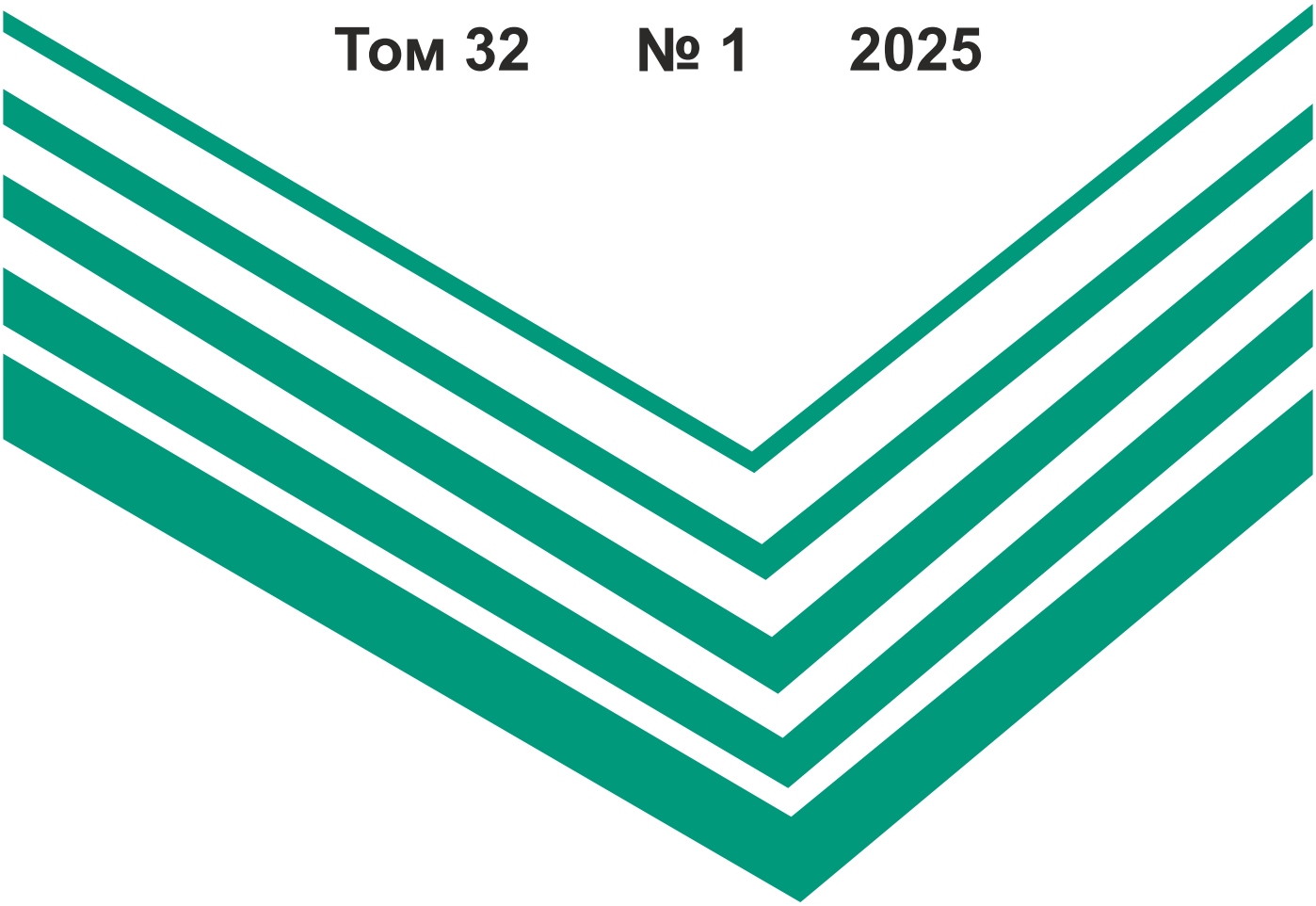
ISSN 2949-5598

# ДИСКРЕТНЫЙ АНАЛИЗ И ИССЛЕДОВАНИЕ ОПЕРАЦИЙ

Том 32 № 1 2025



Новосибирск
Издательство Института математики

ДИСКРЕТНЫЙ АНАЛИЗ И ИССЛЕДОВАНИЕ ОПЕРАЦИЙ
Январь–март 2025. Т. 32, № 1. С. 75–98

УДК 519.8 DOI: 10.33048/daio.2025.32.806

# ОБ ОДНОЙ ЗАДАЧЕ ОПТИМИЗАЦИИ РАЗМЕЩЕНИЯ ТОВАРОВ НА СКЛАДЕ

*К. О. Моторин*[1, a], *А. В. Пяткин*[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: [a] k.motorin@g.nsu.ru, [b] artem@math.nsc.ru

**Аннотация.** Рассматривается задача поиска расположения товаров на складе, при котором минимизируется суммарное время составления заказов из заданного списка. Доказано, что задача NP-трудна даже в простейшем частном случае. Построена математическая модель ЦЛП для этой задачи. Предложено два эвристических алгоритма её решения, работа которых проанализирована на случайно сгенерированных примерах. Табл. 3, ил. 9, библиогр. 17.

**Ключевые слова:** оптимизация склада, размещение товаров, NP-трудная задача, генетический алгоритм, поиск с запретами.

## Введение

Объектом исследования в этой статье является следующая задача оптимизации размещения товаров на складе. Имеется склад прямоугольной формы с расположенными в нём стеллажами, на которых хранятся товары. Между стеллажами есть проходы, движение по каждому из которых в целях безопасности может осуществляться только в одном направлении. Дан список заказов, которые нужно составить с помощью сборщиков заказов. Каждый заказ состоит из нескольких товаров, и один сборщик собирает только один заказ. Чтобы составить заказ, сборщик должен двигаться в строго заданных направлениях обхода склада мимо нужных стеллажей, с которых он собирает входящие в заказ товары. Если на пути основного обхода склада есть стеллажи, на которых нет нужных товаров, то путь иногда можно сократить, не подходя к этим стеллажам (если при этом не нарушается общее правило одностороннего

© К. О. Моторин, А. В. Пяткин, 2025

движения по проходам). Каждый вид товара лежит ровно в одной ячейке, и гарантируется, что товаров хватит на сборку всех заказов. Требуется найти такое расположение товаров на складе, при котором суммарное время перемещения сборщиков будет минимальным.

Эта задача возникла из дискуссии на индустриальной секции конференции MOTOR-2023 (Екатеринбург, Россия, 2–8 июля 2023 г.). Насколько известно авторам, ранее в такой постановке задача не рассматривалась.

Отметим, что задачам оптимизации работы склада посвящено очень много публикаций (см., например, обзор [1]). Можно выделить несколько направлений в зависимости от того, что оптимизируется:

- структура склада [2–4];
- взаимодействие склада с внешними предприятиями [5];
- обход склада при составлении заказов [6, 7];
- размещение товаров на складе [4, 8, 9].

Рассматриваемая задача лежит на стыке двух последних типов. Однако в задачах оптимизации обхода склада при составлении заказов размещение товаров обычно фиксировано, и требуется найти оптимальный маршрут движения по складу (при этом требования односторонности движения нет). В задачах же размещения товаров в расчёт принимается не заданный список заказов, а некие общие статистические соображения (например, складывать рядом наиболее часто используемые товары или размещать рядом товары из близких групп).

С математической точки зрения наиболее близкой к рассматриваемой задаче является квадратичная задача о назначениях (КЗОН) [10], в которой требуется так расположить предприятия, чтобы суммарное произведение объёмов грузопотока на расстояние между каждой парой предприятий было бы наименьшим. В отличие от обычной задачи о назначениях, КЗОН NP-трудна, так как её частным случаем является задача о гамильтоновом цикле. И хотя в рассматриваемой задаче зависимость целевой функции от способа размещения значительно более сложная, некоторые подходы к решению КЗОН оказались полезными и для исследуемой задачи.

