

ГИБРИДНЫЙ АЛГОРИТМ РЕШЕНИЯ ЗАДАЧИ МАРШРУТИЗАЦИИ БУРОВЫХ УСТАНОВОК

И. Н. Кулаченко^{1,2, a}, П. А. Кононова^{1,2, b}

¹ Новосибирский гос. университет,
ул. Пирогова, 2, 630090 Новосибирск, Россия

² Институт математики им. С. Л. Соболева,
пр. Акад. Коптюга, 4, 630090 Новосибирск, Россия
E-mail: ^asoge.ink@gmail.com, ^bpkononova@math.nsc.ru

Аннотация. Исследуется задача маршрутизации буровых установок. Известно множество объектов, требующих изыскательских работ, и временное окно, т. е. период, в который необходимо успеть провести работы. На одном объекте может работать несколько установок, в этом случае работы будут проведены быстрее. Необходимо определить маршруты и график работ буровых установок на объектах так, чтобы все работы были выполнены вовремя, а суммарное время переезда было минимальным.

Для этой новой задачи составлена модель задачи смешанного целочисленного линейного программирования (СЦЛП). Для поиска допустимого решения используется метаэвристика поиска с чередующимися окрестностями. Алгоритм также включает в себя решение подзадачи СЦЛП для перераспределения работ на объектах. Полученный метод сочетает в себе достоинства как точных, так и эвристических подходов. Представлены результаты сравнения разработанного алгоритма с Gurobi и альтернативными схемами поиска с чередующимися окрестностями. Табл. 3, ил. 1, библиогр. 30.

Ключевые слова: транспортное средство неограниченной грузоподъёмности, метаэвристика, разделяемое обслуживание, временные окна.

Введение

В промышленности нередко возникают задачи, требующие составления маршрутов для обслуживания некоторого множества объектов, для которых необходимо выполнить независимые времязатратные операции.

Исследование выполнено в рамках государственного задания ИМ СО РАН (проект № 0314-2019-0014).

Распространённым ограничением для задач маршрутизации является введение временных окон для всех или части объектов [1–4]. Для обслуживания всех объектов в соответствии с желаемыми временными окнами может потребоваться совместное обслуживание некоторых объектов несколькими машинами.

В работе рассматривается задача маршрутизации, возникающая при планировании буровых работ, необходимых для газо- и нефтедобывающей промышленности. Задано множество объектов, на которых необходимо провести работы, и множество буровых установок, способных произвести эти работы. Для каждого объекта известно количество скважин, которые необходимо пробурить, и временной период, в который нужно успеть выполнить все работы. Считается, что работы по бурению могут начинаться на следующий день после приезда буровой установки на объект, но не раньше начала временного периода, в который необходимо выполнить работы на объекте. Все скважины на одном объекте имеют одинаковое время бурения. Работы на одном объекте могут выполняться несколькими буровыми установками независимо, для этого необходимо распределить всё множество скважин между этими буровыми установками. Работы по бурению одной скважины могут производиться только одной буровой установкой, т. е. на объект не может быть назначено буровых установок больше, чем количество скважин на этом объекте. Задано начальное положение всех буровых установок, а также время перемещения, в днях, между любыми объектами. Заметим, что буровые установки могут быть разными. Они могут отличаться как по скорости бурения, так и по скорости перемещения между объектами. Кроме того, некоторые объекты могут быть вне зоны досягаемости каких-то буровых установок (это может быть связано с географической удалённостью объектов от обычных дорог). В конце планового периода буровым установкам не нужно возвращаться в депо, они могут оставаться в произвольном месте. Необходимо определить маршрут каждой буровой установки и распределение по числу обслуживаемых скважин на каждом из объектов так, чтобы все работы были выполнены в срок, а суммарное время перемещений было минимальным. Ранее нами уже была рассмотрена задача с одинаковым для всех объектов временем бурения и идентичными буровыми установками [5].

Описанную задачу можно отнести к классу задач маршрутизации транспортных средств (Vehicle Routing Problem, VRP) [6, 7]. Близкой по смыслу к рассматриваемой является задача маршрутизации транспортных средств с разделёнными поставками и временными окнами (Split Delivery VRP with Time Windows, SDVRPTW). В этой задаче парку транспортных средств (ТС) необходимо обслужить некоторое множество клиентов, требующих посещения в течение заданного временного

окна, но в отличие от классической задачи маршрутизации транспортных средств один клиент может быть обслужен несколькими ТС [3, 8]. В таких задачах необходимость в совместном обслуживании обычно связана с ограниченной грузоподъемностью ТС, а не временными ограничениями. В рассматриваемой же нами задаче ТС имеют неограниченную грузоподъемность, что встречается редко в задачах маршрутизации транспортных средств [9]. Обычно грузоподъемность не учитывается, если размер груза является пренебрежимо малым [10, 11]. Задачи, где ТС начинают свои маршруты из разных депо, а не из единого, называют Multiple Depot VRP, или MDVRP [12]. То, что ТС не должны возвращаться в депо после окончания всех работ, позволяет отнести задачу к классу открытых задач маршрутизации (Open VRP, OVRP) [13]. В [14] рассматривается задача открытой маршрутизации ТС с неограниченной грузоподъемностью для клиентов с разделяемым временем обслуживания. Однако в ней не рассматриваются временные окна и в качестве критерия оптимизации используется максимальное время окончания обслуживания.

В [15, 16] рассматриваются задачи маршрутизации мобильных установок для ремонта скважин (Workover Rig Routing Problem, WRRP). В этой задаче имеется множество нефункционирующих скважин, ожидающих проведения некоторого вида обслуживания, например очистки или работ по восстановлению. Для выполняющих эти работы мобильных установок требуется составить расписание обслуживания скважин так, чтобы минимизировать потери прибыли от простоя скважин.

Несмотря на развитие аппаратных средств и улучшение точных методов решения, для большинства высокоразмерных примеров задач маршрутизации с временными окнами всё ещё не получено оптимальных решений [2, 4, 6, 17], поэтому для этих задач обычно используют эвристические алгоритмы. Гибридные метаэвристики, в особенности матэвристики, активно исследуются в последние годы благодаря многообещающим результатам [6, 18–20].

В данной статье представлен матэвристический алгоритм, сочетающий в себе поиск с чередующимися окрестностями (Variable Neighborhood Search, VNS) и методы смешанного целочисленного линейного программирования (СЦЛП) [21–23]. Алгоритм VNS используется для поиска лучшего набора маршрутов для ТС при заданном распределении работ по скважинам, а СЦЛП используется для поиска лучшего распределения работ по скважинам при заданных маршрутах посещений.

Статья организована следующим образом. В разд. 1 представлены математические модели для общей задачи и распределительной подзадачи. Разд. 2 посвящён описанию алгоритма локального поиска и набора

окрестностей, используемых в нём. Результаты численных экспериментов представлены в разд. 3. В последнем разделе приводятся заключения и направления дальнейших исследований.

1. Математическая модель

1.1. Математическая модель. Пусть K — множество транспортных средств (ТС), каждое из которых изначально расположено в точке v_k , начальные положения назовём депо. Определим полный ориентированный граф $G = (\mathcal{V}, \mathcal{A})$, где \mathcal{V} — множество вершин, состоящее из множества депо ТС $\mathcal{V}^{\text{DEP}} = \{v_1, v_2, \dots, v_{|K|}\}$ и множества объектов $\mathcal{V}^{\text{CST}} = \{v_{|K|+1}, v_{|K|+2}, \dots, v_{|K|+n}\}$, а \mathcal{A} — множество дуг. Для каждого объекта $i \in \mathcal{V}^{\text{CST}}$ задано количество скважин w_i , которое необходимо пробурить на нём. Все работы на объекте i необходимо выполнить во временном интервале $[e_i, l_i]$. Работы на объекте i нельзя начинать до e_i , даже если ТС уже прибыло. Определим горизонт планирования как $H = \max_{i \in \mathcal{V}^{\text{CST}}} l_i$. Бурение любой скважины на объекте i ТС k требует d_{ik} дней. Для $i \in \mathcal{V}^{\text{DEP}}$ будем считать $d_{ik} = 0$.

Дуги $(i, j) \in \mathcal{A}$ задают время перемещения t_{ijk} от i до j ТС k в днях. Если ТС k не может переместиться от i до j , то зададим $t_{ijk} = H$. Поскольку в конце планового периода ТС не должны возвращаться в депо, а могут оставаться на последнем объекте, зададим нулевыми времена перемещения от всех объектов до всех депо $t_{i,v_k,k} = 0$, $i \in \mathcal{V}^{\text{CST}}$, $k \in K$, что позволит рассматривать модель с кольцевыми маршрутами.

