прикладные науки

MDPI

Статья

Оптимизация складских операций с помощью генетических алгоритмов

Мирослав Кордос [1,*], Ян Борычко [1], Марцин Блачник [2] и Славомир Голак [2]

[1] Факультет вычислительной техники и автоматики, Бельско-Бяльский университет, 43-340 Бельско-Бяла, Польша; jb054384@student.ath.edu.pl

[2] Факультет прикладной информатики, Силезский технологический университет, 44-100 Гливице, Польша; marcin.blachnik@polsl.pl (М.Б.);slawomir.golak@polsl.pl (С.Г.)

* Корреспонденциямkordos@ath.bielsko.pl

Получено: 7 июня 2020 г.; Принято: 9 июля 2020 г.; Опубликовано: 13 июля 2020 г.

проверка ротора обновления

Аннотация: Мы представляем полное, полностью автоматическое решение на основе генетических алгоритмов для оптимизации дискретного размещения товаров и маршрутов комплектации заказов на складе. Решение принимает на вход структуру склада и список заказов и возвращает оптимизированное размещение продукции, которое минимизирует сумму времен комплектации заказов. Маршруты комплектации заказов оптимизируются в основном генетическими алгоритмами с оператором многородительского кроссинговера, но в некоторых случаях могут использоваться также перестановки и методы локального поиска. Размещение продукции оптимизируется другим генетическим алгоритмом, где сумма длин оптимизированных маршрутов комплектации заказов используется в качестве стоимости данного размещения продукции. Мы представляем несколько идей, которые улучшают и ускоряют оптимизацию, такие как правильное количество родителей при кроссинговере, процедура кэширования, множественный перезапуск и группировка заказов. В представленных экспериментах, по сравнению со случайным размещением товаров и случайным порядком комплектации, оптимизация маршрутов комплектации заказов позволила сократить общее время комплектации заказов до 54%, оптимизация размещения товаров с помощью базовой версии метода позволила сократить это время до 26%, а оптимизация размещения товаров с помощью методов с улучшениями, такими как множественный рестарт и многородительский кроссовер - до 21%.

Ключевые слова: оптимизация склада; генетические алгоритмы; кроссовер

1. Введение

Значительная часть операционных расходов, связанных с хранением продукции, связана с комплектацией заказов. На основании многочисленных исследований было установлено, что около 60 % затрат на эксплуатацию склада составляют расходы комплектацию товаров при выполнении заказов [1]. Поскольку скорость выполнения этой операции является решающим фактором времени реагирования на заказы клиентов и одним из факторов, влияющих на их решение о выборе или отказе от той же компании при следующей покупке, представляется, что роль скорости выполнения заказа еще более возрастает.

Таким образом, сокращение времени комплектации заказов является важнейшим и наиболее выгодным фактором снижения затрат на эксплуатацию склада. Его можно достичь без существенных затрат, оптимизировав расположение конкретных товаров на складе и определив наиболее быстрые маршруты выполнения заказов при оптимизированном размещении товаров.

Некоторые аспекты оптимизации кажутся очевидными, например, что товары, которые часто заказываются вместе, должны располагаться близко друг к другу, а часто покупаемые товары должны быть расположены близко к пункту доставки. Однако для рассматриваемого в данной работе случая дискретных переменных, когда размеры размещения фиксированы (некоторые товары, например, хранение песка, гравия и так далее, могут быть выражены непрерывными переменными, но эта проблема здесь не рассматривается), при N товарах на складе,

количество всех возможных их размещений равно $N!$ Для $N = 100$ это дает $N! = 9{,}33 \times 10^{157}$, и даже если бы компьютер мог анализировать 1 миллиард перестановок в секунду, ему потребовалось бы $9{,}33 \times 10^{157}$ лет, чтобы найти оптимальное размещение товара. Таким образом, на практике разработать вручную оптимальное размещение товара невозможно, так как количество возможных вариантов размещения значительно превышает возможности анализа всех решений человеком или даже компьютерной программой, которая перебирает все возможные конфигурации.