Статья имеет следующую структуру. В разд. 1 приводится подробная постановка задачи, строится её математическая модель и доказывается, что задача NP-трудна даже в простейшем частном случае. В следующих двух разделах описываются эвристические алгоритмы, предложенные для решения задачи (гибридный генетический алгоритм представлен в разд. 2, а итеративный алгоритм поиска с запретами — в разд. 3). В разд. 4 анализируются полученные результаты. В заключении даются финальные комментарии.

## 1. Постановка задачи и её сложность

В этом разделе даются содержательная и математическая постановки задачи, а также исследуется её сложность.

**1.1. Содержательная постановка.** Имеется склад прямоугольной формы, составленный из стеллажей и проходов между ними. Также задан список заказов, каждый из которых состоит из нескольких видов товаров. Для каждого стеллажа известна его вместимость (число ячеек). В каждую ячейку можно положить лишь один вид товара; при этом считается, что количество товара данного вида в ячейке достаточно для удовлетворения (сборки) всех заказов. На складе задан односторонний порядок движения, позволяющий обойти все стеллажи. Сборщик заказа не обязан обходить весь склад: если на каких-то стеллажах нет товаров из собираемого заказа, он иногда имеет возможность сократить маршрут (но при этом он не имеет права двигаться против заданного направления обхода склада). За один обход сборщик собирает только один заказ. Требуется так разместить товары на складе, чтобы суммарный пройденный путь при составлении всех заказов был минимальный.

*Рис. 1.* Пример склада и нумерация точек размещения

На рис. 1 представлен пример такого склада. Чёрными стрелками обозначено направление обхода всего склада, в котором двигаются сборщики заказов. Пунктирные стрелки показывают места, где путь можно сократить, если на стеллажах основного пути нет нужных товаров для сбора текущего заказа. Каждый стеллаж разделён на две части, доступ к которым можно получить только с разных сторон.

Отметим, что при такой постановке задачи не возникает проблемы маршрутизации. Действительно, зная распределение товаров по ячейкам

стеллажей, можно однозначно восстановить кратчайший путь каждого из сборщика заказов. Для этого достаточно пронумеровать точки между стеллажами (будем называть их *точками размещения*) в порядке обхода склада (рис. 1) и задать матрицу кратчайших расстояний между ними. Тогда для вычисления кратчайшего пути обхода для сборщика заказов нужно сложить расстояния между точками размещения, которые нужно посетить для сборки заказа, в порядке возрастания, добавив расстояния от входа до первого стеллажа в пути и от последнего до выхода. Отметим, что в терминах точек размещения не имеет значения, в какой именно ячейке примыкающих к ней стеллажей находится нужный товар. Это значительно упрощает генерацию и формулировку примеров задачи для решения её с помощью алгоритмов.

**1.2. Математическая модель.** Формализуем приведённую выше содержательную задачу в виде следующей модели смешанного целочисленного линейного программирования.

Параметры задачи:

- $N$ — число точек размещения;
- $M$ — число видов товаров;
- $K$ — число заказов;
- $d_{ij}$ — кратчайшее расстояние от точки размещения $i$ до точки размещения $j$;
- $b_j$ — вместимость точки размещения $j$;
- $o_i^k = \begin{cases} 1, & \text{если в заказе } k \text{ есть товар } i, \\ 0 & \text{иначе.} \end{cases}$

Переменные задачи:

- $x_{ij} = \begin{cases} 1, & \text{если товар } i \text{ расположен в точке размещения } j, \\ 0 & \text{иначе;} \end{cases}$
- $p_j^k = \begin{cases} 1, & \text{если точка } j \text{ должна быть посещена в заказе } k, \\ 0 & \text{иначе;} \end{cases}$
- $f_{ij}^k$ — вспомогательная переменная, представляющая результат произведения $p_i^k p_j^k$;
- $l_{ij}^k = \begin{cases} 1, & \text{если для заказа } k \text{ точка } i \text{ посещается после точки } j, \\ 0 & \text{иначе.} \end{cases}$

Индексы:

- $M_{\text{idx}} = \{0, \ldots, M-1\}$ — индексы товаров;
- $N_{\text{idx}} = \{1, \ldots, N-2\}$ — индексы точек размещения; точки размещения 0 и $N-1$ соответствуют входу и выходу;
- $K_{\text{idx}} = \{0, \ldots, K-1\}$ — индексы заказов.

Целевая функция

$$\sum_{k=0}^{K-1}\sum_{i=0}^{N-2}\sum_{j=i+1}^{N-1} d_{ij}l_{ij}^k \to \min. \tag{1}$$

Ограничения:

$$\sum_{j=1}^{N-2} x_{ij} = 1, \quad i \in M_{\text{idx}}, \tag{2}$$

$$\sum_{i=0}^{M-1} x_{ij} \leqslant b_j, \quad j \in N_{\text{idx}}, \tag{3}$$

$$p_j^k \geqslant x_{ij}o_i^k, \quad i \in M_{\text{idx}},\ j \in N_{\text{idx}},\ k \in K_{\text{idx}}, \tag{4}$$

$$p_0^k = 1, \quad k \in K_{\text{idx}}, \tag{5}$$

$$p_{N-1}^k = 1, \quad k \in K_{\text{idx}}, \tag{6}$$

$$f_{ij}^k \leqslant p_i^k, \quad i,j \in N_{\text{idx}},\ k \in K_{\text{idx}}, \tag{7}$$

$$f_{ij}^k \leqslant p_j^k, \quad i,j \in N_{\text{idx}},\ k \in K_{\text{idx}}, \tag{8}$$

$$p_i^k + p_j^k - f_{ij}^k \leqslant 1, \quad i,j \in N_{\text{idx}},\ k \in K_{\text{idx}}, \tag{9}$$

$$l_{ij}^k \leqslant f_{ij}^k, \quad i,j \in N_{\text{idx}},\ i < j,\ k \in K_{\text{idx}}, \tag{10}$$

$$l_{ij}^k \leqslant 1 - p_l^k, \quad i,j,l \in N_{\text{idx}},\ i < l < j,\ k \in K_{\text{idx}}, \tag{11}$$

$$l_{ij}^k \geqslant f_{ij}^k - \sum_{l=i+1}^{l=j-1} p_l^k, \quad i,j \in N_{\text{idx}},\ i < j-1, \tag{12}$$

$$l_{jj+1}^k \geqslant f_{jj+1}^k, \quad j \in \{0,\ldots,N-2\},\ k \in K_{\text{idx}}. \tag{13}$$

Целевая функция (1) определяет суммарную длину всех маршрутов сборки заказов. Ограничение (2) гарантирует, что один вид товара расположен ровно в одной точке размещения. Неравенства (3) задают максимальную вместимость для каждой точки размещения. Ограничения (4)–(6) нужны для определения вспомогательной переменной, которая показывает, нужно ли посещать точку размещения для конкретного заказа. В модели предполагается, что вход на склад и выход со склада — это точки размещения 0 и $N-1$ соответственно, с нулевой вместимостью, которые обязательно нужно посетить во всех заказах. Неравенства (7)–(9) задают переменную $f_{ij}^k$, которая нужна для индикации, нужно ли посещать обе точки размещения $i,j$ в заказе $k$. Вспомогательная переменная $f_{ij}^k$ является результатом линеаризации произведения $p_i^k p_j^k$. Ограничения (10)–(13) задают переменную $l_{ij}^k$, участвующую в целевой

функции. Эта переменная равна единице, если в заказе $k$ для сборки заказа требуется посетить подряд точки размещения $i$ и $j$ (т. е. во всех промежуточных точках из интервала $(i, j)$ нет нужных товаров для $k$-го заказа). Тогда для подсчёта суммарного пути сборщиков заказов требуется просуммировать все переменные $l_{ij}^k$, умноженные на соответствующие кратчайшие длины между точками размещения.