Представим переменные задачи:

$$x_{ijk} = \begin{cases} 1, & \text{если ТС } k \in K \text{ едет по дуге } (i, j) \in \mathcal{A}, \\ 0 & \text{в противном случае,} \end{cases}$$

$$y_{ik} = \begin{cases} 1, & \text{если ТС } k \in K \text{ посещает вершину } i \in \mathcal{V}, \\ 0 & \text{в противном случае.} \end{cases}$$

Переменная s_{ik} , $i \in \mathcal{V}$, $k \in K$, задаёт время начала работ ТС k на объекте $i \in \mathcal{V}^{\text{CST}}$, а если $i \in \mathcal{V}^{\text{DEP}}$, то $s_{ik} = 0$ для собственного депо и не определено для остальных депо $v_i \neq v_k$.

Переменная $w_{ik} \in \mathbb{Z}_{\geq 0}$, $i \in \mathcal{V}$, $k \in K$, задаёт общее число скважин, которое ТС k должно пробурить на объекте i или 0, если $i \in \mathcal{V}^{\text{DEP}}$.

Введём обозначение $\mathcal{V}_k = \mathcal{V}^{\text{CST}} \cup \{v_k\}$ и сформулируем задачу в виде задачи смешанного целочисленного линейного программирования:

$$\min \sum_{k \in K} \sum_{i \in \mathcal{V}_k} \sum_{j \in \mathcal{V}_k} t_{ijk} x_{ijk} \quad (1)$$

при ограничениях

$$y_{ik} = \begin{cases} 1, & \text{если } v_i = v_k, \\ 0, & \text{если } v_i \neq v_k, \end{cases} \quad i \in \mathcal{V}^{\text{DEP}}, k \in K, \quad (2)$$

$$s_{v_k, k} = 0, \quad k \in K, \quad (3)$$

$$w_{v_k, k} = 0, \quad k \in K, \quad (4)$$

$$\sum_{j \in \mathcal{V}_k} x_{ijk} = \sum_{j \in \mathcal{V}_k} x_{jik} = y_{ik}, \quad i \in \mathcal{V}_k, k \in K, \quad (5)$$

$$\sum_{k \in K} w_{ik} = w_i, \quad i \in \mathcal{V}^{\text{CST}}, \quad (6)$$

$$w_{ik} \geq y_{ik}, \quad i \in \mathcal{V}^{\text{CST}}, k \in K, \quad (7)$$

$$w_{ik} \leq w_i y_{ik}, \quad i \in \mathcal{V}^{\text{CST}}, k \in K, \quad (8)$$

$$s_{ik} \geq e_i, \quad i \in \mathcal{V}^{\text{CST}}, k \in K, \quad (9)$$

$$s_{ik} + d_{ik} w_{ik} \leq l_i, \quad i \in \mathcal{V}^{\text{CST}}, k \in K, \quad (10)$$

$$s_{ik} + d_{ik} w_{ik} + t_{ijk} - s_{jk} \leq M(1 - x_{ijk}), \quad i \in \mathcal{V}_k, j \in \mathcal{V}^{\text{CST}}, k \in K, \quad (11)$$

$$x_{ijk}, y_{ik} \in \{0, 1\}, \quad s_{ik} \geq 0, \quad w_{ik} \in \mathbb{Z}_{\geq 0}, \quad i, j \in \mathcal{V}_k, k \in K. \quad (12)$$

Целевая функция (1) минимизирует суммарное время перемещений для всех ТС за весь плановый период. Неравенства (2)–(4) расставляют ТС по депо и запрещают проезд через другие депо. Ограничения (5) задают кольцевые маршруты из депо. Неравенства (6) гарантируют, что на всех объектах будут выполнены буровые работы в полном объёме. Вспомогательные неравенства (7) показывают, что если ТС посещает объект, то бурит на нём хотя бы одну скважину, а неравенства (8) запрещают бурить скважины на объекте, который ТС не посещает. Неравенства (9), (10) гарантируют, что все работы по бурению должны уложиться во временное окно. Неравенства (11), где константы M достаточно большие, устанавливают время начала буровых работ на объектах и запрещают одному ТС несколько раз возвращаться на объект. Ограничения (12) определяют типы переменных.

Заметим, что задача (1)–(12) может не иметь допустимых решений из-за ограниченного количества ТС и временных окон, поэтому мы ослабим ограничение (10) и внесём его в целевую функцию со штрафом $\lambda\gamma$, где $\lambda \geq 0$, $\gamma \geq 0$. Вместо задачи (1)–(12) будем рассматривать новую задачу с целевой функцией

$$L(x, \tau) = \min \sum_{k \in K} \sum_{i \in \mathcal{V}_k} \left(\sum_{j \in \mathcal{V}_k} t_{ijk} x_{ijk} + \lambda \gamma \tau_{ik} \right) + \sum_{i \in \mathcal{V}_k} \gamma \tau_{\max}^i, \quad (13)$$

ограничениями (2)–(9), (11)–(12) и новыми дополнительными переменными $\tau_{ik} \geq 0$, $\tau_{\max}^i \geq 0$ со следующими ограничениями:

$$\tau_{ik} \geq s_{ik} + d_{ik} w_{ik} - l_i, \quad i \in \mathcal{V}^{\text{CST}}, k \in K, \quad (14)$$

$$\tau_{\max}^i \geq \tau_{ik}, \quad i \in \mathcal{V}^{\text{CST}}, k \in K. \quad (15)$$

Основной вклад в штраф в (13) вносит слагаемое $\sum_{i \in \mathcal{V}_k} \gamma \tau_{\max}^i$, а слагаемое $\sum_{i \in \mathcal{V}_k, k \in K} \lambda \gamma \tau_{ik}$ необходимо для более плавного спуска к допустимому решению задачи (1)–(12).

Новая упрощённая задача (2)–(9), (11)–(15) всегда имеет допустимое решение, и методами локального поиска [24] можно пытаться найти допустимое решение без нарушения ограничения на временные окна. Такое решение будет допустимым и для исходной задачи.

1.2. Математическая модель для распределительной задачи.

Рассмотрим подзадачу нашей задачи. Пусть для каждого ТС уже задан порядок обслуживания объектов, при этом каждый объект назначен одному или нескольким ТС. Необходимо распределить, сколько скважин должно пробурить каждое ТС на объекте, чтобы все работы были выполнены в полном объёме с учётом минимизации целевой функции (13). При построении маршрута движения ТС выбираются только те объекты, которые назначены данному ТС, и движение происходит в заданном порядке обслуживания. Подчеркнём, что объект, на котором ТС не производит работы, в маршрут не попадает, даже если этот объект был задан в порядке обслуживания для данного ТС (это возможно, если все работы на данном объекте выполнили другие ТС). Докажем, что при таком упрощении задача остаётся NP-трудной.

Теорема 1. *Задача распределения скважин при заданных порядках обслуживания NP-трудна.*

ДОКАЗАТЕЛЬСТВО. Рассмотрим частный случай известной NP-полной задачи о разбиении, который также является NP-полным [25]. Пусть задано конечное множество A из элементов a_1, a_2, \dots, a_{2n} и их веса $s(a_i) \in \mathbb{Z}^+$. Вопрос: существует ли такое подмножество $A' \subset A$, которое содержит ровно по одному элементу из каждой пары a_{2i-1}, a_{2i} и

$$\sum_{a \in A'} s(a) = \sum_{a \in A \setminus A'} s(a)?$$

Построим пример задачи распределения скважин при заданном порядке обслуживания. Каждому элементу a_i из множества A сопоставим

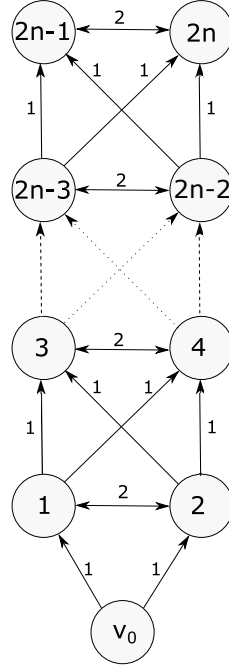


Рис. 1. Построенный пример

объект i с количеством скважин $w_i = s(a_i) \geq 1$. Пусть имеется ровно два ТС и одинаковый порядок обслуживания $v_0, i_1, i_2, \dots, i_{2n}$. Времена переезда между всеми объектами равны 1, кроме времени переезда между объектами в одной паре $2i-1$ и $2i$ — они равны 2. Для «фиктивного» возвращения в депо время переездов от объектов i_{2n-1} и i_{2n} до v_0 зададим равным 0. Зададим время бурения $d_{ik} = 1$ для всех $i \in \{i_1, i_2, \dots, i_{2n}\}$, $k \in \{1, 2\}$. Для всех объектов все работы должны быть выполнены в интервале $[0, S/2 + n]$, где $S = \sum_{a \in A} s(a)$. Необходимо узнать, существует ли

допустимое решение со значением целевой функции не более чем $2n$. Такое значение может существовать только в решении, в котором никакое ТС не обслуживает объекты из одной пары и оба ТС обслуживают разные объекты. Если такое допустимое решение существует, то существует и решение соответствующей задачи о разбиении, в котором в A' попадает ровно по одному элементу из каждой пары. Теорема 1 доказана.

