

## МАТЭВРИСТИКА ДЛЯ МИНИМИЗАЦИИ ВРЕМЕНИ ОЖИДАНИЯ ТРЕЙЛЕРОВ ПРИ НЕТОЧНЫХ ВРЕМЕНАХ ПРИБЫТИЯ

А. В. Ратушный<sup>1, a</sup>, Ю. А. Кочетов<sup>2, b</sup>

<sup>1</sup> Новосибирский гос. университет,  
ул. Пирогова, 2, 630090 Новосибирск, Россия

<sup>2</sup> Институт математики им. С. Л. Соболева,  
пр. Акад. Коптюга, 4, 630090 Новосибирск, Россия

E-mail: <sup>a</sup>alexeyratushny@gmail.com, <sup>b</sup>jkochet@math.nsc.ru

**Аннотация.** Рассматривается новая задача планирования погрузки/разгрузки трейлеров на складах логистической компании. Имеется здание с несколькими складами. На каждом складе хранятся поддоны с различными видами продукции для погрузки в трейлеры. Каждый склад имеет ворота с двух противоположных сторон здания. Ворота на одной стороне предназначены для обслуживания трейлеров, ворота на другой стороне — для двух погрузчиков из центральной зоны, которая является производственной линией. Центральная зона производит продукты, которые должны быть размещены на складах сразу же после готовности. Время прибытия каждого трейлера является неопределённым. Требуется распределить все трейлеры по складам и составить расписание их обслуживания с максимальным радиусом устойчивости при ограничении на суммарное время ожидания. Для этой NP-трудной задачи разработана двухэтапная эвристика. На первом этапе решается упрощённая модель с помощью коммерческого решателя Gurobi. Затем используется алгоритм локального поиска, чтобы вернуть решение в допустимую область с учётом информации о наличии поддонов на каждом складе. Для вычислительных экспериментов рассматривается несколько наборов примеров, созданных на основе реальных данных одной голландской компании. Обсуждаются результаты вычислительных экспериментов для 6 складов, 18 видов продукции и 90 трейлеров. Табл. 4, ил. 4, библиогр. 15.

**Ключевые слова:** радиус устойчивости, матэвристика, VNS, неопределённость.

---

Исследование выполнено при поддержке Российского научного фонда (проект № 21–41–09017).

© А. В. Ратушный, Ю. А. Кочетов, 2022

### Введение

Рассматривается новая задача поиска расписания обслуживания трейлеров для логистической компании с производственной линией и несколькими складами (рис. 1). Каждый склад имеет несколько комнат хранения с известной вместимостью. Каждая комната либо пуста, либо содержит продукты одного и того же типа. Известно содержимое каждой комнаты. Каждый склад имеет двое ворот на противоположных сторонах. Одни ворота используются для обслуживания трейлеров от клиентов. Другие ворота предназначены для загрузки продуктов из центральной зоны. Внешняя зона хранения используется для вечернего перераспределения поддонов и может быть опущена в задаче оптимизации. Центральная зона представляет собой производственную линию, на которую поступают поддоны с продуктами из силосных башен. Все продукты на каждом поддоне имеют один и тот же тип. График работы производственной линии известен. Она работает без простоев, т. е. каждый поддон, как только он готов, с производственной линии перемещается на один из складов для пополнения запасов в его комнатах. Центральная зона может производить не более двух поддонов в любой момент времени. Её обслуживают два погрузчика. В соответствии с правилами техники безопасности на каждом складе может работать только один погрузчик. Другими словами, невозможно обслуживать какой-либо трейлер, когда на том же складе работает погрузчик из центральной зоны. Только один погрузчик может обслуживать трейлер клиента. Таким образом, на каждом складе в любое время может обслуживаться не более одного трейлера. В случае очереди трейлеры должны ждать в зоне ожидания. Общее время ожидания трейлеров имеет важное значение для компании. Оно не может превышать заданный порог.

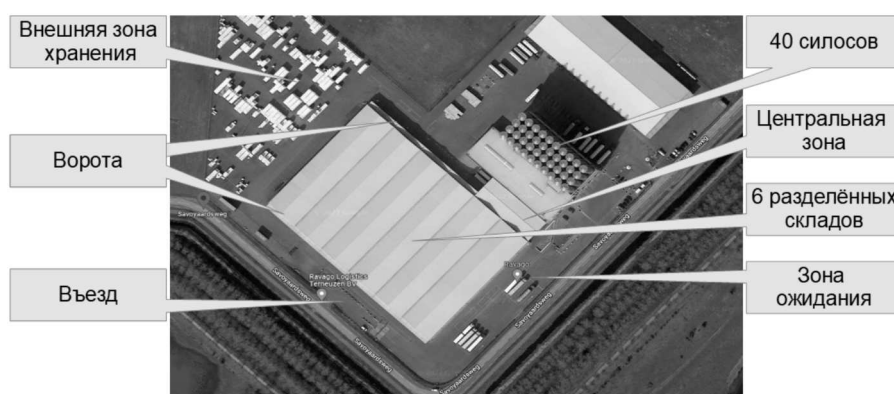


Рис. 1. Пример склада

Трейлеры клиента могут загружать или выгружать поддоны. Мы предполагаем, что каждый трейлер может загружать или выгружать поддоны только с одним типом продукта. Известны количество поддонов для каждого трейлера и время его погрузки/разгрузки. Время прибытия также известно, но это неопределённые данные. На эти значения влияет множество непредсказуемых факторов. Известно только самое раннее возможное время прибытия каждого трейлера. Мы не знаем исторических данных, распределения вероятностей и другой информации для этих входных параметров. Требуется найти распределение трейлеров по складам и определить их перестановки (очереди), чтобы максимизировать радиус устойчивости при ограничении общего времени ожидания. Другими словами, мы применяем идею пороговой робастности [1] и стараемся максимально повысить надёжность нашего расписания.