**1.3. Сложность задачи.** Покажем, что задача NP-трудна и в простейшем частном случае, когда на складе есть всего два стеллажа и каждый заказ содержит лишь два вида товара. Нам потребуется следующая известная NP-трудная задача [11, 12].

**Задача о бисекции.** *Дан граф $G = (V, E)$ на $2n$ вершинах. Найти в $G$ минимальную бисекцию, т. е. такое разбиение $V = V_1 \cup V_2$, что $|V_1| = |V_2| = n$ и число рёбер, соединяющих $V_1$ и $V_2$, минимально.*

**Теорема 1.** *Задача о размещении товаров на складе в случае, когда имеется два стеллажа и все заказы состоят из двух видов товаров,* NP*-трудна.*

Доказательство. Рассмотрим произвольный пример задачи о бисекции (граф $G = (V, E)$ на $2n$ вершинах) и построим по нему следующий пример задачи о размещении товаров на складе. Сопоставим каждой вершине графа $G$ один вид товара, а каждому ребру — заказ, товары в котором соответствуют инцидентным этому ребру вершинам. Рассмотрим склад с двумя стеллажами длины $L$, вместимость каждого из которых равна $n$, причём размещать товары можно только с внешних сторон стеллажа. Также между внутренними сторонами стеллажей есть проход, позволяющий вернуться от конца первого стеллажа к началу второго. Ширина стеллажей считается незначительной. Схема такого склада представлена на рис. 2. Склад имеет три точки размещения, вторая из которых имеет вместимость 0. Очевидно, что каждое распределение товаров по стеллажам взаимно однозначно соответствует некоторой бисекции графа.

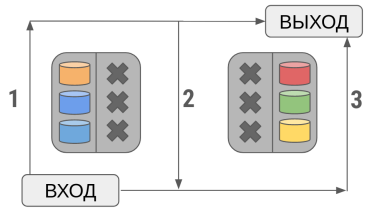



*Рис. 2.* Схема склада из теоремы 1

Нетрудно видеть, что если оба товара в заказе лежат на одном стеллаже, то маршрут сборщика имеет длину $L$, а если на разных, то — $3L$, поэтому для минимизации пройденного пути нужно расположить товары так, чтобы максимизировать число заказов, для сборки которых требуется посетить только один стеллаж. Тогда в соответствующей бисекции графа число рёбер, концы которых лежат в разных множествах, будет минимально. И наоборот, если известна минимальная бисекция графа, то предметы, соответствующие вершинам из одного множества, нужно расположить на одном стеллаже. В таком случае число заказов, для которых потребуется посетить два стеллажа, будет минимальным.

Таким образом, по любому примеру задачи о минимальной бисекции можно построить эквивалентный пример исследуемой задачи о размещении товаров на складе с двумя стеллажами и заказами из двух видов товаров. Из этого следует, что задача о размещении товаров на складе NP-трудна в таком частном случае. Теорема 1 доказана.

## 2. Гибридный генетический алгоритм

В качестве первого алгоритма для исследуемой проблемы был выбран гибридный генетический алгоритм.

**2.1. Структура алгоритма.** Генетический алгоритм является вероятностным алгоритмом поиска, основанным на биологических принципах живой природы. Идея алгоритма заключается в генерации начальной популяции — набора решений исследуемой задачи, называемых индивидами, и последующем улучшении этой популяции. На каждой

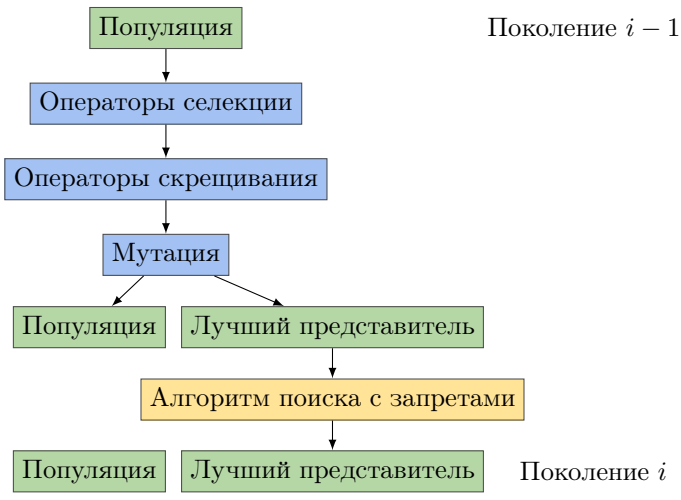



*Рис. 3.* Схема гибридного генетического алгоритма

итерации алгоритма представители популяции изменяются с помощью операций селекции, скрещивания и мутации, образуя новое поколение, в котором может найтись решение с лучшим значением целевой функции. Однако, в процессе работы было экспериментально обнаружено, что применение генетического алгоритма само по себе не даёт желаемых результатов, поэтому в качестве дополнительной эвристики выбран алгоритм поиска с запретами, который на каждой итерации улучшает рекордного индивидуума в популяции. Общая схема алгоритма представлена на рис. 3.

Каждое решение оптимизационной задачи (индивид популяции) кодируется $M$-мерным вектором, $i$-я координата которого указывает номер точки размещения, в которой расположен $i$-й вид товара. Тогда для допустимости решения необходимо и достаточно, чтобы каждое значение $j$ встречалось не более чем в $b_j$ координатах вектора. В алгоритме используется два оператора скрещивания и один оператор мутации.

1. Оператор равномерного скрещивания. Сначала потомок наследует одинаковые признаки от обоих родителей, затем остальные признаки наследуются равномерно. Если добавление признака нарушает допустимость решения, то текущая позиция пропускается. Далее все пропущенные позиции заполняются оставшимися свободными точками размещения случайным образом. Применение такого оператора требует $O(M)$ операций. На рис. 4 изображено схематичное представление данного оператора (во всех примерах ниже предполагается, что все $b_j = 2$).

2. Оператор двухточечного скрещивания. Сначала случайным образом выбираются две точки скрещивания — два целых числа $i, j \in [1, M]$, причём $i \leqslant j$. Затем потомок наследует интервал $[i, j]$ от первого родителя, а оставшиеся координаты наследуются от второго родителя, если

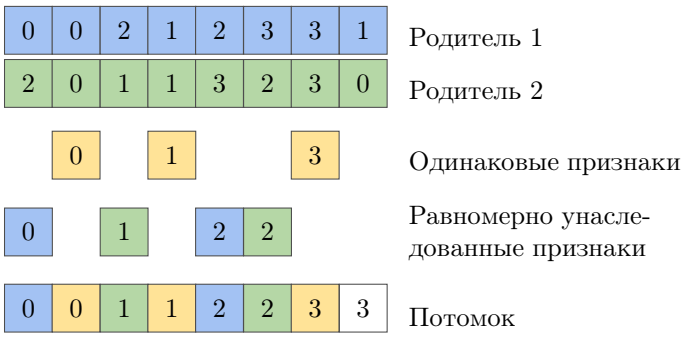



*Рис. 4.* Оператор равномерного скрещивания

*Рис. 5.* Оператор двухточечного скрещивания

это не нарушает условия допустимости решения. Далее те ячейки решения, для которых значения остались неопределёнными, заполняются случайным образом из оставшихся допустимых точек размещения. Этот оператор также требует $O(M)$ операций. Схема оператора двухточечного скрещивания представлена на рис. 5.

Отметим, что также рассматривался оператор множественного скрещивания, в котором выбирается несколько родителей и координаты вектора заполняются в случайно выбранном порядке по принципу «выбираем наиболее часто встречающееся значение этой координаты среди родителей». Известно, что такой оператор скрещивания показывает хорошие результаты для КЗОН [13]. Однако в настоящей работе было решено отказаться от его использования, так как на больших примерах время работы алгоритма значительно росло при отсутствии заметного улучшения значения целевой функции.