Приведём математическую модель задачи распределения работ на объектах. Пусть задан порядок I_k обслуживания объектов для каждого ТС k и можно говорить, что $i \prec_{I_k} j$, если $i, j \in I_k$ и ТС k не может

обслуживать объект i когда-либо после объекта j . Зададим x_{ijk} и y_{ik} в соответствии с порядками I_k и добавленными депо. Теперь x_{ijk} и y_{ik} не являются переменными, но и не определяют полностью маршруты ТС. На самом деле $y_{ik} = 0$ означает, что объект i не посещается ТС k . Даже если $y_{ik} = 1$, то ТС k может не посещать объект i в том случае, если ему не досталось работы на этом объекте, поэтому введём новые переменные для точного задания маршрута: переменные x'_{ijk} , определённые только для $i \prec_{I_k} j$, и y'_{ik} такие, что

$$y'_{ik} \leq y_{ik}, \quad i \in \mathcal{V}_k, k \in K. \quad (16)$$

Остальные переменные имеют тот же смысл, что и раньше. В итоге получим задачу меньшей размерности:

$$\min \sum_{k \in K} \sum_{i \in I_k} \left(\sum_{j \in I_k, i \prec_{I_k} j} t_{ijk} x'_{ijk} + \lambda \gamma \tau_{ik} \right) + \sum_{i \in \mathcal{V}_k} \gamma \tau_{\max}^i, \quad (17)$$

$$\sum_{k \in K} w_{ik} = w_i, \quad i \in \mathcal{V}^{\text{CST}}, \quad (18)$$

$$w_{ik} \geq y'_{ik}, \quad k \in K, i \in \mathcal{V}^{\text{CST}}, \quad (19)$$

$$w_{ik} \leq w_i y'_{ik}, \quad k \in K, i \in \mathcal{V}^{\text{CST}}, \quad (20)$$

$$s_{ik} \geq e_i, \quad k \in K, i \in I_k, \quad (21)$$

$$s_{ik} + d_{ik} w_{ik} \leq l_i + \tau_{ik}, \quad k \in K, i \in I_k, \quad (22)$$

$$s_{ik} + d_{ik} w_{ik} + t_{ij} - s_{jk} \leq M(1 - x'_{ijk}), \quad (23)$$

$$i \in I_k, j \in I_k \setminus \{v_k\}, i \prec_{I_k} j, k \in K,$$

$$\sum_{j \in I_k, i \prec_{I_k} j} x'_{ijk} = \sum_{j \in I_k, j \prec_{I_k} i} x'_{jik} = y'_{ik}, \quad i \in I_k, k \in K, \quad (24)$$

$$\tau_{ik} \geq s_{ik} + d_{ik} w_{ik} - l_i, \quad i \in I_k, k \in K, \quad (25)$$

$$\tau_{\max}^i \geq \tau_{ik}, \quad i \in I_k, k \in K, \quad (26)$$

$$x'_{ijk} \in \{0, 1\}, \quad s_{ik} \geq 0, \quad \tau_{ik} \geq 0, \quad i, j \in I_k, i \prec_{I_k} j, k \in K, \quad (27)$$

$$y'_{ik} \in \{0, 1\}, \quad w_{ik} \in \mathbb{Z}_{\geq 0}, \quad \tau_{\max}^i \geq 0, \quad i \in \mathcal{V}_k, k \in K. \quad (28)$$

Поскольку число переменных и ограничений в данной задаче меньше, чем в исходной задаче (2)–(9), (11)–(15), и она может быть решена за приемлемое время, будем использовать решение этой задачи смешанного целочисленного линейного программирования в локальном поиске.

2. Локальный поиск

Для поиска хорошего допустимого решения задачи используется метаэвристика, комбинирующая поиск с чередующимися окрестностями и решение распределительной задачи посредством смешанного целочисленного линейного программирования [5]. Такая комбинация метаэвристического алгоритма с методами математического программирования часто приводит к улучшению результатов [18–20, 22], а VNS хорошо зарекомендовал себя в решении многих задач маршрутизации [11, 15, 23].

Пусть $\mu_k = (\mu_0^k, \mu_1^k, \dots, \mu_{u_k}^k)$ — последовательность посещений для маршрута ТС $k \in K$, где $\mu_u^k, u \in \{1, \dots, u_k\}$, — это u -й посещенный объект, $\mu_0^k = v_k$. И пусть $\tilde{\mu}_k$ будет последовательностью той же длины, элементы $\tilde{\mu}_u^k$ которой равны числу скважин на этом объекте, которые бурит ТС k , $\tilde{\mu}_0^k = 0$. Тогда маршрут каждого ТС $k \in K$ может быть представлен как последовательность $\sigma_k = (\sigma_0^k, \sigma_1^k, \dots, \sigma_{u_k}^k)$, где $\sigma_u^k = (\mu_u^k, \tilde{\mu}_u^k)$.

Любое допустимое решение σ легко может быть сопоставлено с набором последовательностей $\{\sigma_1, \dots, \sigma_{|K|}\}$. Теперь определим окрестности для допустимого решения σ . Окрестности N_1 – N_3 и N_6 нацелены на улучшение маршрутов, в то время как N_4 – N_5 нацелены на улучшение распределения по скважинам. Эти окрестности, за исключением N_4 и N_7 , уже были представлены в [5].

- $N_1(\sigma)$ (relocate): переместить одно или два последовательных посещения объектов либо из одного маршрута в другой, либо внутри одного маршрута на новое место.

- $N_2(\sigma)$ (exchange): заменить друг на друга две непересекающиеся подпоследовательности $(\sigma_i^k, \dots, \sigma_j^k)$ и $(\sigma_{i'}^{k'}, \dots, \sigma_{j'}^{k'})$, содержащие 1 или 2 посещения объектов.

- $N_3(\sigma)$ (cross): заменить друг на друга две подпоследовательности $(\sigma_i^k, \dots, \sigma_{u_k}^k)$, $(\sigma_{i'}^{k'}, \dots, \sigma_{u_{k'}}^{k'})$, являющиеся концевыми для двух различных маршрутов. Одна из этих подпоследовательностей может быть пустой.

- $N_4(\sigma)$ (merge): объединить два посещения одного объекта разными ТС в одно посещение каким-либо одним из данных ТС.

- $N_5(\sigma)$ (split): разделить одно посещение объекта на посещение в той же позиции маршрута, но с меньшим числом обслуживаемых скважин (в том числе 0), и новое посещение в маршруте другого ТС, которое до этого не посещал данный объект.

- $N_6(\sigma)$ (cross-exchange): заменить друг на друга две непересекающиеся подпоследовательности $(\sigma_i^k, \dots, \sigma_j^k)$ и $(\sigma_{i'}^{k'}, \dots, \sigma_{j'}^{k'})$, содержащие r и r' посещений объектов, где $1 \leq r \leq R$, $0 \leq r' \leq R$, а R является параметром алгоритма.

• $N_7(\sigma)$ (Kernighan–Lin combined): окрестность Кернигана — Лина для окрестностей N_3 – N_6 . Идея этой окрестности схожа с поиском с запретами по ограниченному числу итераций [5, 26]. Данным четырём окрестностям сопоставляются веса $\{c_{\text{merge}}, c_{\text{split}}, c_{\text{cross}}, c_{\text{cross-exchange}}\}$, и на каждой итерации выбор одной из них происходит в соответствии с распределением, по которому вероятность окрестности быть выбранной равна весу этой окрестности, делённому на сумму всех весов. Обратные изменения хранятся в списке запретов в течение $\mu \in [\mu_1, \mu_2]$ итераций, где μ_1, μ_2 являются параметрами алгоритма.

Для всех окрестностей получающиеся маршруты не должны содержать посещение одного объекта более одного раза. Одна или две подпоследовательности в N_1 – N_3 и N_6 могут быть перевёрнуты, если они содержат ровно два посещения.

Так как описанные окрестности большие, а время, затрачиваемое на процедуру пересчёта целевой функции со штрафами за опоздания, существенно, для ускорения работы алгоритма применяются средства рандомизации и эффективной оценки значения целевой функции из [4, 27], которые описаны в нашей предыдущей работе [5].