Робастная оптимизация является популярной темой в области исследования операций. Пороговая робастность — достаточно хорошо известная идея. Например, в [2] она применяется для классических задач размещения: задачи о  $p$ -медиане и задачи размещения предприятий. В [3] разработана интересная эвристика на основе частично целочисленного программирования (MIP) для поиска решения задачи балансировки автоматической линии. В [4] авторы ищут расписание для судов с неопределённым временем прибытия, пытаясь обеспечить временные коридоры. Различные подходы к робастной оптимизации можно найти в [5–9].

Для решения рассматриваемой задачи разработана двухэтапная метавэристика. Сначала решается упрощённая математическая модель с помощью решателя Gurobi [10]. Таким образом, быстро получается хорошее, но недопустимое решение. Наша упрощённая формулировка игнорирует нелинейные ограничения, связанные с описанием содержимого комнат для каждого склада. На втором этапе мы пытаемся устранить нарушения ограничений. Одна из идей состоит в том, чтобы добавить относительно небольшое количество новых неравенств с использованием существующих переменных. Это помогает значительно сократить количество нарушений, что демонстрируется в вычислительных экспериментах. Другой способ — применить эвристику локального поиска с методом штрафных функций. С этой целью мы разработали алгоритм локального поиска (BVNS) с четырьмя окрестностями. Предлагаемый подход помогает вернуться в допустимую область и не оказывает критического влияния на значение целевой функции.

Статья организована следующим образом. В разд. 1 вводятся обозначения и формулируется математическая модель. В этом же разделе приводится доказательство NP-трудности задачи. Разд. 2 содержит описание метавэристики и полиномиального алгоритма вычисления целевой функции. Результаты вычислительных экспериментов обсуждаются

в разд. 3. В заключении подводятся итоги и обсуждаются возможные направления дальнейших исследований.

### 1. Математическая модель

Рассмотрим конечное множество складов  $I$  и обозначим через  $V_i$  количество комнат на складе  $i \in I$ . Предполагаем, что все комнаты идентичны, а все поддоны в каждой комнате с продуктом одного и того же типа. Через  $K$  обозначим конечный набор типов продукта, а через  $\varkappa_k$  ( $k \in K$ ) — вместимость каждой комнаты для поддона с данным типом продукта. Известно начальное содержимое  $a_{ik}$ ,  $k \in K$ , каждого склада  $i \in I$  и предполагается, что  $\lceil a_{ik}/\varkappa_k \rceil$  комнат заняты продуктом типа  $k$ .

Мы знаем расписание центральной зоны и моделируем её с помощью набора  $J^\nu$  виртуальных трейлеров. Известно время прибытия  $r_j$  каждого трейлера  $j$ , время его обслуживания  $p_j$  и количество поддонов  $\hat{p}_j$ . Напомним, что виртуальные трейлеры обслуживаются без задержек. Набор клиентских трейлеров обозначим через  $J^\mu$ . В случае очереди они могут дожидаться своего обслуживания в зоне ожидания. Предполагаем, что общее время ожидания не может превышать порогового значения  $F^*$ . Известны время прибытия  $r_j$ , время обслуживания  $p_j$  и количество поддонов  $\hat{p}_j$  для каждого клиентского трейлера, при этом параметр  $\hat{p}_j$  отрицателен для выгружающихся трейлеров. Обозначим через  $J = J^\mu \cup J^\nu$  общий набор трейлеров и введём набор  $J(k)$  клиентских трейлеров с типом продукта  $k \in K$ .

Введём следующие переменные:

- $x_{ij} = 1$ , если трейлер  $j \in J$  назначен на склад  $i \in I$ , и  $x_{ij} = 0$  в противном случае,
- $z_{jl} = 1$ , если трейлер  $j \in J$  обслуживается после трейлера  $l \in J$  на том же складе, и  $z_{jl} = 0$  в противном случае,
- $s_j \geq 0$  — время начала обслуживания трейлера  $j$ ,
- $w_j \geq 0$  — время ожидания трейлера  $j \in J^\mu$ ,
- $\theta_{ikj} \geq 0$  — целое число комнат с типом продукта  $k$  на складе  $i$  после обслуживания трейлера  $j$ .

Представим задачу в виде следующей частично целочисленной модели с двумя нелинейными ограничениями:

$$\min_{j \in J^\mu} w_j \rightarrow \max, \quad (1)$$

$$s_j \geq r_j + w_j, \quad j \in J^\mu, \quad (2)$$

$$s_j = r_j, \quad j \in J^\nu, \quad (3)$$

$$\sum_i x_{ij} = 1, \quad j \in J, \quad (4)$$

$$s_j \geq s_l + p_l + M(x_{il} + x_{ij} - 2) + M(z_{jl} - 1), \quad j, l \in J, \quad j \neq l, \quad i \in I, \quad (5)$$

$$z_{jl} + z_{lj} = 1, \quad j, l \in J, \quad j \neq l, \quad (6)$$

$$\sum_{j \in J^\mu} (s_j - r_j) \leq F^*, \quad (7)$$

$$\sum_{j \in J(k)} \hat{p}_j x_{ij} z_{lj} \leq a_{ik}, \quad i \in I, \quad k \in K, \quad l \in J(k), \quad (8)$$

$$\sum_{k \in K} \theta_{ikl} \leq V_i, \quad i \in I, \quad l \in J, \quad (9)$$

$$\theta_{ikl} \kappa_k \geq a_{ik} + \sum_{j \in J(k)} \hat{p}_j x_{ij} z_{jl}, \quad i \in I, \quad k \in K, \quad l \in J(k), \quad (10)$$

$$x_{ij}, z_{jl} \in \{0, 1\}, \quad i \in I, \quad j, l \in J, \quad (11)$$

$$\theta_{ikj} \geq 0, \quad \theta_{ikj} \in \mathbb{Z}, \quad s_j, w_j \geq 0, \quad i \in I, \quad j \in J, \quad k \in K. \quad (12)$$