3. В качестве оператора мутации была использована многократная последовательная замена значений двух случайных координат исходного решения друг другом . На каждом шаге случайным образом выбираются два целых числа $i, j \in [1, M]$, происходит обмен значений этих координат текущего вектора и вычисление целевой функции. Результирующим решением считается мутировавший вектор с минимальным значением целевой функции (при этом целевая функция исходного решения не учитывается). На рис. 6 представлена схема данного оператора.

Отметим, что число шагов оператора мутации регулируется специальным параметром — степенью мутации $\mu$, который варьируется между минимальным и максимальным значениями $\mu_{\min}$ и $\mu_{\max}$. На первой итерации алгоритма этот параметр равен минимальному значению, чтобы дать возможность алгоритму найти лучшее решение в текущей области

*Рис. 6.* Оператор мутации

локального минимума. На каждой следующей итерации алгоритма значение параметра увеличивается на 1, каждый раз всё больше изменяя представителей популяции. Таким образом осуществляется поиск решения в отдалённых областях пространства решений. Степень мутации возвращается к минимальному значению, если на предыдущей итерации алгоритма либо значение параметра было максимальным, либо произошло обновление лучшего найденного решения. Похожая идея использовалась в [14] при разработке итерационного алгоритма локального поиска для КЗОН.

**2.2. Локальное улучшение.** На каждой итерации генетического алгоритма рекордный представитель популяции (в смысле значения целевой функции) улучшается с помощью алгоритма поиска с запретами. Такой алгоритм был впервые предложен в [15] как модифицированная версия алгоритма локального поиска, в которой имеется список запретов, не позволяющий алгоритму выбирать некоторые решения из рассматриваемой окрестности. Такой механизм позволяет не зацикливаться в областях локального минимума. Алгоритм поиска с запретами зарекомендовал себя во многих задачах дискретной оптимизации и имеет множество модификаций [16].

В качестве окрестности, в которой происходит поиск нового решения, выбирается множество всех решений, получающихся переменой различных точек размещения ровно двух товаров, т. е.

$$N(x_1,\dots,x_M) = \{(x'_1,\dots,x'_M) \mid \exists\, i,j\colon x_i \neq x_j,\\ x'_i = x_j,\ x'_j = x_i,\ x'_k = x_k \text{ при } k \neq i,j\}.$$

Эта окрестность имеет мощность $(M^2 - M)/2$. Будем использовать обозначение $x' = \mathrm{swap}(x,i,j)$, если $x'_i = x_j, x'_j = x_i$ и $x'_k = x_k$ для всех $k \neq i,j$. Для реализации механизма списка запретов используется матрица $T$ размера $M \times N$, элемент $T_{i,j}$ которой показывает, до какой итерации алгоритма запрещено назначение элемента $i$ в точке размещения $j$. После выбора нового текущего решения из окрестности новые назначения двух изменённых элементов решения попадают под запрет до итерации $\mathrm{iter} + h$, где iter — номер текущей итерации, а $h$ — время запрета. Значение параметра $h$ выбирается случайным образом из отрезка $[h_{\min}, h_{\max}]$ для большей стабильности алгоритма.

Так как время просмотра всей окрестности равно $O(M^2)$, на каждой итерации просматривается не вся окрестность, а только её часть. Для этого в алгоритме задаётся специальный параметр $p$ — вероятность, с которой очередное решение из окрестности будет взято в рассмотрение. Такой подход уменьшает трудоёмкость одной итерации алгоритма и может повысить вероятность нахождения глобального минимума.

Приведём формальное описание всего алгоритма. Отметим, что процедура поиска с запретами вынесена в отдельный алгоритм, поскольку она будет использована и в следующем разделе.

**Алгоритм 1.** Гибридный генетический алгоритм.

Вход: число товаров $M$, размер популяции $P$, минимальная и максимальная степени мутации $\mu_{\min}$ и $\mu_{\max}$, минимальное и максимальное время запрета $h_{\min}$ и $h_{\max}$.

Шаг 1. Выбрать начальную популяцию $S = \{s_1,\dots,s_P\}$ и найти в ней рекордное решение $s^*$ с минимальным значением целевой функции $\mathrm{obj}(s^*)$. Задать $\mu = \mu_{\min}$. Положить $g = 0$. Пока не выполнен критерий остановки, выполнять шаги 2–7.

Шаг 2. Применить операторы скрещивания.

Шаг 3. Применить оператор мутации с числом шагов $\mu$. Найти в популяции индивида $s'$ с минимальным значением целевой функции.

Шаг 4. Применить алгоритм поиска с запретами (Алгоритм 2) с входом $M, s', h_{\min}$ и $h_{\max}$ и получить улучшенное решение $x^*$.

Шаг 5. Положить $s'' = x^*$.

Шаг 6. Если $\mu = \mu_{\max}$, то положить $\mu = \mu + 1$, иначе $\mu = \mu_{\min}$.

Шаг 7. Если $\mathrm{obj}(s'') < \mathrm{obj}(s^*)$, то положить $s^* = s''$ и $\mu = \mu_{\min}$.

Выход: решение $s^*$.

**Алгоритм 2.** Локальный поиск с запретами.

Вход: число товаров $M$, начальное решение $s'$, минимальное и максимальное время запрета $h_{\min}$ и $h_{\max}$.

Шаг 1. Задать текущее решение $x = s'$, лучшее решение $x^* = s'$, матрицу запретов $T = 0$, вероятность выбора элемента окрестности $p \in (0, 1)$ и номер итерации $\mathrm{iter} = 0$. Пока не выполнен критерий остановки, выполнять шаги 2–8.

Шаг 2. Выбрать $h = \mathrm{rand}(h_{\min}, h_{\max})$. Положить $i_{\mathrm{best}} = M+1$, $j_{\mathrm{best}} = M + 1$, $R = \infty$.

Шаг 3. Для всех $i = 1, \ldots, M - 1$ и $j = i + 1, \ldots, M$, если $x_i \neq x_j$ и $\mathrm{rand}(0, 1) < p$, то выполнить шаги 4–6.

Шаг 4. Задать решение $x' = \mathrm{swap}(x, i, j)$ и вычислить $\mathrm{obj}(x')$.

Шаг 5. Если $(\mathrm{obj}(x') < \mathrm{obj}(x^*))$ или $(\mathrm{obj}(x') < R$ и $T_{i,x(j)} \leqslant \mathrm{iter}$ и $T_{j,x(i)} \leqslant \mathrm{iter})$, то положить $i_{\mathrm{best}} = i$, $j_{\mathrm{best}} = j$, $R = \mathrm{obj}(x')$.

Шаг 6. Если $\mathrm{obj}(x') < \mathrm{obj}(x^*)$, то положить $x^* = x'$.

Шаг 7. Если $i_{\mathrm{best}} \leqslant M$, то положить $x = \mathrm{swap}(x, i_{\mathrm{best}}, j_{\mathrm{best}})$, а также $T_{i_{\mathrm{best}},x(j_{\mathrm{best}})} = \mathrm{iter} + h$ и $T_{j_{\mathrm{best}},x(i_{\mathrm{best}})} = \mathrm{iter} + h$.

Шаг 8. Положить $\mathrm{iter} = \mathrm{iter} + 1$.

Выход: решение $x^*$ и номер последней итерации iter.

Отметим также технический момент, позволяющий ускорить пересчёт значения целевой функции при перестановке двух элементов. Во-первых, путь сбора заказа изменится только у тех сборщиков, у которых в заказе имеется только один тип товара из перестановки. Во-вторых, значение целевой функции для заказа изменится только при условии, что на точке размещения, где находится товар из перестановки, нет других товаров из заказа, либо в текущем маршруте для сборщика нет точки размещения, на место которой был назначен товар из перестановки, поэтому хранение информации о текущих путях обхода склада для каждого заказа организовано в виде списка, в элементе которого указан номер точки размещения для посещения и число типов товаров из заказа на этой точке. Тогда для пересчёта значений целевой функции нужно знать старое и новое расположения элемента решения, а также предыдущее значение целевой функции. Всего для подсчёта нового значения целевой функции потребуется $O(KL)$ операций, где $K$ — это число всех заказов, а $L$ — максимальная длина одного заказа.