В этой статье мы представляем полную, полностью автоматизированную систему, основанную на искусственном интеллекте, в частности на генетических алгоритмах, которая может преодолеть ограничения пространства поиска с помощью интеллектуального поиска. Обычно система анализирует от нескольких тысяч до нескольких десятков тысяч возможных вариантов размещения товара, чтобы найти оптимальное решение или, по крайней мере, решение, настолько близкое к оптимальному, что на практике это не будет иметь никакого значения. Более того, система возвращает и самые быстрые маршруты комплектации заказов. Преимуществом применения такого решения является ускорение операций и, следовательно, снижение операционных расходов склада (где обычно 60 % составляют расходы на комплектацию заказов [1]), а также возможность обслуживать больше клиентов теми же сотрудниками в то же время и, таким образом, еще больше увеличить продажи и прибыль.

Хотя этот вопрос анализируется уже , и особенно в последние годы он интенсивно развивается [2-4], нам не удалось найти в литературе полного, точного, полностью автоматического решения, которое бы учитывало реальное распределение заказов при оптимизации размещения товаров. Во всех работах, которые нам удалось найти, представлены лишь некоторые частичные подходы с большими упрощениями, например, длина маршрута определяется только одним товаром с наибольшим расстоянием от входа [2], или пользователь отвечает за перераспределение товаров, размещая более часто используемые ближе к входу [5].

Кроме того, часто список товаров в рамках одного заказа бывает достаточно длинным, особенно на складах, которые продают товары в основном розничным торговцам, что делает оптимизацию еще более важной. Цель данной статьи - восполнить этот пробел, представив наше решение и обсудив его особенности и их влияние на точность и скорость процесса оптимизации склада.

Сначала мы определяем проблему (раздел 2), затем перечисляем основные положения нашего вклада (раздел 3), после чего переходим к деталям в следующем порядке: обзор литературы (раздел 4), представление предлагаемого решения (разделы 5.1-5.4), экспериментальные результаты (раздел 6) и выводы (раздел 7).

2. Постановка проблемы

Размещение продукции определяет местоположение (обычно полки) конкретных продуктов на складе. Мы определяем стоимость размещения товара как сумму кратчайших маршрутов сбора заказов по всем заказам, включенным в список заказов для данного размещения товара (уравнение (1)). Предполагается, что если для сбора заказов используются роботы, то они будут следовать по кратчайшим маршрутам. Если же заказы собирают люди, то в большинстве случаев они будут следовать тем же маршрутам, однако иногда они могут решить изменить путь. Однако это можно рассматривать как случайный процесс и, следовательно, не учитывать при оптимизации.

$$Стоимость = \sum_{n=1}^{N_{ord}} Маршрут_{min}(n), \quad (1)$$

где N_{ord} - количество заказов в списке заказов, а $Route_{min}(n)$ - кратчайший маршрут комплектации заказа (выполнения заказа), найденный для *n-го* заказа.

Задача состоит в том, чтобы найти размещение товара с как можно меньшими затратами. Другими словами, нам нужно минимизировать стоимость, заданную уравнением (1), путем правильного размещения товаров в определенных местах на складе.

Для этого мы разработали алгоритм оптимизации размещения товаров, который мы опишем в следующих разделах.

Входными данными алгоритма являются:

- планировка склада в виде затрат на переход между соседними помещениями
- список заказов

Выходами алгоритма являются:

- оптимизированное размещение продукции на складе
- кратчайшие маршруты комплектования заказов для каждого заказа из списка заказов для оптимизированного размещения продукции