Целевая функция (1) отвечает за радиус устойчивости решения. Хотим получить максимальную свободу для возможной задержки каждого трейлера. Заметим, что целевая функция (1) не совпадает с нормой вектора  $w = (w_j)$ , как в [2]. Неравенства (2) и (3) определяют время начала обслуживания клиентских и виртуальных трейлеров. Равенства (4) гарантируют, что каждый трейлер будет отнесён точно к одному складу. Ограничения (5) определяют перестановку трейлеров для каждого склада в соответствии с булевыми переменными  $z_{jl}$  и большой константой  $M$ . Равенства (6) определяют порядок для каждой пары трейлеров. Неравенство (7) — общее ограничение времени ожидания для всех трейлеров. Ограничения (8) гарантируют достаточное количество поддонов на складе во время загрузки. С помощью (9) ограничиваем общее число комнат на каждом складе. Наконец, неравенства (10) позволяют рассчитать число комнат с каждым типом продукции на каждом складе после каждого трейлера.

Представленная задача NP-трудна в сильном смысле даже для одного склада с неограниченной вместимостью. Чтобы показать это, используем задачу о 3-разбиении [11]. Входными данными задачи является множество  $S$  из  $n = 3m$  положительных целых чисел  $t_j$  таких, что  $\sum_{j=1}^{3m} t_j = mT$  и  $T/4 < t_j < T/2$ . Результатом является ответ на вопрос, существует ли такое разбиение множества  $S$  на  $m$  триплетов  $S_1, S_2, \dots, S_m$ , чтобы сумма чисел в каждом из них была равна  $T$ .

**Теорема 1.** *Задача о 3-разбиении может быть сведена к задаче (1)–(7), (11)–(12) с одним складом.*

**ДОКАЗАТЕЛЬСТВО.** По исходным данным задачи о 3-разбиении построим пример задачи (1)–(7), (11)–(12) следующим образом. Определим

множество  $J^\mu$  как  $3m$  трейлеров с  $r_j = 0$ ,  $p_j = t_j$ ,  $j \in J^\mu$ . Пусть множество  $J^\nu$  содержит  $m-1$  трейлеров с  $r_j = (2j-1)T$ ,  $p_j = T$  и дополнительный трейлер  $j^*$  с  $r_{j^*} = 2mT$ ,  $p_{j^*} = F^*$ . Значение  $F^*$  определим позже. Напоминаем, что все виртуальные трейлеры должны обслуживаться без ожидания (тёмные блоки на рис. 2).

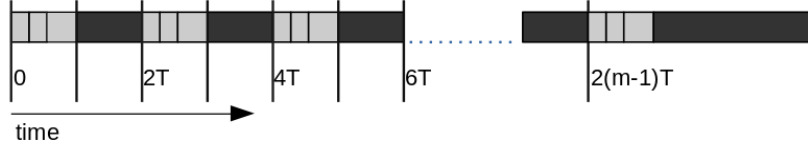


Рис. 2. Структура решения

Пусть трейлеры клиентов  $j_1, j_2, j_3$  обслуживаются в первый интервал времени  $[0, T]$ . Тогда  $w_{j_1} = 0$ ,  $w_{j_2} = p_{j_1}$ ,  $w_{j_3} = p_{j_1} + p_{j_2}$ . Следовательно, можем считать, что  $p_{j_1} \leq p_{j_2} \leq p_{j_3}$ . Таким образом,  $p_{j_1} \leq T/3$  и  $p_{j_2} < T/2$  и, как результат,  $w_{j_1} + w_{j_2} + w_{j_3} < 7T/6$ . Теперь можно оценить общее время ожидания следующим образом:

$$\begin{aligned} \sum_{j \in J^\mu} w_j &< \frac{7T}{6} + \left(6T + \frac{7T}{6}\right) + \left(12T + \frac{7T}{6}\right) \\ &+ \dots + \left(6(m-1)T + \frac{7T}{6}\right) = \frac{7T}{6}m + 3(m-1)mT. \end{aligned}$$

Положим  $F^* = 7mT/6 + 3(m-1)mT$ . Если нужного разбиения в задаче о 3-разбиении не существует, то по крайней мере один из клиентских трейлеров должен быть обслужен после  $j^*$  и задача (1)–(7), (11)–(12) не имеет допустимых решений. В противном случае получаем желаемое 3-разбиение. Теорема 1 доказана.