### 3. Итеративный алгоритм поиска с запретами

Так как модель смешанного целочисленного линейного программирования для исследуемой задачи позволяет найти решение с помощью специальных программ-решателей только для примеров малой размерности, требуются дополнительные алгоритмы, с которыми можно сравнивать полученные результаты. В этом разделе представлен второй разработанный алгоритм для исследуемой проблемы, а именно алгоритм итеративного поиска с запретами. В нём используются три различные модификации алгоритма 2 в качестве процедуры локального улучшения на каждой итерации. Алгоритм заключается в итеративном применении процедур интенсификации и диверсификации, которые используются для борьбы с проблемами зацикливания алгоритма в областях локальных минимумов. Первый вариант такого поиска был предложен в [17]. Фаза интенсификации нацелена на поиск решения в специальном «многообещающем» регионе пространства решений задачи, который находится рядом с текущим найденным локальным минимумом, в то время как фаза диверсификации позволяет выйти из текущей области локального минимума в сторону новых неизученных регионов. Отметим, что аналогичный подход был успешно применён к КЗОН, где с помощью него удалось найти новые лучшие решения для примеров, часто используемых в тестировании алгоритмов для данной задачи [14].

Алгоритм состоит из следующих этапов (см. схему на рис. 7).

1. Выбрать случайное решение в качестве текущего.

2. Текущее решение улучшается с помощью модификации алгоритма 2, в которой вычисляется матрица частоты назначения товаров в точки размещения.

3. На фазе интенсификации происходит поиск нового улучшенного решения в «хороших» областях пространства решения вблизи текущего значения локального минимума.

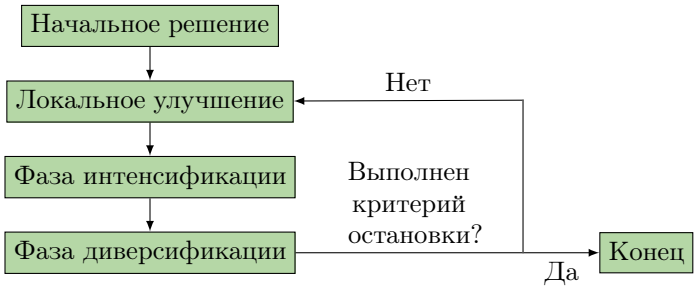



*Рис. 7.* Схема алгоритма итеративного локального поиска

4. После фазы интенсификации применяется фаза диверсификации для выхода из текущей области локального минимума. Если критерий остановки не выполнен, то алгоритм переходит на этап 2.

Опишем подробнее фазы интенсификации и диверсификации.

Для реализации фазы интенсификации алгоритм поиска с запретами на втором этапе заполняет матрицу $F$ размера $M \times N$, значение $F_{i,j}$ которой показывает, как часто вид товара под номером $i$ находился в точке размещения $j$. В фазе интенсификации в рекордном решении фиксируется часть компонент, которые чаще всего находились на местах лучшего решения. Доля фиксированных компонент решения задаётся специальным параметром $P_I$. После этого на рекордном решении запускается модификация алгоритма поиска с запретами, в которой перестановки, содержащие зафиксированные элементы решения, запрещены. Это позволяет сократить размерность задачи, тем самым ускорив работу алгоритма поиска с запретами, а также сосредоточиться на поиске лучшего решения вблизи текущего найденного локального минимума. Параметр rate задаёт порог, по которому отбираются самые частые назначения в лучшем решении алгоритма. Этот порог уменьшается до тех пор, пока число фиксированных элементов решения не достигнет $MP_I$.

В фазе диверсификации используется похожая стратегия. Сначала задаётся матрица доступных назначений $A$ размера $M \times N$ (т. е. $i$-й вид товара можно разместить в $j$-й точке тогда и только тогда, когда $A_{i,j} = 1$). Назначения выбираются из вариантов, которые находились в текущих решениях реже всего. Специальный параметр $P_D$ регулирует долю возможных назначений типов товаров на точки размещения, которые будет разрешены в матрице $A$. Далее запускается модификация алгоритма 2, в которой разрешены только назначения из матрицы $A$, а число итераций выбирается небольшим. Это позволяет продолжить поиск лучшего решения в ранее не рассмотренных областях. Матрица $A$ заполняется следующим образом. Выбирается начальное значение порога rate, которое ограничивает частоту назначения типов товаров. Если доля назначений товара $i$ на место $j$ от общего числа итераций меньше этого порога, то такое назначение становится разрешённым. Параметр rate увеличивается на малое значение до тех пор, пока число разрешённых назначений не достигнет $NMP_D$.

Приведём формальное описание итеративного алгоритма поиска с запретами. Шаги 4–8 соответствуют фазе интенсификации, а шаги 9–13 — фазе диверсификации.

**Алгоритм 3.** Итеративный алгоритм поиска с запретами.

Вход: число товаров $M$ и точек назначения $N$, минимальное и максимальное время запрета $h_{\min}$ и $h_{\max}$, доли фиксированных элементов

решения фазы интенсификации $P_I \in (0,1)$, доля допустимых назначений в фазе диверсификации $P_D \in (0,1)$, начальное решение $x_0$.

Шаг 1. Задать текущее решение $x = x_0$ и рекордное решение $x^* = x_0$.

Шаг 2. Пока не выполнен критерий остановки, применять шаги 3–13.

Шаг 3. Задать матрицу $F = 0$ и применить модификацию алгоритма 2 с входом $M$, $x$, $h_{\min}$ и $h_{\max}$, в которой на шаге 8 выполняется дополнительное действие: для всех $i = 1, \ldots, M$ положить $F_{i,x(i)} = F_{i,x(i)} + 1$. На выходе получим рекордное решение $x^*$, а также число итераций $I$.

Шаг 4. Задать вектор фиксированных товаров $\mathrm{fix} = 0$ и их число $M_{\mathrm{fix}} = 0$, а также порог $\mathrm{rate} = 0{,}95$.

Шаг 5. Пока $M_{\mathrm{fix}} < MP_I$ выполнять шаги 6–7.

Шаг 6. Для всех $i = 1, \ldots, M$ если $F_{i,x^*(i)} > I \cdot \mathrm{rate}$ и $\mathrm{fix}[i] = 0$, то положить $\mathrm{fix}[i] = 1$ и $M_{\mathrm{fix}} = M_{\mathrm{fix}} + 1$.

Шаг 7. Положить $\mathrm{rate} = \mathrm{rate} - 0{,}05$.

Шаг 8. Применить следующую модификацию алгоритма 2: на вход подаются $M$, $x$, $h_{\min}$ и $h_{\max}$, а также вектор fix; при этом на шаге 3 дополнительно проверяются условия $\mathrm{fix}[i] = 0$ и $\mathrm{fix}[j] = 0$.

Шаг 9. Задать матрицу доступных назначений $A = 0$ и их число $N_A = 0$, а также порог $\mathrm{rate} = 0{,}05$.

Шаг 10. Пока $N_A < NMP_D$ выполнять шаги 11–12.

Шаг 11. Для всех $i = 1, \ldots, M$ и $j = 1, \ldots, N$, если $F_{i,x^*(i)} < I \cdot \mathrm{rate}$ и $A_{i,j} = 0$, то положить $A_{i,j} = 1$ и $N_A = N_A + 1$.

Шаг 12. Положить $\mathrm{rate} = \mathrm{rate} + 0{,}05$.