2.1. Построение начального решения. Для построения начального решения σ используется жадная эвристика, состоящая из трёх шагов: распределение работ по объектам для ТС, построение маршрутов для данного распределения и распределение работ по бурению скважин между ТС. Сначала каждый объект $i \in \mathcal{V}^{\text{CST}}$ назначается каждому из $\lceil 2 \cdot \max_{k \in K} d_{ik} \cdot w_i / |l_i - e_i| \rceil$ ближайших ТС. Затем строятся маршруты с использованием эвристики ближайшего соседа относительно меры близости (29) между вершинами $i, j \in \mathcal{V}_k$, относящимися к маршруту ТС $k \in K$ [28],

$$\rho(i, j, k) = t_{ijk} + \gamma^E \max\{e_j - s_{ik} - d_{ik}w_{ik} - t_{ijk}, 0\} + \gamma^T(l_j - s_{ik} - d_{ik}w_{ik} - t_{ijk} - d_{jk}w_{jk}). \quad (29)$$

Эта мера учитывает не только расстояния, но и временные окна. Коэффициенты γ^E и γ^T , отвечающие за вклад времени ожидания и разницы между окончанием работ и окончанием временного окна, устанавливаются равными 1 и 0,5 соответственно. Такие значения были выбраны в силу избыточного распределения работ по объектам для ТС. Числа обслуживаемых скважин w_{ik} устанавливаются равными для каждого ТС, обслуживающего объект i . После построения маршрутов приближённо решается распределительная задача (16)–(28).

2.2. Общий поиск с чередующимися окрестностями. Схема VNS состоит из трёх основных шагов: процедура встряски, процедура

локального улучшения и шаг по принятию решения и смене окрестности [22]. В целях различимости окрестностей, применяемых в процедурах встряски и локального улучшения, мы используем для них два различных обозначения: \mathcal{N} и N соответственно. Псевдокод алгоритма общего поиска с чередующимися окрестностями представлен ниже.

Алгоритм 1. Общий поиск с чередующимися окрестностями

```

1: function GENERAL_VNS( $\sigma, k_{\max}, l_{\max}, \mathcal{N}, N$ )
2:   while не достигнут критерий остановки do
3:      $k \leftarrow 1$ 
4:     while  $k \leq k_{\max}$  do
5:        $\sigma' \leftarrow \text{Shake}(\sigma, k, \mathcal{N})$ 
6:        $\sigma'' \leftarrow \text{VND}(\sigma', l_{\max}, N)$ 
7:       Neighborhood_change_sequential_SA( $\sigma, \sigma'', k$ )
8:     end while
9:   end while
10:  Оптимально решить распределительную задачу относительно
    маршрутов лучшего найденного решения
11:  return лучшее найденное решение
12: end function

```

В качестве критерия остановки (шаг 2) в экспериментах использовалось максимальное время исполнения.

Процедура встряски (шаг 5) необходима для диверсификации поиска и нацелена на предотвращение застревания в локальном оптимуме. Пусть $\mathcal{N} = \{\mathcal{N}_1, \dots, \mathcal{N}_{k_{\max}}\}$ — набор окрестностей, используемых в данной процедуре. Функция $\text{Shake}(\sigma, k, \mathcal{N})$ заключается в выборе случайного решения из окрестности $\mathcal{N}_k(\sigma)$. Значение k_{\max} является параметром алгоритма.

Алгоритм 2. Процедура встряски

```

1: function SHAKE( $\sigma, k, \mathcal{N}$ )
2:   for  $i \in \{1, \dots, k\}$  do
3:     Случайно выбрать  $j \in \{1, 2, 3\}$  с вероятностями, соответствующими весам  $\{c_1, c_2, c_3\}$ 
4:     Случайно выбрать  $\sigma' \in \mathcal{N}_k^j(\sigma)$ 
5:      $\sigma \leftarrow \sigma'$ 
6:   end for
7:   return  $\sigma$ 
8: end function

```

Окрестность $\mathcal{N}_k^1(\sigma)$ в алгоритме 2 соответствует $N_4(\sigma)$, а $\mathcal{N}_k^2(\sigma)$ соответствует cross-exchange окрестности, где две подпоследовательности, содержащие r и r' посещений, $1 \leq r \leq k$, $0 \leq r' \leq k$, меняются друг с другом местами. Окрестность $\mathcal{N}_k^3(\sigma)$ соответствует решениям, отличным от σ лишь распределением работ по бурению скважин какого-либо одного объекта. При этом значения для нового распределения могут отличаться от значений прошлого распределения максимум на k скважин.

В качестве процедуры локального улучшения используется алгоритм спуска с чередующимися окрестностями (Variable Neighborhood Descent, VND) [21, 22]. Он также включает в себя решение распределительной задачи (16)–(28) средствами математического программирования (шаг 12 алгоритма 3). Во внутреннем цикле процедуры при улучшении решения мы возвращаемся к окрестности N_1 , иначе переходим к следующей окрестности. При этом процедура завершается после того, как для некоторого решения σ ни в одной из окрестностей $N_1(\sigma), \dots, N_{l_{\max}}(\sigma)$ не было найдено решений с лучшей, чем $L(\sigma)$, целевой функцией, и решение распределительной задачи также не смогло улучшить решение σ .

Алгоритм 3. Спуск с чередующимися окрестностями

```

1: function VND( $\sigma, l_{\max}, N$ )
2:   repeat
3:      $stop \leftarrow true$ 
4:      $l \leftarrow 1$ 
5:      $\sigma' \leftarrow \sigma$ 
6:     repeat
7:        $\sigma'' \leftarrow \arg \min_{x \in N_l(\sigma)} L(x)$ 
8:       Neighborhood_change_sequential( $\sigma, \sigma'', l$ )
9:     until  $l = l_{\max}$ 
10:    if  $L(\sigma) \neq L(\sigma')$  then
11:       $\sigma' \leftarrow \sigma$ 
12:      Решить распределительную задачу относительно маршру-
        тов решения  $\sigma$ 
13:    end if
14:    if  $L(\sigma) < L(\sigma')$  then
15:       $stop \leftarrow false$ 
16:    end if
17:  until  $stop = true$ 
18:  return  $\sigma$ 
19: end function

```

Шаг по принятию решения и смене окрестности несколько отличается для алгоритмов VNS и VND. В алгоритме VND используется последовательная смена окрестностей, описанная в алгоритме 4.

Алгоритм 4. Последовательная схема смены окрестности

```

1: procedure NEIGHBORHOOD_CHANGE_SEQUENTIAL( $\sigma, \sigma', k$ )
2:   if  $L(\sigma') < L(\sigma)$  then
3:      $\sigma \leftarrow \sigma'$ 
4:      $k \leftarrow 1$ 
5:   else
6:      $k \leftarrow k + 1$ 
7:   end if
8: end procedure

```

В общей схеме VNS на шаге 7 (подробнее в алгоритме 5) допускается переход в ухудшающее решение по критерию, схожему с используемым в алгоритме имитации отжига [29]. Понижение температуры происходит по формуле $T = \alpha^{\frac{\eta}{100}} \cdot T$, где η — число итераций, прошедших с предыдущего понижения, а $\alpha \in (0, 1)$ — параметр, отвечающий за скорость понижения температуры. Дополнительное слагаемое $T_{\text{offset}} > 0$ отвечает за смещение значения температуры T в целях избежания приравнивания температуры к нулю. В наших экспериментах используется значение $T_{\text{offset}} = 0,2$. Стоит отметить, что алгоритм 5 также включает смену штрафа в случае, если не удалось улучшить результат. Штраф изменяется по следующему правилу:

$$\gamma = \begin{cases} \gamma, & \text{если } \sum_{i \in \mathcal{V}_k, k \in K} \tau_{ik} > 0 \text{ и } \gamma \geq 10, \\ \gamma\theta, & \text{если } \sum_{i \in \mathcal{V}_k, k \in K} \tau_{ik} > 0 \text{ и } \gamma < 10, \\ \gamma/\theta & \text{в противном случае,} \end{cases}$$

где θ является параметром алгоритма. Во время всей процедуры поиска запоминается решение с лучшим значением целевой функции (13) со значениями параметров $\lambda = 0$, $\gamma = 10$, и именно это решение выдаётся в качестве итогового на шаге 11 алгоритма 1. Значения для параметров выбраны такими, так как мы стремимся к решениям без нарушения временных окон.

В конце алгоритма 1 оптимально решается распределительная задача (16)–(28) для маршрутов, соответствующих лучшему найденному решению.

Алгоритм 5. Последовательная схема с имитацией отжига для смены окрестности

```

1: procedure NEIGHBORHOOD_CHANGE_SEQUENTIAL_SA( $\sigma, \sigma', k$ )
2:   Случайно выбрать  $r \in [0, 1]$ 
3:   if  $r < \exp \frac{L(\sigma) - L(\sigma')}{T + T_{\text{offset}}}$  then
4:      $\sigma \leftarrow \sigma'$ 
5:      $k \leftarrow 1$ 
6:   else
7:      $k \leftarrow k + 1$ 
8:     Обновить значение штрафа  $\gamma$ 
9:   end if
10:  Обновить значение температуры  $T$ 
11: end procedure

```

3. Численные эксперименты

Описанный алгоритм реализован на языке C++ с использованием компилятора MSVC++ 14.27 со стандартными release параметрами. Все численные эксперименты, представленные в данном разделе, проводились на компьютере с процессором Intel Core i7-9700 3,0 ГГц и 16 ГБ ОЗУ, работающем под операционной системой Microsoft Windows 10 (64-bit). Просмотр окрестностей в реализованном алгоритме выполнялся параллельно.

