиент, $\mathcal{NP}$-трудна на дереве, однако на звезде решается за линейное время.

Известно, что оба указанных выше варианта задачи UCFLP на цепи решаются за время $\mathcal{O}(m^5n^2+m^3n^3)$ [13]. Мы показали, как ускорить время работы алгоритма из [13] до $\mathcal{O}(m^2n^2)$, что завершает цепочку улучшений трудоемкости этого алгоритма: $\mathcal{O}(m^4n^2)$ в работе [3] и $\mathcal{O}(m^3n^2)$ в работе [22]. Поскольку схема динамического программирования (4.1)–(4.2) требует вычисления элементов 4-мерной таблицы Q, потенциал ускорения данного алгоритма исчерпан, и для получения более быстрых алгоритмов решения этой задачи потребуется другой подход. С другой стороны, было бы интересно изучить возможность обобщения этого алгоритма для решения задачи UCFLP, в которой в каждой вершине графа находится либо предприятие, либо клиент, на графе-гусенице, т. е. дереве, в котором все вершины находятся на расстоянии не более одного ребра от центральной цепи.

Наконец, для более общей неоднородной задачи CFLP на цепи, которая в этом случае уже является $\mathcal{NP}$-трудной, мы рассмотрели псевдополиномиальный алгоритм динамического программирования, предложенный Мирчандани и др. [28], и показали, как понизить его трудоемкость с $\mathcal{O}(mB\min\{a_{max}, B\})$ до $\mathcal{O}(mB)$.

СПИСОК ЛИТЕРАТУРЫ

1. **Агеев А.А.** Графы, матрицы и простейшая задача размещения // Управляемые системы. 1989. Вып. 29. С. 3–11.
2. **Агеев А.А.** Полиномиальный алгоритм решения задачи размещения на последовательно-параллельной сети // Управляемые системы. 1990. Вып. 30. С. 3–16.
3. **Агеев А.А., Гимади Э.Х., Курочкин А.А,** Полиномиальный алгоритм решения задачи размещения на цепи с одинаковыми производственными мощностями предприятий // Дискретный анализ и исследование операций. 2009. Т. 16, вып. 5. С. 3–18.
4. **Ахо А., Хопкрофт Дж., Ульман Дж.** Построение и анализ вычислительных алгоритмов. М.: Мир, 1979. 536 с.
5. **Береснев В.Л., Гимади Э.Х., Дементьев В.Т.** Экстремальные задачи стандартизации. Новосибирск: Наука, 1978. 333 с.
6. **Вознюк И.П.** Задача размещения на сети с ограниченными пропускными способностями коммуникаций // Дискретный анализ и исследование операций. Сер. 2. 1999. Т. 6, вып. 1. С. 3–11.
7. **Вознюк И.П.** Задача размещения пунктов производства на два-дереве с ограниченными пропускными способностями коммуникаций // Дискретный анализ и исследование операций. Сер. 2. 2000. Т. 7, вып. 1. С. 3–8.
8. **Гимади Э.Х.** Эффективный алгоритм размещения с областями обслуживания, связными относительно ациклической сети // Управляемые системы. Вып. 23. Новосибирск, 1983. С. 12–23.
9. **Гимади Э.Х.** Точный алгоритм решения внешнепланарной задачи размещения с улучшенной временной сложностью // Тр. Ин-та математики и механики УрО РАН. 2017. Т. 23, вып. 3. С. 74–81.
10. **Трубин В.А.** Эффективный алгоритм решения задачи размещения на сети в форме дерева // Докл. АН СССР. 1976. Т. 231, вып. 3. С. 547–550.
11. **Ageev A.A.** A Criterion of polynomial-time solvability for the network location problem // Integer Programming and Combinatorial Optimization: Proc. IPCO II Conf. Campus Printing. Pittsburg: Carnegi Mellon University, 1992. P. 237–245.
12. **Ageev A.A.** Complexity of the network median problem on planar grids // Siberian Adv. Math. 1995. Vol. 5. P. 1–9.
13. **Ageev A.A.** A polynomial-time algorithm for the facility location problem with uniform hard capacities on path graph // Proc. of the 2-nd Intern. Workshop Discrete Optimization Methods in Production and Logistics (DOM'2004). Omsk, 2004. P. 28–32.
14. **Aggarwal A., Klawe M.M., Moran S., Shor P., Wilber R.** Geometric applications of a matrix searching algorithm // Algorthmica. 1987. Vol. 2. P. 195–208. doi: 10.1007/BF01840359 .
15. **Aggarwal A., Louis A., Bansal M. et al.** A 3-approximation algorithm for the facility location problem with uniform capacities // Math. Programming. 2013. Vol. 141. P. 527–547. doi: 10.1007/s10107-012-0565-4 .
16. **Bansal M., Garg N., Gupta N.** A 5-approximation algorithm for capacitated facility location // Algorithms — ESA 2012 / eds. L. Epstein, P. Ferragina. 2012. P. 133–144. (Ser. Lecture Notes in Computer Science; vol. 7501). doi: 10.1007/978-3-642-33090-2_13 .
17. **Bilde O., Krarup J.**, Sharp lower bounds and efficient algorithms for the simple plant location problem // Annals Discret. Math. 1977. Vol. 1. P. 79–97. doi: 10.1016/S0167-5060(08)70728-3 .
18. **Billionet A., Costa M.-C.** Solving the uncapacitated plant location problem on trees // Discret. Appl. Math. 1994. Vol. 49, iss. 1–3. P. 51–59.
19. **Blum M., Floyd R.W., Pratt V., Rivest R.L., Tarjan E.** Time bounds for selection // J. of Computer and System Sciences. 1973. Vol. 7, no. 4. P. 448–461. doi: 10.1016/S0022-0000(73)80033-9 .
20. **Cornuéjols G., Nemhauser G.L., Wolsey L.A.** The uncapacitated facility location problem // Discret. Location Theory. NY: Wiley, 1990. P. 119–171.
21. **Garey M.R., Johnson D.S.** Computers and intractability: A guide to the theory of NP-completeness. San Francisco, 1990. 338 p. doi: 10.5555/574848 .