Задача (1)–(12) довольно сложна для решения, в частности, из-за нелинейных ограничений (8), (10) и большого числа переменных  $\theta_{ikj}$ . Мы рассматриваем упрощённую задачу (1)–(7), (11)–(12) и решаем её с помощью коммерческого решателя Gurobi. Как правило, получаемые решения нарушают отсутствующие ограничения (8)–(10). Некоторые трейлеры получают назначение на склады без необходимого продукта. Чтобы избежать этого, можно добавить следующие неравенства:

$$x_{ij} \leq 1 + \frac{a_{ik} - \hat{p}_j}{\varkappa_k}, \quad i \in I, \quad j \in J(k), \quad k \in K. \quad (13)$$

Однако этот подход не всегда возвращает решение в допустимую область, поэтому используем метаэвристику локального поиска.

## 2. Матэвристический подход

В этом разделе вводятся кодировка решений и процедура вычисления целевой функции за полиномиальное время и обсуждаются детали VNS алгоритма [12–14] для поиска решений, близких к оптимальным. Для кодировки решения будем использовать переменные  $(x_{ij})$  назначения трейлеров на склады и переменные, задающие очереди  $(z_{jl})$ . Покажем, как рассчитать время начала обслуживания для каждого трейлера с максимально возможным радиусом стабильности (1). Во время вычисления, если у клиентского трейлера недостаточно времени для обслуживания перед виртуальным, то он обслуживается после него и соответствующая очередь перестраивается.

Введём следующие обозначения:

$$w_{(1)} = \min_{j \in J^\mu} w_j, \quad w_{(2)} = \min_{j \in J^\mu} \{w_j \mid w_j > w_{(1)}\}$$

для минимального времени ожидания среди клиентских трейлеров и второго минимального времени ожидания. Обозначим через  $Q_j$  блок клиентских трейлеров, обслуживаемых сразу друг за другом (без простоя склада) на одном и том же складе, начиная с трейлера  $j \in J^\mu$ . После каждого блока идёт либо временное окно, либо виртуальный трейлер. Длину временного окна обозначим через  $\Delta w(Q_j)$ . Псевдокод процедуры вычисления целевой функции представлен в виде Алгоритма 1.

Идея подхода состоит в том, чтобы начать с раннего расписания и постепенно сдвигать блоки клиентских трейлеров, задержка которых минимальна. После каждой итерации возможны следующие ситуации:

- (i) достигнуто пороговое значение  $F^*$  для общего времени ожидания;
- (ii) целевая функция увеличилась достаточно, чтобы добавить новый блок или расширить существующий;
- (iii) один из блоков упирается в виртуальный трейлер.

В первом случае мы должны остановить вычисления и предъявить полученный результат. В третьем случае нужно перестроить расписание так, чтобы последний трейлер в текущем блоке обслуживался после виртуального. Такая перестановка расписания может привести к нарушению неравенства (7). В таком случае мы должны остановить вычисления. Через  $\alpha$  обозначено значение минимально возможного сдвига для достижения одного из случаев (i), (ii) или (iii).

На каждом шаге Алгоритма 1 увеличиваем по крайней мере один блок или перемещаем клиентский трейлер за виртуальный. Таким образом, проводится  $O(|J^\nu||J^\mu| + |J^\mu|)$  итераций. Если  $J^\mu = \bigcup_{j \in M} Q_j$  и все клиентские трейлеры обслуживаются после всех виртуальных трейлеров, то требуется только одна итерация. Следовательно, общая временная сложность алгоритма не превышает  $O(|J^\mu|^2(|J^\nu| + 1))$ .

**Алгоритм 1.** Алгоритм вычисления целевой функции**Вход:** Закодированное решение  $e = (x_{ij}, z_{jl})$ .**Выход:**  $w_{(1)}$ .

- 1: Вычислить значения  $s_j$  для активного расписания
- 2: Положить  $w_j \leftarrow s_j - r_j, j \in J^\mu$
- 3: **while** true **do**
- 4:   Найти набор клиентских трейлеров с минимальным временем ожидания  $M \leftarrow \{j \in J^\mu \mid w_j = w_{(1)}\}$
- 5:   Сформировать блок  $Q_j$  и вычислить  $\Delta w(Q_j)$  для всех  $j \in M$
- 6:   **if**  $\exists j: \Delta w(Q_j) = 0$  **then**
- 7:     Установить время начала последнего трейлера в  $Q_j$  таким образом, чтобы он следовал за соответствующим виртуальным трейлером
- 8:     **if** неравенство (7) нарушается **then break**
- 9:      $\alpha \leftarrow \min \left\{ \min_{j \in M} \Delta w(Q_j), w_{(2)} - w_{(1)}, \left\lfloor \frac{F^* - \sum_{j \in J} (s_j - r_j)}{\sum_{j \in M} |Q_j|} \right\rfloor \right\}$
- 10:   Сдвинуть все блоки  $Q_j$  на  $\alpha$
- 11:   **if**  $\alpha = \left\lfloor \frac{F^* - \sum_{j \in J} (s_j - r_j)}{\sum_{j \in M} |Q_j|} \right\rfloor$  **then break**
- 12: **return**  $w_{(1)}$

В схеме кодирования  $e = (x_{ij}, z_{jl})$  игнорируются ограничения (8)–(10) и может быть получено недопустимое решение. Чтобы вернуться в допустимую область, применяется метод штрафных функций [15], а целевая функция модифицируется следующим образом:

$$\min_{j \in J^\mu} w_j - \lambda(P_1 + P_2) \rightarrow \max, \quad (14)$$

$$P_1 = \sum_{i,j} \sum_k \max(\theta_{ikj} - V_i, 0), \quad (15)$$

$$P_2 = \sum_{i,k} \sum_{j \in J(k)} \max(\hat{p}_j x_{ij} z_{jl} - a_{ik}, 0), \quad (16)$$

где новые параметры  $P_1$  и  $P_2$  являются штрафами за нарушения ограничений (8)–(9). Значение  $\lambda$  может равняться верхней границе для (1), например  $\lambda = F^*/|J^\mu|$ .