3.1. Тестовые примеры. Для исследования эффективности алгоритма были сгенерировано 3 набора примеров $\{S1, S2, S3\}$, отличающихся подходами к генерации значений времени бурения скважин. Для генерации координат расположения объектов и ТС использовался открытый набор данных с координатами населённых пунктов России, включающий 1804 элемента. Значения для числа скважин на объекте и временных окон генерировались так же, как в [5]. Каждому объекту назначается 30 скважин с вероятностью 1%, 20 скважин с вероятностью 5%, 10 скважин с вероятностью 20% и 5 скважин с вероятностью 74%. Горизонт планирования составляет 365 дней, и временные окна для посещения объектов генерируются равномерно по этому периоду. Ширина окна варьируется между временем, необходимым для завершения всех работ по бурению на этом объекте, и этим значением плюс небольшое случайное число. Таким образом можно получить пример с данными для 1804 объектов, и получение примеров меньшей размерности возможно через выбор случайного подмножества данного большого примера. Координаты для ТС выбираются случайно из тех, что не вошли в сгенерированный

пример. Для набора S1 все скважины считаются идентичными и требующими одинакового времени бурения, равного двум дням. Для набора S2 время бурения для скважин генерируется случайно с равной вероятностью для значений из множества $\{1, 2, 3\}$ и одинаково для каждого типа ТС. Типы для ТС, влияющие на матрицу времён перемещения и время бурения для набора S3, генерировались следующим способом. Сначала первым трём ТС назначаются типы 1, 2 и 3. Затем значение для типа выбирается случайно из множества $\{1, 2, 3\}$. Для генерации времени бурения в наборе S3 каждому объекту сопоставляется одно из значений множества $\{3, 6, 12\}$ с вероятностями 50, 40 и 10%; эта величина символизирует глубину скважин. Тогда время бурения скважин на объекте равно его глубине, делённой нацело на скорость бурения ТС. Для ТС типа 1 скорость бурения равна 2, а для типов 2 и 3 скорость равна 3. Для 495 объектов, расположенных в северо-восточной части России, допускается обслуживание только ТС типа 3, что отражается в матрице времён перемещений (t_{ijk}) , где для ТС других типов значения времени перемещения устанавливаются равным горизонту планирования — 365 дней. Таким образом, для каждого из трёх наборов примеров $\{S1, S2, S3\}$ были сгенерированы по 7 следующих различных примеров: 50 объектов и 6 ТС, 150 объектов и 15 ТС и их разновидности $\{S1', S2', S3'\}$, когда временные окна для посещения объектов уменьшаются вдвое путём уменьшения соответствующего значения наиболее позднего времени окончания работ. Для последних примеров ТС дублируются и их общее число становится равным 12 для примеров с 50 объектами и 30 для примеров со 150 объектами.

3.2. Настройка параметров. Разработанный алгоритм имеет множество параметров, от которых существенно зависят результаты. Для более эффективного подбора значений параметров алгоритма используется средство автоматической настройки параметров SMAC (v3) [30]. Для тренировочного процесса было выбрано 6 из 42 примеров с 50 объектами, по 2 для каждого набора — с начальными значениями временных окон и уменьшенными вдвое. Примеры запускались на 120 секунд. Всего в процессе настройки было рассмотрено 211 конфигураций.

В табл. 1 показаны значения параметров для начальной конфигурации и конфигурации, полученной в результате настройки. В среднем результаты для алгоритма с настроенными параметрами оказались на 2,43% лучше, чем в случае ненастроенных параметров. Сравнение производилось на примерах с 50 объектами, где каждый пример запускался 30 раз на 90 с.

В табл. 1 параметры γ и T соответствуют начальным значениям для штрафа в целевой функции L и температуры в схеме VNS. Параметры n_1 и n_2 отвечают за степень рандомизации в процессе локального

Таблица 1

Результаты настройки параметров

Тип	γ	λ	θ	T	α	R	n_1	n_2
начальная	10	0,2	1,05	50	0,992	12	0,7	0,5
настроенная	8,3	0,09	1,115	8,6	0,99	10	0,84	0,33
Тип	KL_{cfg}		l	$[\mu_1, \mu_2]$	MIPGap	k_{max}	$\{c_1, c_2, c_3\}$	
начальная	$\{1, 1, 2, 2\}$		20	$\{5, 10\}$	0,05	10	$\{1, 1, 1\}$	
настроенная	$\{1, 1, 2, 2\}$		35	$\{5, 10\}$	0,03	5	$\{1, 1, 0, 2\}$	

поиска. Для каждой из окрестностей N_1-N_6 соседи описываются парами подпоследовательностей, одна из которых может быть пустой [5]. В процессе просмотра окрестности сначала случайно выбирается первая непустая подпоследовательность, которая пропускается с вероятностью $1 - n_1$, а затем для неё подбирается наилучшая вторая подпоследовательность, где кандидаты пропускаются с вероятностью $1 - n_2$. Параметр KL_{cfg} соответствует весам $\{c_{\text{merge}}, c_{\text{split}}, c_{\text{cross}}, c_{\text{cross-exchange}}\}$, используемым в окрестности N_7 , а l отвечает за число итераций в ней. Параметр MIPGap соответствует одноимённому параметру в Gurobi, используемому для решения распределительной задачи в алгоритме VND.

3.3. Сравнение с Gurobi. Используя модель СЦЛП, можно получить оптимальное решение задачи методом ветвей и границ (решатель Gurobi). К сожалению, такой подход работает только на примерах малой размерности. В экспериментах для примеров с 50 объектами проведено сравнение результатов Gurobi (9.1.1) и разработанного гибридного алгоритма VNS (табл. 2). Для каждого примера Gurobi запускался с временным ограничением в 3 часа и параметрами `Presolve = 2`, `GomoryPasses = 0`, `Method = 0`, `MinRelNodes = 10627`, `ImproveStartTime = 8640`, найденными при помощи встроенной функции `Model.tune()`. Алгоритм VNS запускался на 90, 270 и 810 секунд с числом запусков каждого примера, равным 30, 10 и 10 соответственно. В табл. 2 показаны относительные различия (в %) результатов, полученных Gurobi и методом VNS. Жирным выделены результаты, лучшие найденные для примера, а подчёркнутые результаты оптимальны. В столбце t указано суммарное время, затраченное на переезды всеми ТС, а $\tau = \sum_{i \in V_k} \tau_{\text{max}}^i$. В столбце LB показаны отклонения между найденными Gurobi значениями для допустимого решения и нижней границы.

Табл. 2 демонстрирует стремление алгоритма к нахождению оптимальных решений с увеличением времени выполнения. Из сравнения столбцов, показывающих лучшие найденные решения при 30 запусках

на 90 с и 10 запусках на 270 с, можно также предположить, что алгоритм даёт лучшие результаты при однократном запуске на большее время, чем при многократных запусках на меньшее время и выборе лучшего результата.