Шаг 13. Применить следующую модификацию алгоритма 2: на вход подаются $M, x, h_{\min}$ и $h_{\max}$, а также матрица $A$; при этом на шаге 3 дополнительно проверяются условия $A_{i,x^*(j)} = 1$ и $A_{j,x^*(i)} = 1$.

Выход: решение $x^*$.

## 4. Результаты

В этом разделе представлены описание способа генерации примеров, на которых производилось тестирование алгоритмов, выбранные параметры каждого из алгоритмов и полученные численные результаты.

**4.1. Тестовые примеры.** Так как исследуемая задача ранее не изучалась, общедоступных библиотек примеров для тестирования алгоритмов её решения нет, поэтому авторами реализована программа, которая генерирует примеры разной размерности для задачи. На вход программе подаётся число видов товаров, размеры стеллажей (число ячеек для

хранения), число стеллажей по горизонтали и по вертикали, число заказов, а также максимальный размер одного заказа. Товары в одном заказе заполняются случайным образом из всех доступных видов товаров для данного примера. Генератор примеров формирует представление задачи в терминах точек размещения, задаёт их вместимость, а также матрицу кратчайших расстояний между ними. Тестовые примеры генерировались в соответствии с несколькими возможностями изменения конфигурации: изменение размеров одного стеллажа, изменение числа стеллажей, изменение размера и числа заказов. Пример представляет собой три файла: информация о кратчайших расстояниях между точками размещения, входом и выходом, вместимость каждой из точки размещения, набор заказов для сборки.

Названия примеров закодированы в виде $a \times b_c_d$, где $a$ — число стеллажей по вертикали, $b$ — число стеллажей по горизонтали, $c$ — вместимость одного стеллажа, $d$ — число заказов. Диапазон числа товаров в одном заказе регулировался одинаковым образом для всех примеров в пределах до 15% от $M$, поэтому не задан в названии.

Описанные алгоритмы реализованы на языке программирования C. Сравнительные эксперименты проводились на компьютере с процессором AMD Ryzen 5 5600U 2,3 ГГц и 16 ГБ ОЗУ под управлением операционной системы Linux.

В ходе численных экспериментов сделана попытка решения исследуемой задачи с помощью программы-оптимизатора, для которой построена модель смешанного целочисленного линейного программирования. Так как число переменных и ограничений даже для примеров малой размерности не позволили запустить оптимизацию с помощью таких пакетов, как CPLEX и GUROBI с пробной лицензией, проведён эксперимент с использованием оптимизатора с открытым исходным кодом CBC. На примере, состоящем из 12 точек размещения вместимости 1 или 2 и пяти заказов размера от 5 до 9, оптимизатору удалось найти оптимальное решение за время, примерно равное 4 ч. Разработанным алгоритмам для нахождения решения с такой же целевой функцией потребовалось меньше секунды. В связи с этим сравнение результатов работы алгоритмов с оптимизаторами не проводилось.

**4.2. Гибридный генетический алгоритм.** Рассмотрено несколько вариантов гибридного генетического алгоритма, отличающихся выбором стратегии селекции (отбора особей для скрещивания).

1. Стратегия, в которой новое поколение формируется с помощью скрещивания «элитного» представителя (индивида с наименьшим значением целевой функции) с остальными членами популяции. В этом случае в новую популяцию всегда наследуются признаки элитного родителя, а для разнообразия популяции используется оператор мутации.

2. Стратегия, использующая для выбора индивидуумов метод турнирного отбора, который работает следующим образом. Каждый раз из популяции выбирается случайным образом заданное число особей (в нашем случае 2), и лучшая из них записывается в промежуточный массив родителей. Процесс повторяется, пока число родителей в массиве не станет равным числу особей в популяции. После этого особи для скрещивания выбираются из массива случайным образом. Преимущество данного метода селекции заключается в том, что он не требует дополнительного пересчёта значений целевой функции и упорядочивания индивидуумов.

3. Стратегия, использующая гибрид двух вышеописанных подходов: сначала формировался один набор индивидуумов первым методом, а затем этот набор принимался как популяция и изменялся вторым методом селекции. Это используется для большей диверсификации алгоритма и большей разности между представителями популяции.

Также был запрограммирован стандартный генетический алгоритм без использования алгоритма поиска с запретами.

На рис. 8 представлено сравнение работы разных подходов к выбору стратегии алгоритма. Как можно заметить, вторая стратегия (использующая только турнирный отбор) показала себя лучшим образом. Стандартный генетический алгоритм с разными стратегиями селекции

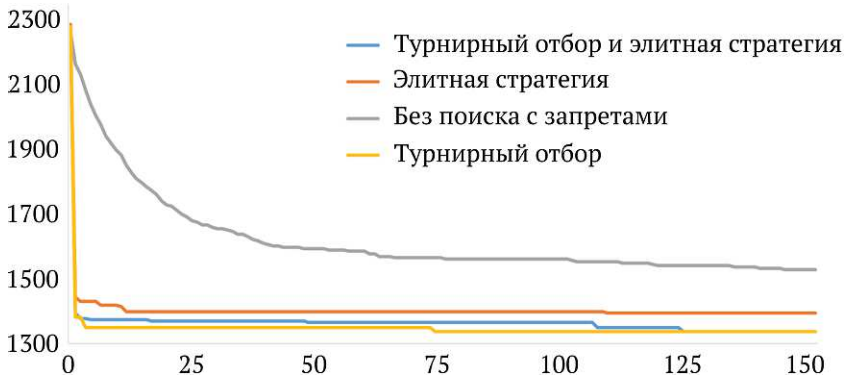



*Рис. 8.* Сравнение разных вариантов генетического алгоритма: по горизонтали — число итераций, по вертикали — значение целевой функции

не смог достичь значения, близкого к лучшему найденному даже на большем числе итераций.

*Таблица 1*

**Параметры гибридного генетического алгоритма**

| Параметр | Значение |
|---|---|
| Размер популяции, $G$ | $M$ |
| Число особей в турнире, $n_{\text{tourn}}$ | 2 |
| Число итераций без улучшения для всего алгоритма, $I_{\text{all}}$ | $[0{,}2M]$ |
| Минимальное время запрета, $h_{\min}$ | $[0{,}1M]$ |
| Максимальное время запрета, $h_{\max}$ | $[0{,}3M]$ |
| Минимальная степень мутации, $\mu_{\min}$ | $[0{,}1M]$ |
| Максимальная степень мутации, $\mu_{\max}$ | $[0{,}8M]$ |
| Параметр рандомизированной окрестности для TS, $p$ | 0,25 |
| Число итераций без улучшения для TS, $I_{\text{TS}}$ | $M$ |

В табл. 1 представлены подобранные параметры гибридного генетического алгоритма. Диапазон времени запрета выбирался из промежутков разной длины от $0{,}1M$ до $0{,}5M$ по лучшему среднему результату за пять запусков алгоритма поиска с запретами на примере 5x5_6_500. Диапазон степени мутации выбирался из промежутков $(0{,}1M; 0{,}2M), \ldots, (0{,}9M; M)$ с шагом $0{,}1M$ при таких же условиях на запусках гибридного генетического алгоритма. В качестве критерия остановки было использовано число итераций без улучшения значения целевой функции.

**4.3. Итеративный алгоритм поиска с запретами.** В табл. 2 представлены подобранные параметры итеративного алгоритма поиска с запретами. Параметр $P_D$ подбирался из диапазона $[0{,}05; 0{,}3]$ с шагом 0,05 при фиксированных остальных параметрах на одном из сгеренированных примеров. Выбор диапазона обоснован тем, что для диверсификации требуется использовать самые редкие возможные назначения, а при увеличении доли допустимых назначений туда начнут попадать уже использованные варианты изменения решения. Критерием остановки в фазе диверсификации являлось достижение итерации с номером $I_D$. Параметр $P_I$ подбирался из диапазона $[0{,}1; 0{,}5]$ с шагом 0,1 при таких же условиях. В качестве критерия остановки выступало максимально допустимое число итераций без улучшения рекорда.