В дополнение к максимизации радиуса устойчивости также важно общее время ожидания в раннем расписании. Поэтому для алгоритма VNS предлагается рассмотреть две целевые функции и оптимизировать их в лексикографическом порядке



$$\text{Lexmax} \left\{ \min_{j \in J^\mu} w_j - \lambda(P_1 + P_2), - \sum_{j \in J^\mu} (s_j - r_j) \right\}. \quad (17)$$

Однако использование этой целевой функции, тем не менее, невозможно в модели (1)–(12). Это связано с тем, что второе выражение вычисляется для раннего расписания, в то время как первое вычисляется для позднего расписания.

Определим процедуры `shake` и `local_search` для описания алгоритма BVNS. Набор окрестностей  $N_{sh}$  для процедуры `shake` включает в себя две окрестности: `cross` и `shuffle`. В окрестности `cross` меняем хвосты очередей для двух случайных складов, начиная со случайных элементов. Очереди формируются в порядке возрастания времени начала обслуживания и включают в себя также и виртуальные трейлеры. Для `shuffle` полностью перетасовываем очередь клиентских трейлеров для одного из складов. После одной из таких `shake`-процедур решение сильно меняется, что позволяет нам отойти от текущего локального оптимума. Для `local_search` ищем наилучшее решение в одной из окрестностей набора  $N_{ls}$ , который состоит из `swap` и `move`. Первая из них меняет местами два трейлера  $j_1, j_2 \in J$  независимо от их складов и требуемых типов продукта. В то же время она сохраняет позицию в очереди. Например, если трейлер  $j_1$  находился на 5-й позиции, то  $j_2$  будет на 5-й позиции после `swap`. Для `move` рассматривается перенос трейлера с одного склада на другой, при этом учитываются все возможные позиции в новой очереди, которые не нарушают ограничений. Псевдокод предлагаемого алгоритма BVNS представлен ниже в виде Алгоритма 2.

---

#### Алгоритм 2. Алгоритм BVNS

---

**Вход:** Закодированное решение  $e = (x_{ij}, z_{jl})$  задачи (1)–(7), (11)–(13).

- 1: **while** не выполнен критерий остановки **do**
  - 2:    $k \leftarrow 1$ ;
  - 3:   **while**  $k \leq k_{\max}$  **do**
  - 4:      $e' \leftarrow \text{shake}(e, k, N_{sh})$
  - 5:      $e'' \leftarrow \text{local\_search}(e', k, N_{ls})$
  - 6:     `pipe_neighborhood_change`( $e, e'', k$ )
  - 7: **return** лучшее найденное решение
- 

Алгоритм 2 требует достаточно много времени. Чтобы его ускорить, используем только часть каждой окрестности для локального поиска на шаге 5. Для этого вводится два дополнительных параметра. Значение  $p_s$  отвечает за размер случайной части окрестности `swap`, а значение  $p_m$  делает то же самое для окрестности `move`. Кроме того, значения  $k_s$  и  $k_m$  отвечают за максимальное количество итераций локального

поиска без улучшений с момента последнего улучшения. Таким образом, локальный поиск всегда выполняет не менее  $k_s$  (или  $k_m$ ) итераций и возвращает наилучшее найденное решение. Значение  $k_{\max}$  равняется 2 и отвечает за количество окрестностей. Процедура `pipe_neighborhood_change` заменяет существующее решение  $e$  решением  $e''$ , если новое решение  $e''$  лучше чем  $e$  в соответствии с целевой функцией (17). Она также заменяет окрестности процедур `shake` и `local_search` другими, независимо от того, является ли решение  $e''$  лучше чем  $e$  или нет.

### 3. Вычислительные эксперименты

Для проверки эффективности предложенного подхода проведены численные эксперименты на полусинтетических тестовых примерах, основанных на реальных данных одной голландской логистической компании. В каждом примере рассматривается 6 складов, 18 видов продукции и 90 трейлеров. Каждый склад состоит из 64 комнат. Вместимость каждой комнаты равна 30 поддонам, независимо от типа продукта. Соотношение трейлеров для всех примеров  $|J^\nu|/|J^\mu| \approx 35/55$ .

Алгоритм BVNS реализован на C++. Все эксперименты проводились на персональном компьютере с Windows 10, процессором Intel Core i7-9700 с частотой 3,00 ГГц и 8 ГБ оперативной памяти. Ограничения по времени составляют две минуты для каждого этапа: для решателя Gurobi и BVNS. Для BVNS использовались параметры  $p_s = p_m = 0,2$  и  $k_s = k_m = 5$ . Каждый тестовый пример решался 30 раз с помощью рандомизированного алгоритма BVNS, после чего сохранялись среднее значение и стандартное отклонение для целевой функции (1).

Результаты вычислений представлены в табл. 1–4. Каждая таблица состоит из следующих столбцов:

- 1) № — номер примера;
- 2)  $F^*$  — пороговое значение для суммарного времени ожидания;
- 3) Gurobi gap — отклонение верхней границы от значения целевой функции в решении, полученном Gurobi;
- 4) BVNS gap avg. — среднее отклонение целевой функции (1) от верхней границы в решениях BVNS;
- 5) BVNS gap std. — стандартное отклонение для целевой функции (1) в решениях BVNS;
- 6) число штрафов — количество неравенств (8)–(9), которые не выполнены для решения Gurobi задачи (1)–(7), (11)–(13);
- 7) время Gurobi — суммарное время ожидания клиентских трейлеров в решении Gurobi;
- 8) время BVNS avg. — среднее время ожидания клиентских трейлеров в решениях BVNS.