Таблица 2

Результаты сравнения с Gurobi, $|\mathcal{V}^{\text{CST}}| = 50$

#	Gurobi		LB	VNS, 90 с		VNS, 270 с		VNS, 810 с	
	t	τ		сред.	мин.	сред.	мин.	сред.	мин.
S1.1	63	<u>0</u>	0	<u>0</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>0</u>
S1.2	74	<u>0</u>	0	1,45	<u>0</u>	0,14	<u>0</u>	<u>0</u>	<u>0</u>
S1.3	79	<u>0</u>	−6,33	0,89	<u>0</u>	0,13	<u>0</u>	<u>0</u>	<u>0</u>
S1.4	61	<u>0</u>	−8,20	2,25	<u>0</u>	0,66	<u>0</u>	<u>0</u>	<u>0</u>
S1.5	82	<u>0</u>	−2,44	0,21	<u>0</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>0</u>
S1.6	64	<u>0</u>	0	1,61	<u>0</u>	0,63	<u>0</u>	<u>0</u>	<u>0</u>
S1.7	66	<u>0</u>	0	0,15	<u>0</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>0</u>
S1'.1	122	<u>0</u>	−6,56	6,53	4,1	4,02	1,64	1,31	<u>0</u>
S1'.2	133	<u>0</u>	−3,76	4,61	3,01	2,78	1,5	1,65	0,75
S1'.3	147	<u>0</u>	−5,44	6,71	2,72	2,86	0,68	1,56	0,68
S1'.4	114	<u>0</u>	−11,40	5,2	2,63	2,81	1,75	1,49	0,88
S1'.5	153	<u>0</u>	−11,11	3,12	0,65	1,11	0,65	0,65	<u>0</u>
S1'.6	137	<u>0</u>	−8,03	16,2	10,22	10,22	8,03	7,45	5,11
S1'.7	146	<u>0</u>	−7,53	6,39	4,11	5,07	2,05	3,77	1,37
S2.1	87	<u>0</u>	−14,94	3,22	<u>0</u>	−0,11	−1,15	−0,8	−1,15
S2.2	72	<u>0</u>	0	1,57	<u>0</u>	0,28	<u>0</u>	<u>0</u>	<u>0</u>
S2.3	88	<u>1</u>	−4,08	4,29	1,02	2,65	1,02	0,82	<u>0</u>
S2.4	73	<u>5</u>	0	1,24	<u>0</u>	0,49	<u>0</u>	0,24	<u>0</u>
S2.5	70	<u>0</u>	0	0,33	<u>0</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>0</u>
S2.6	61	<u>0</u>	0	1,75	<u>0</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>0</u>
S2.7	59	<u>0</u>	0	1,14	<u>0</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>0</u>
S2'.1	116	<u>0</u>	−0,86	3,56	1,72	2,16	0,86	1,03	0,86
S2'.2	125	<u>2</u>	−2,76	10,12	7,59	7,45	6,21	5,38	4,14
S2'.3	140	<u>0</u>	−1,43	6,81	3,57	4	2,14	2,43	0,71
S2'.4	119	<u>3</u>	−1,34	5,79	2,68	2,55	0,67	1,81	0,67
S2'.5	130	<u>0</u>	−0,77	1,82	<u>0</u>	1	<u>0</u>	0,15	<u>0</u>
S2'.6	129	<u>0</u>	−1,55	12,53	6,2	6,43	3,88	4,34	2,33
S2'.7	131	<u>3</u>	−1,86	4,33	3,11	3,17	2,48	2,67	1,86
S3.1	61	<u>0</u>	0	1,26	<u>0</u>	0,33	<u>0</u>	<u>0</u>	<u>0</u>
S3.2	64	<u>0</u>	0	4,69	<u>0</u>	0,31	<u>0</u>	0,78	<u>0</u>
S3.3	66	<u>0</u>	0	0,8	<u>0</u>	0,15	<u>0</u>	<u>0</u>	<u>0</u>
S3.4	54	<u>0</u>	0	4,26	<u>0</u>	2,59	<u>0</u>	0,37	<u>0</u>
S3.5	66	<u>0</u>	0	3,59	<u>0</u>	2,58	<u>0</u>	0,15	<u>0</u>
S3.6	56	<u>0</u>	0	2,73	<u>0</u>	1,07	<u>0</u>	0,36	<u>0</u>
S3.7	57	<u>0</u>	0	1,93	1,75	1,93	1,75	1,75	1,75

S3'.1	<u>60</u>	<u>0</u>	0	4,55	1,67	2,33	<u>0</u>	2	<u>0</u>
S3'.2	<u>66</u>	<u>0</u>	0	6,36	1,52	2,42	1,52	0,76	<u>0</u>
S3'.3	<u>78</u>	<u>0</u>	0	5,47	2,56	2,95	2,56	1,54	<u>0</u>
S3'.4	<u>57</u>	<u>0</u>	0	4,16	<u>0</u>	0,7	<u>0</u>	0,7	<u>0</u>
S3'.5	<u>70</u>	<u>0</u>	0	2,39	<u>0</u>	1,29	<u>0</u>	1	<u>0</u>
S3'.6	<u>74</u>	<u>0</u>	0	10	5,41	7,16	5,41	3,65	1,35
S3'.7	<u>71</u>	<u>0</u>	0	6,62	4,23	4,93	4,23	4,37	2,82
Сред.	×		-2,39	4,11	1,67	2,17	1,14	1,27	0,57

Без передачи начального решения Gurobi способен показать лучшие результаты, чем алгоритм VNS, только на малых размерностях при достаточно долгом времени выполнения. Так, для большинства примеров размерности $|\mathcal{V}^{\text{CST}}| = 50$ при времени выполнения, равном 90 с, результаты Gurobi сильно хуже, чем у реализованного алгоритма. При времени 270 с результаты Gurobi в среднем лучше, чем у VNS, на 0,4%, а при времени 810 с Gurobi обходит VNS в среднем на 0,26%. Для большинства примеров размерности $|\mathcal{V}^{\text{CST}}| = 150$ Gurobi не способен найти сравнимые результаты в течение часа без передачи ему начального решения. При запусках Gurobi для примеров с 50 объектами использовались те же параметры, что определены выше в данном разделе, кроме параметра `ImproveStartTime`, который устанавливался равным 70% от общего времени выполнения, и параметра `MIPFocus = 1`. Для примеров со 150 объектами Gurobi запускался в режиме, использующем эвристический подход для поиска хорошего допустимого решения, посредством задания параметра `NoRelHeurTime = TimeLimit`.

Передача Gurobi хорошего допустимого решения задачи в качестве начального меняет ситуацию. Реализация схемы, где решение, найденное алгоритмом VNS, постоптимизируется при помощи запуска Gurobi с вышеописанными параметрами, позволяет достичь существенного улучшения результатов как на примерах размерности $|\mathcal{V}^{\text{CST}}| = 50$, так и на примерах размерности $|\mathcal{V}^{\text{CST}}| = 150$. Постоптимизация с помощью Gurobi для данных запусков начиналась с половины общего времени запуска или с момента, когда в алгоритме VNS в течение 20000 предыдущих итераций не произошло изменения лучшего найденного решения. На примерах с 50 объектами для общего времени одного запуска, равного 90, 270 и 810 с, такая схема приводит к улучшению результатов в сравнении со схемой, представленной в п. 2.2, в среднем на 1,95, 0,88 и 0,29% соответственно. Для примеров со 150 объектами указанная схема позволяет улучшить результаты в среднем на 1,82% при общем времени одного запуска, равном 900 с.

На примерах большой размерности с числом объектов $|\mathcal{V}^{\text{CST}}| = 500$ и числом ТС $|\mathcal{V}^{\text{DEP}}| = 30$, сгенерированных по тому же принципу, что

описан в п. 3.1, Gurobi с передачей ему начального решения не способен найти сопоставимые результаты ни для одного из примеров за 3 часа на запуск с параметрами $\text{NoRelHeurTime} = \text{TimeLimit}$, в сравнении с результатами запусков алгоритма VNS на 100 с.

3.4. Сравнение различных схем. Здесь приведём эксперименты по изучению влияния некоторых из компонент реализованного алгоритма на результаты его работы. В табл. 3 приведено сравнение трёх схем VNS: 1) метода, описанного в п. 2.2, со всеми окрестностями N_1-N_7 ; 2) схемы, в которой из алгоритма VND исключено решение распределительной задачи, а используются только локальные спуски; 3) алгоритма, отличающегося от основного метода тем, что в процедуре локального поиска вместо полного набора окрестностей N_1-N_7 используется лишь одна окрестность N_7 . Для второй и третьей схемы были отдельно настроены параметры KL_{cfg} , k_{max} и KL_{cfg} , l , $[\mu_1, \mu_2]$ соответственно. Сравнение производилось на примерах со 150 объектами, описанными ранее. Каждый пример запускался 10 раз на 900 с. В столбцах с 4 по 8 указано относительное отклонение (в %) от значения целевой функции для среднего результата схемы 1. Для каждого примера лучший средний результат выделен жирным шрифтом. Из табл. 3 видно превосходство схемы 1, используемой нами, над альтернативными вариантами.

Таблица 3

Результаты сравнения схем, $|\mathcal{V}^{\text{CST}}| = 150$

#	Схема 1			Схема 2		Схема 3	
	сред.		мин.	сред.	мин.	сред.	мин.
	t	τ					
S1.1	175,5	0	−1,42	0,97	−0,28	7,41	5,98
S1.2	193,6	3	−1,16	2,37	−2,06	6,84	1,52
S1.3	186,2	0	−1,18	0,7	−0,64	4,14	2,04
S1.4	198,9	0,5	−6,33	3,58	−3,87	6,87	1,52
S1.5	193,9	0	−2,01	3,51	0,05	5,67	3,15
S1.6	180,3	0	−1,83	2,38	−0,72	4,99	3,16
S1.7	170	0	−1,76	1,18	−0,59	7,88	4,71
S1'.1	367,9	1,6	−8,05	−1,07	−8,05	−3,23	−4,4
S1'.2	359,7	1,2	−5,03	−0,43	−4,76	−0,03	−1,53
S1'.3	351,9	0,1	−1,67	2,81	0,31	2,81	1,45
S1'.4	360,6	2,7	−5,83	0,57	−5,83	−3,87	−6,09
S1'.5	357,5	0,2	−2,64	3,89	0,14	2,53	0,7
S1'.6	369,6	5,6	−4,14	1,22	−1,79	−4,58	−7,66
S1'.7	357,5	0,7	−4,53	0,69	−3,16	1,23	−1,23
S2.1	193,9	0	−1,5	1,03	0,05	6,76	4,18
S2.2	186,8	0,8	−1,95	4,36	−1,44	4,36	2,67