Здесь мы излагаем только основную идею, а подробности о каждом входе и выходе можно найти в последующих разделах. Список заказов содержит все рассматриваемые заказы, которые должны быть заказами, ожидаемыми в определенный будущий период времени. В большинстве случаев это могут быть прошлые заказы, так как можно ожидать, что будущие заказы будут иметь такое же распределение товаров. В противном случае это могут быть прошлые заказы из соответствующего сезона прошлого года или прогнозируемые будущие заказы. Каждый заказ состоит из нескольких (по крайней мере, одного) продуктов. Чтобы выполнить заказ, необходимо посетить места расположения всех продуктов, включенных в заказ, и выбрать продукты. Таким образом, подпроблема основной задачи оптимизации размещения продукции заключается в нахождении самого быстрого (кратчайшего) маршрута выполнения заказа для каждого заказа. Это необходимо, так как сумма кратчайших маршрутов комплектации заказов является основной целью, которую мы хотим минимизировать, оптимизируя размещение конкретных продуктов на складе.

Эта подпроблема поиска кратчайшего пути выполнения заказа эквивалентна задаче о путешествующем продавце (Traveling Salesman Problem, TSP). В то время как вся оптимизация размещения товара - это другая, гораздо более сложная задача. Основными отличиями являются:

1. При оптимизации размещения товаров задача заключается в поиске оптимальных мест размещения товаров. В TSP задача состоит в том, чтобы найти кратчайший маршрут, соединяющий все рассматриваемые места.
2. При оптимизации размещения продукции функция затрат, которую мы используем, представляет собой сумму кратчайших маршрутов комплектации заказов для данного размещения продукции (уравнение (1)). В TSP функция стоимости - это длина одного маршрута, соединяющего все рассматриваемые места.
3. Оптимизация размещения продукции - это двухуровневая вложенная задача, где основной (внешний) процесс - собственно оптимизация размещения продукции, а внутренний процесс вычисляет стоимость (фитнес) каждого рассматриваемого размещения продукции путем нахождения кратчайших маршрутов комплектации заказов и последующего их суммирования для получения стоимости. TSP - это одноуровневая задача без вложенной оптимизации.
4. При оптимизации размещения товаров методы, разработанные специально для TSP, такие как ближайший сосед или кроссовер HGReX в генетических алгоритмах, не могут быть использованы в основном процессе, потому что мы не ищем никакого маршрута, не существует никакой последовательности мест и нет такого понятия, как "перейти к следующему месту", которое используют эти методы.

3. Взнос

Основными положениями данной статьи являются:

1. Проектирование всей системы с вложенной генетической оптимизацией, где стоимость размещения товара выражается суммой времени выполнения заказа (раздел 5.2).
2. Улучшенная многородительская версия кроссовера HGreX для оптимизации маршрутов (раздел 5.4.1).
3. Процедура многократного быстрого перезапуска для повышения точности при минимальном росте времени оптимизации (раздел 5.4.3).
4. Автоматический выбор метода расчета маршрута комплектации заказов для обеспечения баланса между скоростью и точностью оптимизации (раздел 5.3).
5. Формат исходных данных в виде стоимости перехода между соседними точками, что обеспечивает максимальную точность при минимальных усилиях пользователя при подготовке данных Раздел 5.1.
6. Кэширование значений пригодности (раздел 5.4.2).

4. Похожие работы

4.1. Планирование и работа склада

Большое количество литературных позиций было посвящено складским операциям. Ван Гилс и другие [4] представили обзор и классификацию научной литературы, исследующей комбинации тактических и оперативных задач планирования комплектации заказов. Гроссе и другие [6] проанализировали человеческие факторы при комплектации заказов. Дийкстра и Рудберген [7] обсудили заранее определенные последовательности комплектования заказов, включая различные варианты планировки склада и его проходов. Они также рассмотрели подход динамического программирования, который определяет назначения мест хранения для этих схем, используя формулы длины маршрута и свойства оптимальности. Боланьос Зуньига [3] представил формальную математическую модель для одновременного определения назначения мест хранения и маршрутизации комплектовщиков с учетом ограничений приоритета, основанных на весе товаров и местоположении каждого товара на общем складе. Бартольди [1] в своей книге представил практические соображения по планированию и строительству склада. Ракеш [8] обсудил методы, определяющие оптимальную глубину полос, количество уровней хранения и другие параметры планировки склада для минимизации затрат на пространство и обработку материалов. В книге Даварзани [9] обсуждаются вопросы планирования склада, технологии, оборудования, управления человеческими ресурсами, связи с другими отделами и компаниями. Зунич [10] рассмотрел различные конструкции складов, особенно V-образные островки, и рассчитал маршруты комплектации заказов для этих конструкций. Дхармаприя [11] обсудил использование метода имитационного отжига для оптимизации планировки склада с учетом общего спроса и стоимости проезда, но без учета сосуществования различных товаров в одном заказе.