На рис. 9 представлен график работы итеративного алгоритма поиска с запретами на примере для 25 стеллажей с вместимостью 20 и 100 заказами. Из графика видно, что сначала алгоритм работает как стандартный алгоритм локального поиска. С течением времени он начинает

*Таблица 2*

**Параметры итеративного алгоритма поиска с запретами**

| Параметр | Значение |
|---|---|
| Минимальное время запрета, $h_{\min}$ | $[0{,}1M]$ |
| Максимальное время запрета, $h_{\max}$ | $[0{,}3M]$ |
| Параметр фазы диверсификации, $P_D$ | 0,15 |
| Параметр фазы интенсификации, $P_I$ | 0,3 |
| Параметр рандомизированной окрестности, $p_{\text{TS}}$ | 0,25 |
| Число итераций без улучшения для всего алгоритма, $I_{\text{all}}$ | 10 |
| Число итераций без улучшения для TS, $I_{\text{TS}}$ | $M$ |
| Число итераций на фазе диверсификации, $I_D$ | $[0{,}1M]$ |

попадать в локальные оптимумы, но из некоторых из них удаётся выйти и получить значения с меньшей целевой функцией. Например, перед фазой интенсификации, отмеченной зелёным цветом, алгоритм поиска с запретами зациклился. Однако в фазе интенсификации благодаря уменьшению размерности задачи удаётся получить решения с лучшими значениями целевой функции. На этапе диверсификации значение целевой функции резко возрастает, но затем алгоритм поиска с запретами запускается снова, и ему удаётся обновить рекордное решение.

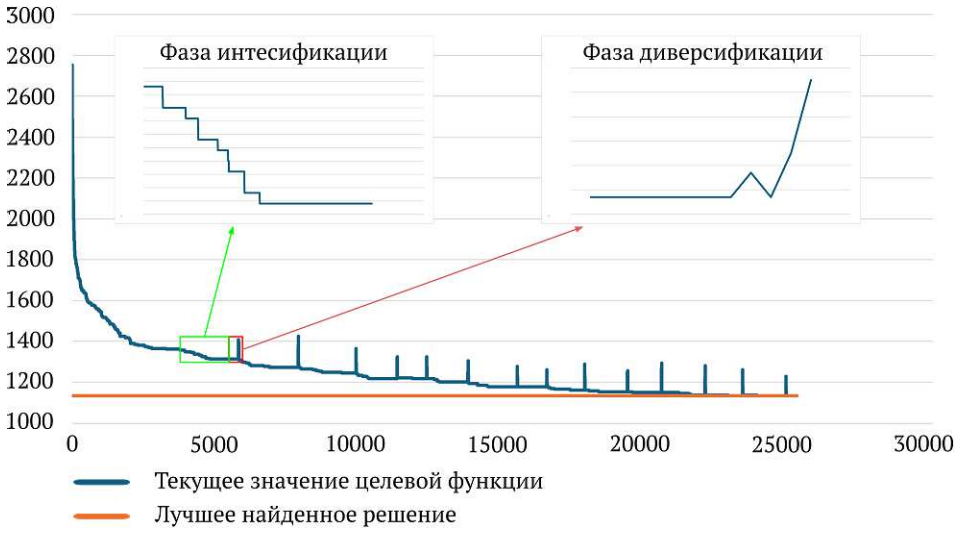



*Рис. 9.* Поведение алгоритма 3 на примере 5x5_20_100:
по горизонтали — номер итерации, по вертикали — значение целевой функции

*Таблица 3*

**Численные результаты**

| Пример | GA + TS | | | | ITS | | | |
|---|---|---|---|---|---|---|---|---|
| | мин. | сред. | макс. | $t$, с | мин. | сред. | макс. | $t$, с |
| 3x3_10_10 | 250 | 250 | 250 | 4,3 | 250 | 250 | 250 | 1,7 |
| 3x3_10_50 | 1 006 | 1 006 | 1 006 | 24 | 1 006 | 1 006 | 1 006 | 7 |
| 3x3_10_100 | 2 592 | 2 592 | 2 592 | 43 | 2 592 | 2 592 | 2 592 | 12 |
| 3x3_10_500 | 12 324 | 12 334 | 12 348 | 84 | 12 324 | 12 331 | 12 360 | 32 |
| 3x3_50_10 | 142 | 142 | 142 | 58 | 142 | 142 | 142 | 38 |
| 3x3_50_50 | 710 | 713 | 714 | 512 | 712 | 716 | 724 | 134 |
| 3x3_50_100 | 1 540 | 1 540 | 1 540 | 2 868 | 1 540 | 1 540 | 1 540 | 809 |
| 3x3_50_500 | 10 634 | 10 636 | 10 642 | 6 588 | 10 636 | 10 648 | 10 658 | 3 192 |
| 5x5_6_10 | 218 | 218 | 218 | 28 | 218 | 218 | 218 | 6 |
| 5x5_6_100 | 4 040 | 4 040 | 4 040 | 412 | 4 040 | 4 040 | 4 040 | 97 |
| 5x5_6_500 | 24 456 | 24 460 | 24 464 | 1 628 | 24 456 | 24 459 | 24 472 | 649 |
| 5x5_6_1000 | 51 376 | 51 404 | 51 416 | 4 745 | 51 382 | 51 412 | 51 416 | 1 593 |
| 5x5_20_10 | 250 | 250 | 250 | 12 | 250 | 250 | 250 | 4.6 |
| 5x5_20_100 | 1 134 | 1 134 | 1 134 | 68 | 1 134 | 1 136 | 1 142 | 21 |
| 5x5_20_500 | 20 996 | 20 999 | 21 006 | 14 456 | 20 996 | 21 027 | 21 058 | 4 632 |
| 5x5_50_1000 | 38 876 | 38 877 | 38 880 | 25 573 | 38 880 | 38 880 | 38 880 | 8 923 |

**4.4. Сравнение двух алгоритмов.** В табл. 3 представлено сравнение работы разработанных алгоритмов на сгенерированных примерах. Приведены результаты пяти запусков каждого алгоритма на примерах разной размерности. В столбцах таблицы находятся минимальное, максимальное и среднее значения целевой функции, а также время работы алгоритмов в секундах.

Для примеров малой размерности можно сделать предположение, что алгоритмы получили значения глобального оптимума, так как они совпадают на всех запусках. Также стоит отметить, что на некоторых примерах большей размерности гибридный генетический алгоритм нашёл решение лучше, чем итеративный алгоритм поиска с запретами, однако для этого алгоритма потребовалось значительно больше времени.

## Заключение

В представленной работе исследована задача оптимизации расположения товаров на складе, состоящем из стеллажей, путь обхода между которыми строго задан. Доказано, что задача NP-трудна даже в частном случае с двумя стеллажами и двумя типами товаров в каждом заказе. Для формализации задачи и исследования её с помощью программ-

оптимизаторов была построена модель смешанного целочисленного линейного программирования. Для решения задачи разработано два эвристических алгоритма: гибридный генетический алгоритм и итеративный алгоритм поиска с запретами. Разработан генератор примеров исследуемой задачи, на которых была протестирована и проанализирована работа предложенных алгоритмов.

**Финансирование работы**

Исследование выполнено в рамках гос. задания Института математики им. С. Л. Соболева СО РАН (проект № FWNF–2022–0019). Дополнительных грантов на проведение или руководство этим исследованием получено не было.

**Конфликт интересов**

Авторы заявляют, что у них нет конфликта интересов.