Таблица 1

**Результаты вычислительных экспериментов для примеров  
с достаточным количеством запасов и пустых комнат**

№	$F^*$	Gurobi gap	BVNS gap avg.	BVNS gap std.	Число штрафов	Время Gurobi	Время BVNS avg.
1	18008	0,87%	0,87%	0,00%	0	7002	5784
2	25484	5,95%	5,93%	0,06%	0	13443	10646
3	23789	10,32%	9,98%	0,90%	0	14254	9627
4	21536	4,46%	4,46%	0,00%	0	10206	8180
5	17083	7,17%	7,17%	0,00%	0	6799	3680

В табл. 1 представлены результаты вычислений для тестовых примеров с большим количеством запасов на складах и достаточным количеством пустых комнат для виртуальных трейлеров из центральной зоны. Неравенства (8)–(10) в этом случае не влияют на решение. Gurobi способен находить почти оптимальные решения с относительной погрешностью не более 10%. Алгоритмом BVNS удалось улучшить эти решения для двух примеров и значительно сократить общее время ожидания.

В табл. 2 и 3 показаны результаты вычислений для сложных примеров. Половина видов продукции находится только на двух складах в начале дня. Во всех случаях BVNS не смог достичь результатов Gurobi. Тем не менее он работает стабильно и всегда способен вернуть решение в допустимую область. Те же примеры были использованы для табл. 3, но они решались без неравенств (13). После первого этапа наблюдаем значительно большее число нарушений в неравенствах (8), (10), но даже в этом случае BVNS способен их исправить. Однако результирующие решения намного хуже. Другими словами, дополнительные неравенства (13) на первом этапе могут быть очень полезны.

Таблица 2

**Результаты вычислительных экспериментов  
для сложных примеров**

№	$F^*$	Gurobi gap	BVNS gap avg.	BVNS gap std.	Число штрафов	Время Gurobi	Время BVNS avg.
1	11423	2,56%	3,26%	0,37%	3	4107	2767
2	15316	1,43%	2,25%	0,28%	2	6531	3723
3	19776	4,19%	5,18%	0,60%	2	9844	6689
4	16062	1,02%	2,39%	0,45%	2	6756	3915
5	11680	1,49%	3,11%	1,09%	1	5041	3629

Таблица 3

**Результаты вычислительных экспериментов для сложных примеров без использования ограничений (13)**

№	$F^*$	Gurobi gap	BVNS gap avg.	BVNS gap std.	Число штрафов	Время Gurobi	Время BVNS avg.
1	11423	1,52%	7,81%	1,41%	17	4596	2870
2	15316	1,43%	2,95%	0,53%	9	5757	3461
3	19776	3,32%	6,56%	0,69%	16	9760	6683
4	16062	1,02%	4,52%	1,45%	14	8059	2943
5	11680	0,99%	8,63%	1,22%	23	5589	2476

Примеры для табл. 4 не имеют в наличии на складе в начале дня продуктов нескольких типов. Во время работы складов недостающие товары либо производятся центральной зоной, либо привозятся клиентскими трейлерами. Набор неравенств (13) может сделать модель несовместной с такими данными, поэтому игнорируем их. Однако алгоритм VNS позволяет получать допустимые решения. Также он способен улучшить качество решений, показывая стабильные результаты.

Таблица 4

**Результаты вычислительных экспериментов для примеров с отсутствующими продуктами**

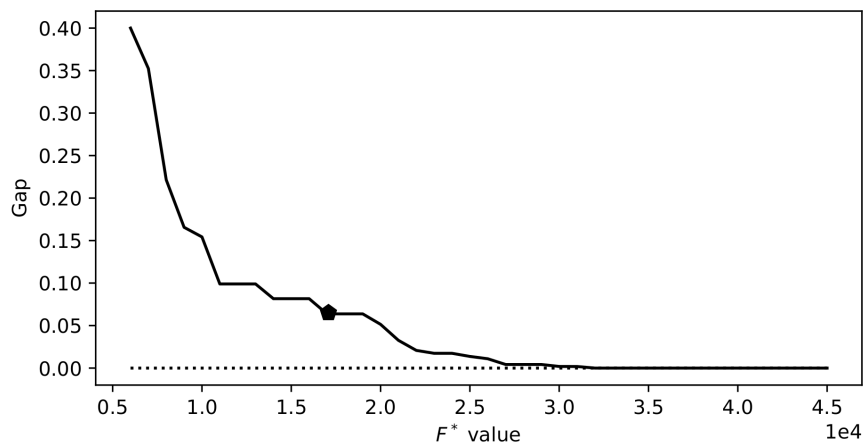
№	$F^*$	Gurobi gap	BVNS gap avg.	BVNS gap std.	Число штрафов	Время Gurobi	Время BVNS avg.
1	15408	4,87%	4,60%	0,21%	1	8030	5981
2	22072	5,07%	5,14%	0,93%	3	9560	7229
3	20197	8,76%	8,58%	0,32%	4	11711	6883
4	22033	7,94%	7,32%	0,74%	2	15232	8771
5	22672	13,55%	10,09%	1,78%	3	12219	9461

Значение параметра  $F^*$  оказывает сильное влияние на качество решений. Чтобы выбрать это значение для любого примера, генерируем множество жадных решений, случайным образом распределяя трейлеры по складам. Затем все клиентские трейлеры обслуживаются после виртуальных (рис. 3). После этого вычисляем значение  $F^* = \sum_{j \in J} (s_j - r_j)$  для каждого решения и используем среднее для вычислительных экспериментов (рис. 4).