4.2. Генетические алгоритмы для оптимизации складов

Методы искусственного интеллекта, в частности генетические алгоритмы, являются решениями, имеющими множество успешных реализаций, которые в последние годы стремительно набирают популярность и применяются в том числе для оптимизации работы склада [12-14].

Генетические алгоритмы обладают двумя важными преимуществами: быстрым интеллектуальным поиском, используемым для определения размещения продукта, и глобальной функцией стоимости.

Первое их преимущество заключается в том, что благодаря интеллектуальному поиску генетическим алгоритмам не нужно анализировать все решения (все возможные перестановки расположения товаров), что невозможно из-за их количества. Обычно для поиска решения они анализируют от нескольких тысяч до нескольких сотен тысяч вариантов размещения товара (а не все $N!$ возможностей). Хотя генетические алгоритмы не гарантируют оптимального решения каждый раз, найденные решения достаточно близки к оптимальному решению, так что на практике это не имеет существенного значения.

Второе их преимущество заключается в том, что при использовании генетических алгоритмов нам не нужно самим определять правила, характеризующие хорошие решения. Это очень важно, поскольку обычно мы не знаем, как оптимально определить правила, и располагаем лишь некоторыми интуитивными знаниями (например, товары, часто покупаемые вместе, должны находиться рядом друг с другом). Однако выразить эти знания строгой математической форме невозможно из-за сложности системы и часто противоположных оптимумов различных комплектаций заказов. В генетических алгоритмах достаточно сформулировать функцию стоимости, которая здесь выражается как сумма длин маршрутов комплектации заказов или как общее время, необходимое для выполнения всех заказов за определенный период.

В генетических алгоритмах задача кодируется с помощью массивов, называемых хромосомами, по аналогии с кодированием в хромосомах биологических организмов [12,13]. Каждое размещение товара, закодированное хромосомой, представляет собой одно решение (одну особь). Первоначально генерируется пул случайных решений (в каждом решении товары случайным образом распределяются по местам на складе). Затем применяется интеллектуальный поиск с помощью трех основных операций - отбора, кроссинговера и мутации. Механизм отбора выбирает особей для оператора кроссинговера. Он устроен по аналогии с биологическим процессом, где лучшие особи имеют большую вероятность стать родителями и обменяться информацией для создания потомства.

Оператор кроссинговера порождает новую особь (ребенка) из уже существующих (родителей). Таким образом, он позволяет объединить информацию, закодированную в хромосомах разных особей, в одну новую особь. Механизм мутации обменивает значения между некоторыми позициями в хромосоме особи. Затем отобранные особи переходят в следующее поколение. Процесс повторяется итеративно до тех пор, пока не будет найдено удовлетворительное решение или пока не произойдет улучшение решений.

Подводя итог, следует сказать, что использование генетических алгоритмов или других эволюционных методов оптимизации в приложениях к складским системам, включая оптимизацию распределения товаров и маршрутов комплектации заказов, может принести ощутимую пользу компаниям, использующим эти решения, ускорив их работу и сократив операционные расходы.

Хотя идея применения генетических алгоритмов для оптимизации склада или маршрута комплектации заказов была представлена в некоторых литературных источниках, мы не нашли полного автоматического решения, учитывающего распределение заказов, как мы представляем в данной статье.