22. **Gimadi E.Kh., Kurochkina A.A.** Time complexity of the Ageev's algorithm to solve the uniform hard capacities facility location problem // Optimization and Applications — 9th Internat. Conf. (OPTIMA 2018). Communications in Computer and Information Science. 2019. Vol. 974. doi: 10.1007/978-3-030-10934-9_9.
23. **Gimadi E.Kh., Tsidulko O.Yu.** On some efficiently solvable classes of the network facility location problem with constraints on the capacities of communication lines // Proc. Steklov Institute Math. 2021. Vol. 313, suppl. 1. P. 1–15. doi: 10.1134/S0081543821030081.
24. **Granot D., Skorin-Kapov D.** On some optimization problems on k-trees and partial k-trees // Discret. Appl. Math. 1994, Vol. 48, no. 2. P. 129–145. doi: 10.1016/0166-218X(92)00122-3.
25. **Hassin R., Tamir A.** Efficient algorithms for optimization and selection on series-parallel graphs // SIAM J. Alg. Disc. Meth. 1986. Vol. 7. P. 379–389. doi: 10.1137/0607043.
26. **Hassin R., Tamir A.** Improved complexity bounds for location problems on the real line // Operations Research Letters. 1991. Vol. 10, iss. 7. P. 395–402. doi: 10.1016/0167-6377(91)90041-M.
27. **Laporte G., Nickel S., Saldanha da Gama F.** Location science. Switzerland: Springer Internat. Publ., 2015. 644 p. doi: 10.1007/978-3-319-13111-5_1.
28. **Mirchandani P., Kohli R., Tamir A.** Capacitated location problems on a line // Transportation Science. 1996. Vol. 30, iss. 1. P. 75–80. doi: 10.1287/trsc.30.1.75.
29. **Pál M., Tardos E., Wexler T.** Facility location with hard capacities // Proc. 42nd IEEE Symposium on Foundations of Computer Science (FOCS). 2001. P. 329–338. doi: 10.1109/SFCS.2001.959907.
30. **Shah D.** An unified limited column generation approach for facility location problem on trees // Ann. Oper. Research. 1999. Vol. 87. P. 363–382.
31. **Shah R., Farach-Colton V.** Undiscretized dynamic programming: faster algorithms for facility location and related problems on trees // Proc. 13th Ann. ACM-SIAM Symp. on discrete algorithms (San Francisco). California: SODA, 2002. P. 108–115. doi: 10.5555/545381.545395.

Поступила 28.04.2022
После доработки 16.05.2022
Принята к публикации 20.05.2022

Агеев Александр Александрович
канд. физ.-мат. наук, старший науч. сотрудник
Институт математики им. С. Л. Соболева СО РАН
г. Новосибирск
e-mail: ageev@math.nsc.ru

Гимади Эдуард Хайрутдинович
д-р физ.-мат. наук, профессор
главный научн. сотрудник
Институт математики им. С. Л. Соболева СО РАН
г. Новосибирск
e-mail: gimadi@math.nsc.ru

Цидулко Оксана Юрьевна
канд. физ.-мат. наук, старший науч. сотрудник
Институт математики им. С. Л. Соболева СО РАН
г. Новосибирск
e-mail: tsidulko@math.nsc.ru