Чтобы изучить влияние порогового значения  $F^*$  на результирующие решения, используем Gurobi для примера 5 из табл. 1 с различными значениями  $F^*$  (см. рис. 4). Чёрным пятиугольником отмечено значение,



Рис. 3. Сдвиг трейлеров

Рис. 4. Поведение решения при различных значениях  $F^*$ 

используемое в таблице. При малых значениях  $F^*$  задача имеет слишком мало свободы для распределения трейлеров, поскольку существует высокая вероятность нарушения неравенства (7). Для некоторых значений  $F^*$  даже найти допустимое решение может быть очень сложно. При больших значениях  $F^*$  задача распадается на две части. Первая заключается в распределении виртуальных трейлеров, вторая — в распределении клиентских трейлеров без перекрытия расписаний. Таким образом, надо ожидать различного поведения решения этой задачи для разных пороговых значений. Структура решения для большого значения  $F^*$  очень проста. Все клиентские трейлеры обслуживаются после виртуальных. Малые значения  $F^*$  наиболее сложны и могут приводить к несовместным системам ограничений.

### Заключение

В данной работе рассмотрена новая задача поиска расписания обслуживания трейлеров для логистической компании с производственной линией и несколькими складами. Целью являлся поиск расписания с максимальным радиусом устойчивости при ограничении на суммарное время ожидания. Показана NP-трудность задачи, а также введена

кодировка, основанная на назначении виртуальных трейлеров по складам, чтобы смоделировать работу производственной линии и отобразить очереди клиентских трейлеров. Разработан полиномиальный алгоритм вычисления целевой функции для заданной кодировки. Решение задачи ищется с помощью двухэтапной эвристики с использованием решателя Gurobi для упрощённой модели и алгоритма BVNS с использованием штрафных функций. В вычислительных экспериментах использовалось несколько наборов примеров, основанных на реальных данных некоторой голландской логистической компании. Результаты вычислений демонстрируют эффективность и стабильность предложенного подхода.

Для дальнейших исследований интересно расширить задачу, добавив новые условия. В дополнение к типам продукции стоит различать группы поддонов, произведённых в разное время и называемых партиями. Из-за особенностей производства разные партии с одним и тем же типом продукта могут незначительно отличаться. Однако для клиентов может быть важно, чтобы все поддоны были из одной партии. Одинаковая вместимость комнат является упрощающим предположением. Размеры поддонов разных типов необязательно должны совпадать. Это может существенно повлиять на вместимость комнат и переполненность складов.

#### ЛИТЕРАТУРА

1. Carrizosa E., Nickel S. Robust facility location // *Math. Methods Oper. Res.* 2003. V. 58. P. 331–349.
2. Carrizosa E., Ushakov A., Vasilyev I. Threshold robustness in discrete facility location problems: A bi-objective approach // *Optim. Lett.* 2015. V. 9. P. 1297–1314.
3. Borisovsky P., Battaïa O. MIP-based heuristics for a robust transfer lines balancing problem // *Optimization and Applications. Proc. Int. Conf. OPTIMA 2021 (Petrovac, Montenegro, Sep. 27–Oct. 1, 2021). Heidelberg: Springer, 2021. P. 123–135. (Lect. Notes Comput. Sci.; V. 13078).*
4. Andersen T., Hove J. H., Fagerholt K., Meisel F. Scheduling ships with uncertain arrival times through the Kiel Canal // *Maritime Transp. Res.* 2021. V. 2, ID 100008. 17 p.
5. Ben-Tal A., Nemirovski A. Robust optimization-methodology and applications // *Math. Program.* 2002. V. 92. P. 453–480.
6. García J., Peña A. Robust optimization: Concepts and applications // *Nature-inspired methods for stochastic, robust and dynamic optimization. London: IntechOpen, 2018. P. 7–22.*
7. Гордеев Э. Н., Леонтьев В. К. Общий подход к исследованию устойчивости решений в задачах дискретной оптимизации // *Журн. вычисл. математики и мат. физики.* 1996. Т. 36, № 1. С. 66–72.
8. Gurevsky E., Battaïa O., Dolgui A. Stability measure for a generalized assembly line balancing problem // *Discrete Appl. Math.* 2013. V. 161, No. 3. P. 377–394.

9. Rossi A., Gurevsky E., Battaia O., Dolgui A. Maximizing the robustness for simple assembly lines with fixed cycle time and limited number of workstations // *Discrete Appl. Math.* 2016. V. 208. P. 123–136.
10. Gurobi optimizer reference manual. Beaverton: Gurobi Optimization, 2021. Available at [www.gurobi.com/documentation/9.5/refman/index.html](http://www.gurobi.com/documentation/9.5/refman/index.html) (accessed May 16, 2022).
11. Гэри М., Джонсон Д. Вычислительные машины и труднорешаемые задачи. М.: Мир, 1982. 420 с.
12. 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.
13. Кулаченко И. Н., Кононова П. А. Гибридный алгоритм решения задачи маршрутизации буровых установок // *Дискрет. анализ и исслед. операций.* 2021. Т. 28, № 2. С. 35–59.
14. Mladenović N., Hansen P. Variable neighborhood search // *Comput. Oper. Res.* 1997. V. 24, No. 11. P. 1097–1100.
15. Smith A. E., Coit D. W. Penalty functions // *Handbook of evolutionary computation.* New York: Oxford Univ. Press, 1997. P. C5.2:1–C5.2:6.