Поскольку оптимизация маршрута комплектации заказа (что эквивалентно задаче о путешествующем продавце - см. раздел 2) является гораздо более простой задачей, чем оптимизация размещения товара на складе, как и следовало ожидать, гораздо больше работ посвящено оптимизации маршрута комплектации заказа и лишь в нескольких работах рассматривается оптимизация размещения товара. Ниже мы кратко представим некоторые из них.

Ванг [5] применил генетические алгоритмы для оптимизации фитнес-функции, состоящей из трех взвешенных условий - оборачиваемости товаров, центра тяжести складских стеллажей и релевантности товаров. Вей [15] использовал аналогичный подход с генетическим алгоритмом с PMX-кроссовером, а фитнес-функция определялась условиями доступа к проходам, весом товаров и размерами полок.

Авдейкины и Саврасовы [2] применили генетические алгоритмы для оптимизации склада с использованием кроссинговера порядка (OX). Каждая особь в популяции представляла собой схему склада. Каждый ген представлял собой уникальный товар. В их решении пригодность каждой особи вычислялась как сумма максимальных расстояний до каждого заказа - $fitness(i) = \sum O_k d_{max}(I)$, где O - заказ из набора 1, 2, ..., k, а $d_{max}(I)$ - расстояние до самой дальней позиции комплектации. Расстояние выражалось целым числом, которое для первого товара было равно 1, для второго 2 и так далее, увеличиваясь на 1 от одного SKU к другому. Использование такой фитнес-функции перемещает наиболее часто продаваемые товары ближе к входу на склад, но при этом не размещает в соседних местах товары, которые часто продаются вместе, поскольку данное решение не учитывает этот аспект, а также не определяет реальные маршруты комплектации заказов, поэтому его нельзя считать полным решением.

Поскольку решение, представленное Авдейкиным и Саврасовым [2], на первый взгляд может показаться похожим наше, стоит указать на различия. В нашем решении удобство рассчитывается как сумма длин всех маршрутов комплектации заказов. Это принципиальное отличие их работы от нашего решения, так как это позволяет минимизировать время реальных операций по комплектации заказов и тем самым получить очень точные решения, которые также минимизируют расстояния между товарами, содержащимися в одном заказе. Следующее отличие заключается в том, что мы используем реальные затраты на переход (или реальные расстояния), а не аппроксимацию путем увеличения расстояния всегда на единицу, как это было у них. Еще одно отличие заключается в том, что наше решение определяет как размещение товаров, так и самые быстрые маршруты комплектации заказов. Кроме того, мы используем более новые, эффективные операторы кроссинговера и предлагаем множество улучшений точности и скорости оптимизации.

4.3. Операторы кроссовера

Правильный выбор оператора кроссинговера является решающим фактором в работе генетического алгоритма. В этом подразделе мы рассмотрим операторы кроссинговера и подробно остановимся на операторах AEX и HGreX, которые используются в нашем решении.

Кроссинговер позволяет объединить наиболее ценную информацию из двух или более различных хромосом (родителей) в одну хромосому (дочернюю), которая может представлять собой лучшее решение, чем ее родители. Для такого рода задач, где каждый товар может занимать только одно место одновременно и каждое место должно быть занято одним товаром, как оптимизация маршрутов комплектации заказов или размещения товаров

оптимизации на складе, необходимо использовать специальные операторы пересечения, чтобы гарантировать отсутствие дублирования элементов и присутствие каждого элемента во вновь созданной особи. Было разработано несколько таких операторов кроссинговера.

Хассанат и Алькафавин [16] предложили несколько операторов кроссинговера, таких как кроссинговер с разрезом по наихудшему гену (COWGC) и коллизионный кроссинговер, а также подходы к выбору, такие как выбор лучшего кроссинговера (SBC). COWGC обменивает гены между родителями, разрезая хромосому в точке, которая максимально снижает стоимость. Коллизионный кроссинговер использует две стратегии выбора операторов кроссинговера. Первая выбирает из нескольких рассмотренных кроссинговера тот, который максимально улучшает пригодность, а вторая выбирает любой оператор случайным образом. Алгоритм SBC применяет несколько операторов кроссинговера одновременно к одним и тем же родителям, и в итоге выбирает двух лучших потомков для включения в популяцию. Хванг представил операторы кроссинговера порядка (OX) и кроссинговера цикла (CX) [17]. Тан предложил эвристический жадный кроссинговер (HGreX) и его варианты HRndX и RProX [18]. Другие популярные операторы кроссинговера включают частично сопоставленный кроссинговер (PMX), кроссинговер с рекомбинацией краев (ERX) и кроссинговер с чередованием краев (ERX) [19].