S2.3	206,4	8	-3,28	-0,49	-1,54	2,62	0,21
S2.4	214,3	1,3	-8,49	-2,16	-8,05	2,07	-2,33
S2.5	180,6	0	-2,55	1,44	-0,33	6,7	4,65
S2.6	221,2	4,7	-3,43	3,24	-7,16	4,62	-1,19
S2.7	219,5	0	-2,51	3,51	-1,59	8,02	6,15
S2'.1	314,4	0,9	-5,69	3,12	-5,69	2,81	-1,67
S2'.2	311,3	5,9	-5,21	0	-5,48	-1,67	-7,1
S2'.3	361,8	2,1	-1,78	0,6	-1,78	3,5	-1,25
S2'.4	301,4	0,9	-5,61	2,03	-5,28	4,93	1,8
S2'.5	316,1	0	-2,25	-0,35	-2,88	8,07	5,66
S2'.6	319,4	4,4	-8,09	0,55	-3,96	-3,52	-7,54
S2'.7	317,8	0,9	-5,14	-1,77	-4,83	1,81	-0,55
S3.1	151,8	0	-2,5	1,19	-1,84	11,46	8,7
S3.2	151,3	0	-1,52	-0,93	-2,84	13,75	8,39
S3.3	150,5	0	-1,66	0,4	-0,33	10,96	6,98
S3.4	133,9	0	-3,66	0,82	-2,91	22,26	16,5
S3.5	146,6	0	-1,77	0,41	-3,14	12,69	8,46
S3.6	144,6	0	-1,8	-1,11	-5,26	15,91	12,03
S3.7	140,4	0	-2,42	1,07	-1,71	16,81	10,4
S3'.1	169,3	0	-3,72	2,42	-1,36	14,35	11,05
S3'.2	171,2	0	-6,54	1,58	-0,12	23,6	15,65
S3'.3	177,8	0	-3,26	1,46	-0,45	8,94	5,17
S3'.4	186,6	0,1	-8,85	8,58	-4,58	12,42	3,94
S3'.5	174,5	0	-3,15	1,49	-0,29	12,72	8,88
S3'.6	176,4	0	-4,76	2,15	-0,23	14	8,28
S3'.7	164,6	0	-2,19	1,64	-2,79	16,04	9,36
Сред.	×		-3,64	1,40	-2,60	7,04	3,35

Заключение

Изучена новая задача маршрутизации транспортных средств, возникающая в контексте передвижения буровых установок. Для этой задачи описана подзадача, которая сформулирована в виде задачи смешанного целочисленного линейного программирования и включена в алгоритм решения общей задачи маршрутизации. В результате проведённых экспериментов подтверждена эффективность использования решения подзадачи в процессе локального поиска.

Одним из направлений дальнейших исследований является улучшение устойчивости алгоритма к изменениям во входных данных. На практике в задачах маршрутизации транспортных средств значения для времени обслуживания и времени перемещения часто бывают заданы неточно, поэтому важно, чтобы решения, получаемые алгоритмом, были устойчивы к таким неопределённостям.

ЛИТЕРАТУРА

1. **Bräysy O., Gendreau M.** Vehicle routing problem with time windows, part I: Route construction and local search algorithms // *Transp. Sci.* 2005. Vol. 39, No. 1. P. 104–118.
2. **Gendreau M., Tarantilis C. D.** Solving large-scale vehicle routing problems with time windows: The state-of-the-art // *CIRRELT-2010-04*. Montreal: Interuniv. Res. Centre on Enterpr. Networks, Logist. and Transp., 2010.
3. **Ho S. C., Haugland D.** A tabu search heuristic for the vehicle routing problem with time windows and split deliveries // *Comput. Oper. Res.* 2004. Vol. 31, No. 12. P. 1947–1964.
4. **Vidal T., Crainic T. G., Gendreau M., Prins C.** A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows // *Comput. Oper. Res.* 2013. Vol. 40, No. 1. P. 475–489.
5. **Kulachenko I. N., Kononova P. A.** A matheuristic for the drilling rig routing problem // *Mathematical Optimization Theory and Operations Research. Proc. 19th Int. Conf. (Novosibirsk, Russia, July 6–10, 2020)*. Cham: Springer, 2020. P. 343–358. (Lect. Notes Comput. Sci.; Vol. 12095).
6. **Toth P., Vigo D.** Vehicle routing: Problems, methods, and applications. Philadelphia, PA: SIAM, 2014.
7. **Golden B. L., Raghavan S., Wasil E. A.** The vehicle routing problem: Latest advances and new challenges. New York: Springer, 2008.
8. **Archetti C., Speranza M. G.** Vehicle routing problems with split deliveries // *Int. Trans. Oper. Res.* 2012. Vol. 19, No. 1–2. P. 3–22.
9. **Braekers K., Ramaekers K., Van Nieuwenhuyse I.** The vehicle routing problem: State of the art classification and review // *Comput. Ind. Eng.* 2016. Vol. 99. P. 300–313.
10. **Lambert V., Laporte G., Louveaux F.** Designing collection routes through bank branches // *Comput. Oper. Res.* 1993. Vol. 20, No. 7. P. 783–791.
11. **Кулаченко И. Н., Кононова П. А.** Гибридный алгоритм локального поиска для задачи маршрутизации транспортных средств с многократным посещением клиентов // *Дискрет. анализ и исслед. операций*. 2020. Т. 27, № 2. С. 43–64.
12. **Salhi S., Imran A., Wassan N. A.** The multi-depot vehicle routing problem with heterogeneous vehicle fleet: Formulation and a variable neighborhood search implementation // *Comput. Oper. Res.* 2014. Vol. 52. P. 315–325.
13. **Li F., Golden B., Wasil E.** The open vehicle routing problem: Algorithms, large-scale test problems, and computational results // *Comput. Oper. Res.* 2007. Vol. 34, No. 10. P. 2918–2930.
14. **Yakici E., Karasakal O.** A min–max vehicle routing problem with split delivery and heterogeneous demand // *Optim. Lett.* 2013. Vol. 7. P. 1611–1625.
15. **Aloise D. J., Aloise D., Rocha C., Ribeiro C. C., Filho R., Moura L.** Scheduling workover rigs for onshore oil production // *Discrete Appl. Math.* 2006. Vol. 154, No. 5. P. 695–702.

16. **Ribeiro G., Desaulniers G., Desrosiers J., Vidal T., Vieira B.** Efficient heuristics for the workover rig routing problem with a heterogeneous fleet and a finite horizon // *J. Heur.* 2014. Vol. 20. P. 677–708.
17. **Pecin D., Contardo C., Desaulniers G., Uchoa E.** New enhancements for the exact solution of the vehicle routing problem with time windows // *INFORMS J. Comput.* 2017. Vol. 29, No. 3. P. 489–502.
18. **Archetti C., Speranza M. G.** A survey on matheuristics for routing problems // *EURO J. Comput. Optim.* 2014. Vol. 2. P. 223–246.
19. **Matheuristics: Hybridizing metaheuristics and mathematical programming.** New York: Springer, 2009. (Ann. Inf. Syst.; Vol. 10).
20. **Talbi El-G.** Hybrid metaheuristics. Berlin: Springer, 2013.
21. **Mladenovic N., Hansen P.** Variable neighborhood search // *Comput. Oper. Res.* 1997. Vol. 24. P. 1097–1100.
22. **Mladenovic N., Hansen P., Todosijević R., Hanafi S.** Variable neighborhood search: Basics and variants // *EURO J. Comput. Optim.* 2017. Vol. 5, No. 3. P. 423–454.
23. **Hemmelmayr V. C., Doerner K. F., Hartl R. F., Vigo D.** Models and algorithms for the integrated planning of bin allocation and vehicle routing in solid waste management // *Transp. Sci.* 2014. Vol. 48. P. 103–120.
24. **Talbi El-G.** Metaheuristics: From design to implementation. Hoboken, NJ: Wiley, 2009.
25. **Гэри М., Джонсон Д.** Вычислительные машины и труднорешаемые задачи. М.: Мир, 1982.
26. **Kernighan B. W., Lin S.** An efficient heuristic procedure for partitioning graphs // *Bell Labs Tech. J.* 1970. Vol. 49, No. 2. P. 291–307.
27. **Nagata Y., Bräysy O., Dullaert W.** A penalty-based edge assembly memetic algorithm for the vehicle routing problem with time windows // *Comput. Oper. Res.* 2010. Vol. 37, No. 4. P. 724–737.
28. **Solomon M. M.** Algorithms for the vehicle routing and scheduling problems with time window constraints // *Oper. Res.* 1985. Vol. 35. P. 254–265.
29. **Kirkpatrick S., Gelatt, Jr. C. D., Vecchi M. P.** Optimization by simulated annealing // *Science.* 1983. Vol. 220, No. 4598. P. 671–680.
30. **Hutter F., Hoos H. H., Leyton-Brown K.** Sequential model-based optimization for general algorithm configuration // *Learning and Intelligent Optimization. Sel. Pap. 5th Int. Conf. (Rome, Italy, Jan. 17–21, 2011).* Heidelberg: Springer, 2011. P. 507–523. (Lect. Notes Comput. Sci.; Vol. 6683).