Штепа Александр Александрович
аспирант
Новосибирский национальный исследовательский государственный университет (НГУ)
г. Новосибирск
e-mail: shoomath@gmail.com

REFERENCES

1. Ageev A.A. Graphs, matrices and an elementary location problem. *Upravliaemie Systemy*, 1989, no. 29, pp. 3–10 (in Russian).
2. Ageev A.A. A polynomial algorithm for solving the location problem on a series-parallel network. *Upravliaemie systemy*, 1990, no. 30, pp. 3–16 (in Russian).
3. Ageev A.A., Gimadi E.K., Kurochkin A.A. A polynomial algorithm for solving the facility location problem on a chain network with identical plant production capacities. *Diskretn. Anal. Issled. Oper.*, 2009, vol. 16, no. 5, pp. 3–18 (in Russian).
4. Aho A., Hopcroft J., Ullman J. *The design and analysis of computer algorithms*. Reading, Mass.: Addison-Wesley, 1974, 470 p. ISBN: 0201000296 . Translated to Russian under the title *Postroenie i analiz vychislitel'nykh algoritmov*. Moscow: Mir Publ., 1979, 536 p.
5. Beresnev V.L., Gimadi E.Kh., Dement'ev V.T. *Ekstremal'nye zadachi standartizatsii* [Extremal standardization problems]. Novosibirsk: Nauka Publ., 1978, 333 p.
6. Voznyuk I.P. The location problem on networks with bounded communication capacities. *Diskret. Anal. Issled. Oper., Ser. 2*, 1999, vol. 6, no. 1, pp. 3–11 (in Russian).
7. Voznyuk I.P. Facility location problem on a two-tree with bounded communication capacities. *Diskret. Anal. Issled. Oper., Ser. 2*, 2000, vol. 7, no. 1, pp. 3–8 (in Russian).
8. Gimadi E.Kh. An efficient algorithm for solving plant location problem with service regions connected with respect to an acyclic network. *Upravl. Sistemy*, 1983, vol. 23, pp. 12–23 (in Russian).
9. Gimadi E.Kh. An optimal algorithm for an outerplanar facility location problem with improved time complexity. *Proc. Steklov Inst. Math.*, 2018, vol. 303, suppl. 1, pp. 87–93. doi: 10.1134/S0081543818090092 .
10. Trubin V.A. An effective algorithm for solving the distribution problem in a network in the form of a tree. *Dokl. Akad. Nauk SSSR*, 1976, vol. 231, no. 3, pp. 547–550 (in Russian).
11. Ageev A.A. A criterion of polynomial-time solvability for the network location problem. In: *Integer Programming and Combinatorial Optimization. Proc. IPCO II Conf., Campus Printing*. Pittsburg: Carnegie Mellon University, 1992, pp. 237–245.
12. Ageev A.A. Complexity of the network median problem on planar grids. *Sib. Adv. Math.*, 1995, vol. 5, no. 2, pp. 1–9.
13. Ageev A.A. A polynomial-time algorithm for the facility location problem with uniform hard capacities on path graph. In: *Proceedings of the 2-nd Intern. Workshop Discrete Optimization Methods in Production and Logistics DOM'2004*. Omsk, 2004, pp. 28–32.
14. Aggarwal A., Klawe M.M., Moran S., Shor P., Wilber R. Geometric applications of a matrix-searching algorithm. *Algorithmica*, 1987, vol. 2, pp. 195–208. doi: 10.1007/BF01840359 .
15. Aggarwal A., Louis A., Bansal M., Garg N., Gupta N., Gupta S., Jain S. A 3-approximation algorithm for the facility location problem with uniform capacities. *Mathematical Programming*, 2013, vol. 141, pp. 527–547. doi: 10.1007/s10107-012-0565-4 .
16. Bansal M., Garg N., Gupta N. A 5-approximation for capacitated facility location. In: *Algorithms – ESA 2012*, Lecture Notes in Computer Science, vol. 7501, Springer, 2012, pp. 133–144. doi: 10.1007/978-3-642-33090-2_13 .
17. Bilde O., Krarup J. Sharp lower bounds and efficient algorithms for the simple plant location problem. *Annal. Discret. Math.*, 1977, vol. 1, pp. 79–97. doi: 10.1016/S0167-5060(08)70728-3 .
18. Billionnet A., Costa M. Solving the uncapacited plant location problem on trees. *Discret. Appl. Math.*, 1994, vol. 49, no. 1–3, pp. 51–59. doi: 10.1016/0166-218X(94)90200-3 .
19. Blum M., Floyd R.W., Pratt V., Rivest R.L., Tarjan E. Time bounds for selection *J. Computer and System Sci.*, 1973, vol. 7, no. 4, pp. 448–461. doi: 10.1016/S0022-0000(73)80033-9 .
20. Cornuéjols G., Nemhauser G.L., Wolsey L.A. The uncapacitated facility location problem. In: *Discrete location theory*, Mirchandani P.B. and Francis R.L. (eds). NY: Wiley, 1990, pp. 119–171.
21. Garey M.R., Johnson D.S. Computers and intractability: A guide to the theory of NP-completeness. San Francisco: Freeman, 1990, 338 p. doi: 10.5555/574848 .
22. Gimadi E.K., Kurochkina A.A. Time complexity of the Ageev's algorithm to solve the uniform hard capacities facility location problem. In: *Optimization and Applications – 9th Internat. Conf., OPTIMA 2018*, Y. Evtushenko et al. (eds). Communications in Computer and Information Science, vol. 974, Springer, 2019, pp. 123–130. doi: 10.1007/978-3-030-10934-9_9 .