В литературе [19] можно найти несколько сравнений эффективности этих операторов кроссинговера. Согласно этим сравнениям, наиболее эффективными методами для задачи о коммивояжере чаще всего оказывались варианты оператора кроссовера HGreX, а вторым по эффективности был оператор кроссовера AEX. По этой причине мы решили начать наш подход с применения этих двух операторов кроссинговера для нашей задачи оптимизации склада.

HGreX подходит только для тех видов задач, где можно определить стоимость перехода между двумя позициями. Например, его можно использовать для поиска кратчайшего пути комплектации заказов, поскольку мы можем определить расстояния (затраты) между конкретными местами, в которых находятся продукты из заказов и которые, таким образом, необходимо посетить. Однако его нельзя применить для оптимизации размещения продукции на складе, поскольку распределение продукции напрямую связано не с одним маршрутом комплектации заказов, а с целым набором различных маршрутов. Таким образом, в этом случае мы можем определить глобальную цель, которая заключается в минимизации суммы длин всех маршрутов комплектации заказов, но не можем выразить стоимость между любыми двумя позициями. AEX, с другой стороны, не использует стоимость перехода и поэтому может быть применен также к задачам, где такая стоимость не может быть определена, как, например, оптимизация размещения товаров на складе. Сначала мы представим оператор AEX, а затем оператор HGreX.

AEX создает ребенка из двух родителей, начиная со значения, которое находится на первой позиции в первом родителе. Затем он добавляет это значение из второго родителя, которое во втором родителе следует за значением, только что взятым из первого родителя. Затем снова значение из первого родителя, которое следует за значением, только что выбранным из второго родителя, и так далее. Если это невозможно, так как какой-то элемент будет повторяться, то выбирается случайный, не выбранный до сих пор элемент. В представленном примере каждая позиция в хромосоме представляет одно место на складе, а каждая буква - один товар.

Предположим, что у нас есть два родителя P1 и P2:

P1= [A B C D E F G H] P2= [
H A D B G F E C]

Чтобы создать ребенка, мы начинаем с любой позиции первого родителя P1. Начнем с A. Затем добавим это значение, которое находится во втором родителе после A, поэтому добавим D

Ch = [A D_____]
и значения, оставшиеся у родителей:
P1= [~~A~~ B C ~~D~~ E F G H] P2= [
H ~~A D~~ B G F E C]

Далее мы добавляем к ребенку значение, которое находится в P1 после D, то есть E

Ch= [A D E _____]
и значения, оставшиеся у родителей:

P1= [~~A~~ B C ~~D E~~ F G H] P2= [
H ~~A D~~ B G F ~~E~~ C]

Далее мы добавляем к ребенку значение, которое находится в P2 после E, то есть C

Ch= [A D E C _____]
и значения, оставшиеся у родителей:
P1= [~~A~~ B ~~C D E~~ F G H] P2= [
H ~~A D~~ B G F ~~E C~~]

Далее мы добавляем к ребенку значение, которое находится в P1 после C, то есть D. Однако D уже было использовано, поэтому такой выбор не является правильным. В этом случае мы выбираем случайным образом одно из оставшихся значений в P1. Выберем G.