Кулаченко Игорь Николаевич
Кононова Полина Александровна

Статья поступила
13 ноября 2020 г.
После доработки —
15 февраля 2021 г.
Принята к публикации
17 февраля 2021 г.

A HYBRID ALGORITHM FOR THE DRILLING RIG
ROUTING PROBLEMI. N. Kulachenko^{1, 2, a} and P. A. Kononova^{1, 2, b}¹ Novosibirsk State University,

2 Pirogov Street, 630090 Novosibirsk, Russia

² Sobolev Institute of Mathematics,

4 Acad. Koptuyug Avenue, 630090 Novosibirsk, Russia

E-mail: ^asoge.ink@gmail.com, ^bpkononova@math.nsc.ru

Abstract. In this research, the drilling rig routing problem is studied. There is a set of objects requiring well-drilling work. Each object has a time window that is the time interval during which all the work has to be started and completed. Several drilling rigs can operate at the same object simultaneously, which makes it possible to speed up the work. The objective is to determine a set of routes for a fleet of drilling rigs to perform all well-drilling requests on time and with minimal traveling costs.

The mixed-integer linear programming (MILP) model is presented for this problem. To find a feasible solution we use the Variable Neighborhood Search (VNS) metaheuristic. The algorithm also includes solving an MILP subproblem to redistribute the well-drilling work. This approach combines advantages of both the exact and heuristic methods. We present the results of comparison of the algorithm with Gurobi and alternative VNS schemes. Tab. 3, illustr. 1, bibliogr. 30.

Keywords: uncapacitated vehicles, matheuristic, split delivery service, time windows.

REFERENCES

1. O. Bräysy and M. Gendreau, Vehicle routing problem with time windows, Part I: Route construction and local search algorithms, *Transp. Sci.* **39** (1), 104–118 (2005).

This research is carried out within the framework of the state contract of the Sobolev Institute of Mathematics (Project 0314–2019–0014).

English version: Journal of Applied and Industrial Mathematics **15** (2), 261–276 (2021), DOI 10.1134/S1990478921020071.

2. **M. Gendreau** and **C. D. Tarantilis**, Solving large-scale vehicle routing problems with time windows: The state-of-the-art, *Technical Report 2010-04* (CIRRELT, Montreal, 2010).
3. **S. C. Ho** and **D. Haugland**, A tabu search heuristic for the vehicle routing problem with time windows and split deliveries, *Comput. Oper. Res.* **31** (12), 1947–1964 (2004).
4. **T. Vidal**, **T. G. Crainic**, **M. Gendreau**, and **C. Prins**, A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows, *Comput. Oper. Res.* **40** (1), 475–489 (2013).
5. **I. N. Kulachenko** and **P. A. Kononova**, A matheuristic for the drilling rig routing problem, in *Mathematical Optimization Theory and Operations Research* (Proc. 19th Int. Conf. MOTOR, Novosibirsk, Russia, July 6–10, 2020) (Springer, Cham, 2020), pp. 343–358 (Lect. Notes Comput. Sci., Vol. 12095).
6. **P. Toth** and **D. Vigo**, *Vehicle Routing: Problems, Methods, and Applications* (SIAM, Philadelphia, 2014).
7. **B. L. Golden**, **S. Raghavan**, and **E. A. Wasil**, *The Vehicle Routing Problem: Latest Advances and New Challenges* (Springer, New York, 2008).
8. **C. Archetti** and **M. G. Speranza**, Vehicle routing problems with split deliveries, *Int. Trans. Oper. Res.* **19** (1–2), 3–22 (2012).
9. **K. Braekers**, **K. Ramaekers**, and **I. Van Nieuwenhuyse**, The vehicle routing problem: State of the art classification and review, *Comput. Ind. Eng.* **99**, 300–313 (2016).
10. **V. Lambert**, **G. Laporte**, and **F. Louveaux**, Designing collection routes through bank branches, *Comput. Oper. Res.* **20** (7), 783–791 (1993).
11. **I. N. Kulachenko** and **P. A. Kononova**, A hybrid local search algorithm for consistent periodic vehicle routing problem, *Diskretn. Anal. Issled. Oper.* **27** (2), 43–64 (2020) [Russian] [*J. Ind. Appl. Math.* **14** (2), 339–351 (2020)].
12. **S. Salhi**, **A. Imran**, and **N. A. Wassan**, The multi-depot vehicle routing problem with heterogeneous vehicle fleet: Formulation and a variable neighborhood search implementation, *Comput. Oper. Res.* **52**, 315–325 (2014).
13. **F. Li**, **B. Golden**, and **E. Wasil**, The open vehicle routing problem: Algorithms, large-scale test problems, and computational results, *Comput. Oper. Res.* **34** (10), 2918–2930 (2007).
14. **E. Yakici** and **O. Karasakal**, A min-max vehicle routing problem with split delivery and heterogeneous demand, *Optim. Lett.* **7**, 1611–1625 (2013).
15. **D. J. Aloise**, **D. Aloise**, **C. Rocha**, **C. C. Ribeiro**, **R. Filho**, and **L. Moura**, Scheduling workover rigs for onshore oil production, *Discrete Appl. Math.* **154** (5), 695–702 (2006).
16. **G. Ribeiro**, **G. Desaulniers**, **J. Desrosiers**, **T. Vidal**, and **B. Vieira**, Efficient heuristics for the workover rig routing problem with a heterogeneous fleet and a finite horizon, *J. Heur.* **20**, 677–708 (2014).
17. **D. Pecin**, **C. Contardo**, **G. Desaulniers**, and **E. Uchoa**, New enhancements for the exact solution of the vehicle routing problem with time windows, *INFORMS J. Comput.* **29** (3), 489–502 (2017).

18. **C. Archetti** and **M. G. Speranza**, A survey on matheuristics for routing problems, *EURO J. Comput. Optim.* **2**, 223–246 (2014).
19. *Matheuristics: Hybridizing Metaheuristics and Mathematical Programming* (Springer, New York, 2009) (Ann. Inf. Syst., Vol. 10).
20. **El-G. Talbi**, *Hybrid Metaheuristics* (Springer, Berlin, 2013).
21. **N. Mladenovic** and **P. Hansen**, Variable neighborhood search, *Comput. Oper. Res.* **24**, 1097–1100 (1997).
22. **N. Mladenovic**, **P. Hansen**, **R. Todosijević**, and **S. Hanafi**, Variable neighborhood search: Basics and variants, *EURO J. Comput. Optim.* **5** (3), 423–454 (2017).
23. **V. C. Hemmelmayr**, **K. F. Doerner**, **R. F. Hartl**, and **D. Vigo**, Models and algorithms for the integrated planning of bin allocation and vehicle routing in solid waste management, *Transp. Sci.* **48**, 103–120 (2014).
24. **El-G. Talbi**, *Metaheuristics: From Design to Implementation* (Wiley, Hoboken, NJ, 2009).
25. **M. R. Garey** and **D. S. Johnson**, *Computers and Intractability: A Guide to the Theory of NP-Completeness* (Freeman, San Francisco, 1979; Mir, Moscow, 1982 [Russian]).
26. **B. W. Kernighan** and **S. Lin**, An efficient heuristic procedure for partitioning graphs, *Bell Labs Tech. J.* **49** (2), 291–307 (1970).
27. **Y. Nagata**, **O. Bräysy**, and **W. Dullaert**, A penalty-based edge assembly memetic algorithm for the vehicle routing problem with time windows, *Comput. Oper. Res.* **37** (4), 724–737 (2010).
28. **M. M. Solomon**, Algorithms for the vehicle routing and scheduling problems with time window constraints, *Oper. Res.* **35**, 254–265 (1985).
29. **S. Kirkpatrick**, **C. D. Gelatt**, and **M. P. Vecchi**, Optimization by simulated annealing, *Science* **220** (4598), 671–680 (1983).
30. **F. Hutter**, **H. H. Hoos**, and **K. Leyton-Brown**, Sequential model-based optimization for general algorithm configuration, in *Learning and Intelligent Optimization* (Sel. Pap. 5th Int. Conf., Rome, Italy, Jan. 17–21, 2011) (Springer, Heidelberg, 2011), pp. 507–523 (Lect. Notes Comput. Sci., Vol. 6683).

Igor N. Kulachenko
Polina A. Kononova

Received November 13, 2020
Revised February 15, 2021
Accepted February 17, 2021