Ратушный Алексей Владленович  
Кочетов Юрий Андреевич

Статья поступила  
4 мая 2022 г.  
После доработки —  
4 мая 2022 г.  
Принята к публикации  
6 мая 2022 г.

## A MATHEURISTIC FOR MINIMIZATION OF WAITING TIME FOR TRAILERS WITH UNCERTAIN ARRIVAL TIMES

A. V. Ratushnyi<sup>1, a</sup> and Y. A. Kochetov<sup>2, b</sup>

<sup>1</sup> Novosibirsk State University,

2 Pirogov Street, 630090 Novosibirsk, Russia

<sup>2</sup> Sobolev Institute of Mathematics,

4 Acad. Koptuyug Avenue, 630090 Novosibirsk, Russia

E-mail: <sup>a</sup>alexeyratushny@gmail.com, <sup>b</sup>jkochet@math.nsc.ru

**Abstract.** We present a new loading/unloading trailer scheduling problem for a logistics company. There is a building with several warehouses. Each warehouse stores pallets of different types of products in rooms for loading into trailers. Each warehouse has two gates. One gate is for the trailers, the other one is for two forklifts from the central zone (production line). It produces some products which must be placed in the warehouses according to the no wait rule. We assume that the arrival time for each trailer is uncertain. Our goal is to assign all trailers to warehouses and find a schedule for servicing all the trailers with the maximum stability radius under the total waiting time constraint. For this NP-hard problem, we design a two-stage matheuristic. First, we solve the simplified model using the Gurobi solver. Then, the VNS algorithm is used to return the solution into the feasible region taking into account the detailed information about pallets in each warehouse. We generate some test instances using real data from a Dutch logistics company. Computational results for 6 warehouses, 18 types of products, and 90 trailers are discussed. Tab. 4, illustr. 4, bibliogr. 15.

**Keywords:** stability radius, matheuristic, VNS, uncertainty.

## REFERENCES

1. E. Carrizosa and S. Nickel, Robust facility location, *Math. Methods Oper. Res.* **58**, 331–349 (2003).

---

The research is supported by the Russian Science Foundation (Project 21–41–09017).

English version: Journal of Applied and Industrial Mathematics **16** (3) (2022).



2. **E. Carrizosa, A. Ushakov, and I. Vasilyev**, Threshold robustness in discrete facility location problems: A bi-objective approach, *Optim. Lett.* **9**, 1297–1314 (2015).
3. **P. Borisovsky and O. Battaïa**, MIP-based heuristics for a robust transfer lines balancing problem, in *Optimization and Applications* (Proc. Int. Conf. OPTIMA 2021, Petrovac, Montenegro, Sep. 27–Oct. 1, 2021) (Springer, Heidelberg, 2021), pp. 123–135 (Lect. Notes Comput. Sci., Vol. 13078).
4. **T. Andersen, J. H. Hove, K. Fagerholt, and F. Meisel**, Scheduling ships with uncertain arrival times through the Kiel Canal, *Maritime Transp. Res.* **2**, ID 100008 (2021).
5. **A. Ben-Tal and A. Nemirovski**, Robust optimization-methodology and applications, *Math. Program.* **92**, 453–480 (2002).
6. **J. García and A. Peña**, Robust optimization: Concepts and applications, in *Nature-Inspired Methods for Stochastic, Robust and Dynamic Optimization* (In-techOpen, London, 2018), pp. 7–22.
7. **Eh. N. Gordeev and V. K. Leont’ev**, A general approach to the study of the stability of solutions in discrete optimization problems, *Zh. Vychisl. Mat. Mat. Fiz.* **36** (1), 66–72 (1996) [Russian] [*Comput. Math. Math. Phys.* **36** (1), 53–58 (1996)].
8. **E. Gurevsky, O. Battaïa, and A. Dolgui**, Stability measure for a generalized assembly line balancing problem, *Discrete Appl. Math.* **161** (3), 377–394 (2013).
9. **A. Rossi, E. Gurevsky, O. Battaïa, and A. Dolgui**, Maximizing the robustness for simple assembly lines with fixed cycle time and limited number of workstations, *Discrete Appl. Math.* **208**, 123–136 (2016).
10. Gurobi Optimizer Reference Manual (Gurobi Optimization, Beaverton, 2021). Available at [www.gurobi.com/documentation/9.5/refman/index.html](http://www.gurobi.com/documentation/9.5/refman/index.html) (accessed May 16, 2022).
11. **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]).
12. **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. Appl. Ind. Math.* **14** (2), 339–351 (2020)].
13. **I. N. Kulachenko and P. A. Kononova**, A hybrid algorithm for the drilling rig routing problem, *Diskretn. Anal. Issled. Oper.* **28** (2), 35–59 (2021) [Russian] [*J. Appl. Ind. Math.* **15** (2), 261–276 (2021)].
14. **N. Mladenović and P. Hansen**, Variable neighborhood search, *Comput. Oper. Res.* **24** (11), 1097–1100 (1997).
15. **A. E. Smith and D. W. Coit**, Penalty functions, in *Handbook of Evolutionary Computation* (Oxford Univ. Press, New York, 1997), pp. C5.2:1–C5.2:6.

Aleksey V. Ratushnyi  
Yury A. Kochetov

Received May 4, 2022  
Revised May 4, 2022  
Accepted May 6, 2022