23. Gimadi E.K., Tsidulko O.Y. On some efficiently solvable classes of the network facility location problem with constraints on the capacities of communication lines. *Proc. Steklov Inst. Math.*, 2021, vol. 313, suppl. 1, pp. S58–S72. doi: 10.1134/S0081543821030081 .
24. Granot D., Skorin-Kapov D. On some optimization problems on k-trees and partial k-trees. *Discr. Appl. Math.*, 1994, vol. 48, no. 2, pp. 129–145. doi: 10.1016/0166-218X(92)00122-3 .
25. Hassin R., Tamir A. Efficient algorithms for optimization and selection on series-parallel graphs. *SIAM J. Alg. Disc. Meth.*, 1986, vol. 7, no. 3, pp. 379–389. doi: 10.1137/0607043 .
26. Hassin R., Tamir A. Improved complexity bounds for location problems on the real line. *Operations Research Letters*, 1991, vol. 10, no. 7, pp. 395–402. doi: 10.1016/0167-6377(91)90041-M .
27. Laporte G., Nickel S., Saldanha da Gama F. *Location science*. Switzerland: Springer Internat. Publ., 2015, 644 p. doi: 10.1007/978-3-319-13111-5 .
28. Mirchandani P., Kohli R., Tamir A. Capacitated location problems on a line. *Transportation Science*, 1996, vol. 30, no. 1, pp. 75–80. doi: 10.1287/trsc.30.1.75 .
29. Pál M., Tardos E., Wexler T. Facility location with hard capacities. In: *Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science (FOCS)*, 2001, pp. 329–338. doi: 10.1109/SFCS.2001.959907 .
30. Shaw D. A unified limited column generation approach for facility location problems on trees. *Ann. Oper. Research*, 1999, vol. 87, pp. 363–382. doi: 10.1023/A:1018901523519 .
31. Shah R., Farach-Colton M. Undiscretized dynamic programming: Faster algorithms for facility location and related problems on trees. *Proc. 13th Ann. ACM-SIAM Symp. on discrete algorithms, San Francisco, California: SODA*, 2002, pp. 108–115. doi: 10.1145/545381.545395 .

Received April 28, 2022
Revised May 16, 2022
Accepted May 20, 2022

Funding Agency: This work was supported within the State Assignment to the Institute of Mathematics of Siberian Branch of the Russian Academy of Sciences (project no. FWNF-2022-0019) and by the Russian Foundation for Basic Research (project no. 20-31-90091).

Alexander Alexandrovich Ageev, Cand. Sci. (Phys.-Math.), Sobolev Institute of Mathematics of the Siberian Branch of the Russian Academy of Sciences, Novosibirsk, 630090 Russia,
e-mail: ageev@math.nsc.ru .

Edward Khairutdinovich Gimadi, Dr. Phys.-Math. Sci., Prof., Sobolev Institute of Mathematics of the Siberian Branch of the Russian Academy of Sciences, Novosibirsk, 630090 Russia,
e-mail: gimadi@math.nsc.ru .

Alexandr Alexandrovich Shtepa, postgrad. student, Novosibirsk State University (NSU), Novosibirsk, 630090 Russia, e-mail: shoomath@gmail.com .

Oxana Yurievna Tsidulko, Cand. Sci. (Phys.-Math.), Sobolev Institute of Mathematics of the Siberian Branch of the Russian Academy of Sciences, Novosibirsk, 630090 Russia,
e-mail: tsidulko@math.nsc.ru .

Cite this article as: A. A. Ageev, E. Kh. Gimadi, O. Yu. Tsidulko, A. A. Shtepa. Capacitated Facility Location Problem on tree-like graphs. *Trudy Instituta Matematiki i Mekhaniki UrO RAN*, 2022, vol. 28, no. 2, pp. 24–44.