**Литература**

1. **Karasek J.** An overview of warehouse optimization // J. Adv. Telecommun. Electrotech. Signals Syst. 2013. V. 2, No. 3. P. 111–117.
2. **Caron F., Marchet G., Perego A.** Optimal layout in low-level picker-to-part systems // Int. J. Prod. Res. 2000. V. 38, No. 1. P. 101–117.
3. **Gu J., Goetschalckx M., McGinnis L. F.** Research on warehouse design and performance evaluation: A comprehensive review // Eur. J. Oper. Res. 2010. V. 203, No. 3. P. 539–549.
4. **De Koster R., Le-Duc T., Roodbergen K. J.** Design and control of warehouse order picking: A literature review // Eur. J. Oper. Res. 2007. V. 182, No. 2. P. 481–501.
5. **Ратушный А. А., Кочетов Ю. А.** Матэвристика для минимизации времени ожидания трейлеров при неточных временах прибытия // Дискрет. анализ и исслед. операций. 2022. Т. 29, № 3. С. 85–101.
6. **Ratliff H. D., Rosenthal A. S.** Order-picking in a rectangular warehouse: A solvable case of the traveling salesman problem // Oper. Res. 1983. V. 31, No. 3. P. 507–521.
7. **Roodbergen K. J., de Koster R.** Routing order pickers in a warehouse with a middle aisle // Eur. J. Oper. Res. 2001. V. 133, No. 1. P. 32–43.
8. **Brynzer H., Johansson M. I.** Storage location assignment: Using the product structure to reduce order picking times // Int. J. Prod. Econ. 1996. V. 46, No. 1. P. 595–603.
9. **Mantel R. J., Schuur P. C., Heragu S. S.** Order oriented slotting: A new assignment strategy for warehouses // Eur. J. Ind. Eng. 2007. V. 1, No. 3. P. 301–316.
10. **Koopmans T. C., Beckmann M.** Assignment problems and the location of economic activities // Econometrica. 1957. V. 25, No. 1. P. 53–76.

**11. Bui T. N., Chaudhuri S., Leighton F. T., Sipser M.** Graph bisection algorithms with good average case behavior // Combinatorica. 1987. V. 7, No. 2. P. 171–191.
**12. Гэри М., Джонсон Д.** Вычислительные машины и труднорешаемые задачи. М.: Мир, 1982. 416 с.
**13. Misevicius A., Kilda B.** Comparison of crossover operators for the quadratic assignment problem // Inf. Technol. Control. 2005. V. 34, No. 2. P. 109–119.
**14. Misevicius A.** An implementation of the iterated tabu search algorithm for the quadratic assignment problem // OR Spectrum. 2012. V. 34. P. 665–690.
**15. Glover F.** Future paths for integer programming and links to artificial intelligence // Comput. Oper. Res. 1986. V. 13, No. 5. P. 533–549.
**16. Glover F., Laguna M.** Tabu search // Modern heuristic techniques for combinatorial problems. New York: Halstead Press, 1993. P. 70–150.
**17. Lourenço H. R., Martin O. C., Stützle T.** Iterated local search // Handbook of metaheuristics. New York: Kluwer Acad. Publ., 2003. P. 321–353.

*Моторин Кирилл Олегович*
*Пяткин Артём Валерьевич*

Статья поступила
15 июля 2024 г.
После доработки —
5 августа 2024 г.
Принята к публикации
22 сентября 2024 г.

DISKRETNYI ANALIZ I ISSLEDOVANIE OPERATSII
/DISCRETE ANALYSIS AND OPERATIONS RESEARCH/
January–March 2025. Vol. 32, No. 1. P. 75–98

UDC 519.8 DOI: 10.33048/daio.2025.32.806

# ON ONE OPTIMIZATION PROBLEM FOR WAREHOUSE GOODS PLACEMENT

*K. O. Motorin*[1, a] and *A. V. Pyatkin*[2, b]

[1] Novosibirsk State University,
2 Pirogov Street, 630090 Novosibirsk, Russia
[2] Sobolev Institute of Mathematics,
4 Acad. Koptyug Avenue, 630090 Novosibirsk, Russia
E-mail: [a] k.motorin@g.nsu.ru, [b] artem@math.nsc.ru

**Abstract.** A warehouse goods placement problem is considered, where the aim is to minimize the total time of fulfillment of orders from a given list. NP-hardness of this problem even in the simplest special case is proved. An ILP model is suggested for this problem. Two heuristic algorithms are developed for solving this problem; their effectiveness is analyzed using randomly generated instances. Tab. 3, illustr. 9, bibliogr. 17.

**Keywords:** warehouse optimization, goods placement, NP-hard problem, genetic algorithm, tabu search.

## References

1. **J. Karasek,** An overview of warehouse optimization, *J. Adv. Telecommun. Electrotech. Signals Syst.* **2** (3), 111–117 (2013).
2. **F. Caron, G. Marchet,** and **A. Perego,** Optimal layout in low-level picker-to-part systems, *Int. J. Prod. Res.* **38** (1), 101–117 (2000).
3. **J. Gu, M. Goetschalckx,** and **L. F. McGinnis,** Research on warehouse design and performance evaluation: A comprehensive review, *Eur. J. Oper. Res.* **203** (3), 539–549 (2010).
4. **R. de Koster, T. Le-Duc,** and **K. J. Roodbergen,** Design and control of warehouse order picking: A literature review, *Eur. J. Oper. Res.* **182** (2), 481–501 (2007).
5. **A. A. Ratushnyi** and **Yu. A. Kochetov,** Matheuristics for waiting time minimization for trailers with uncertain arrival times, *Diskretn. Anal. Issled. Oper.* **29** (3), 85–101 (2022) [Russian] [*J. Appl. Ind. Math.* **16** (3), 540–549 (2022)].

English transl.: *Journal of Applied and Industrial Mathematics* **19** (1), 77–91 (2025), DOI: 10.1134/S1990478925010089.

6. **H. D. Ratliff** and **A. S. Rosenthal,** Order-picking in a rectangular warehouse: A solvable case of the traveling salesman problem, *Oper. Res.* **31** (3), 507–521 (1983).
7. **K. J. Roodbergen** and **R. de Koster,** Routing order pickers in a warehouse with a middle aisle, *Eur. J. Oper. Res.* **133** (1), 32–43 (2001).
8. **H. Brynzer** and **M. I. Johansson,** Storage location assignment: Using the product structure to reduce order picking times, *Int. J. Prod. Econ.* **46** (1), 595–603 (1996).
9. **R. J. Mantel, P. C. Schuur,** and **S. S. Heragu,** Order oriented slotting: A new assignment strategy for warehouses, *Eur. J. Ind. Eng.* **1** (3), 301–316 (2007).
10. **T. C. Koopmans** and **M. Beckmann,** Assignment problems and the location of economic activities, *Econometrica* **25** (1), 53–76 (1957).
11. **T. N. Bui, S. Chaudhuri, F. T. Leighton,** and **M. Sipser,** Graph bisection algorithms with good average case behavior, *Combinatorica* **7** (2), 171–191 (1987).
12. **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]).
13. **A. Misevicius** and **B. Kilda,** Comparison of crossover operators for the quadratic assignment problem, *Inf. Technol. Control.* **34** (2), 109–119 (2005).
14. **A. Misevicius,** An implementation of the iterated tabu search algorithm for the quadratic assignment problem, *OR Spectrum* **34**, 665–690 (2012).
15. **F. Glover,** Future paths for integer programming and links to artificial intelligence, *Comput. Oper. Res.* **13** (5), 533–549 (1986).
16. **F. Glover** and **M. Laguna,** Tabu search, in *Modern Heuristic Techniques for Combinatorial Problems* (Halstead Press, New York, 1993), pp. 70–150.
17. **H. R. Lourenço, O. C. Martin,** and **T. Stützle,** Iterated local search, in *Handbook of Metaheuristics* (Kluwer Acad. Publ., New York, 2003), pp. 321–353.


*Kirill O. Motorin*
*Artem V. Pyatkin*

Received July 15, 2024
Revised August 5, 2024
Accepted September 22, 2024