Ch= [A D E C G_____]
P1= [~~A~~ B ~~C D E~~ F ~~G~~ H] P2= [
H ~~A D~~ B ~~G~~ F ~~E C~~]

Далее мы добавляем к ребенку значение, которое находится в P2 после G, то есть F

Ch= [A D E C G F _ _]
и значения, оставшиеся у родителей:
P1= [~~A~~ B ~~C D E F G~~ H] P2= [
H ~~A D~~ B ~~G F E C~~]

Затем мы добавляем к ребенку это значение, которое находится в P1 после F, то есть в данный момент H

Ch= [A D E C G F H _]
и значения, оставшиеся у родителей:
P1= [~~A~~ B ~~C D E F G H~~] P2= [
~~H A D~~ B ~~G F E C~~]

И наконец, мы добавляем к ребенку это значение, которое находится в P2 после H, то есть в данный момент B, и ребенок становится:

Ch= [A D E C G F H B]

Оператор кроссинговера HGreX работает аналогично AEX. Разница в том, что он не берет поочередно элементы от обоих родителей, а всегда выбирает тот элемент из двух родителей, до которого расстояние (стоимость) от текущего элемента короче (меньше).

Предположим, что у нас есть два одинаковых родителя P1 и P2:

P1= [A B C D E F G H] P2= [
H A D B G F E C]

Чтобы создать ребенка, мы начинаем с любой позиции первого родителя P1, пусть с A. Затем мы добавляем значение, которое имеет меньшую стоимость перехода (меньшее расстояние) к A из двух значений, которые появляются непосредственно после A в обоих родителях, то есть из B и D. Предположим, что стоимость перехода от A к B равна 12, а от A к D - 15. Поэтому мы выбираем B в качестве следующей позиции в ребенке.

Ch = [A B _____]
и значения, оставшиеся у родителей:
P1= [~~A B~~ C D E F G H] P2= [
H ~~A~~ D ~~B~~ G F E C]

Однако если бы затраты были от A до B: 18, а от A до D: 15, то мы бы выбрали D в качестве следующей позиции в ребенке. Конфликты разрешаются так же, как и в AEX.

В разделе 5.4.1 мы представили многородительские версии операторов кроссинговера.

5. Предлагаемый метод

В этом разделе мы описываем предложенный метод на основе генетического алгоритма, который оптимизирует размещение продукции и маршруты комплектации заказов на складе. Цель метода, как описано в разделе 2, - минимизировать стоимость размещения продукции, заданную уравнением (1), то есть найти такое распределение конкретных продуктов по позициям на складе, которое минимизирует сумму кратчайших маршрутов комплектации заказов по всем заказам из списка заказов. Таким образом, процесс оптимизации состоит из внешней процедуры (основного процесса), которая представляет собой оптимизацию размещения продукции (представлена в разделе 5.2), и внутренней процедуры (подпроцесса), которая представляет собой оптимизацию маршрутов комплектации заказов для каждого рассматриваемого размещения продукции (представлена в разделе 5.3).

5.1. Формат данных и кодирование проблем

В этом подразделе объясняется формат входных данных и кодирование задачи в хромосомах генетического алгоритма. В конце также объясняется расчет матрицы стоимости перехода.

Чтобы определить качество размещения товара (см. Алгоритм 1), сначала нужно маршруты комплектации заказов для каждого заказа (см. Алгоритм 2). Чтобы найти их, необходимо рассчитать матрицу стоимости перехода (стоимость перемещения между каждой парой мест на складе), а для расчета полной матрицы пользователь должен предоставить стоимость перехода (или расстояние) между соседними местами.

На рисунке 1 показан примерный план очень небольшого склада, который мы используем для объяснения формата данных и кодирования задач. Разумеется, реальные склады, для которых создавалась методология, будут гораздо больше. В этом примере расстояния, введенные пользователем, показаны цветными линиями на рисунке 1, а матрица расстояний с этими записями приведена в таблице 1. Поскольку расстояния симметричны (например, *dist*(*loc5*, loc8)= *dist*(*loc8*, loc5)), достаточно заполнить расстояния по диагонали. Остальные расстояния (например, *dist*(*loc7*, loc10)) будут вычислены автоматически.

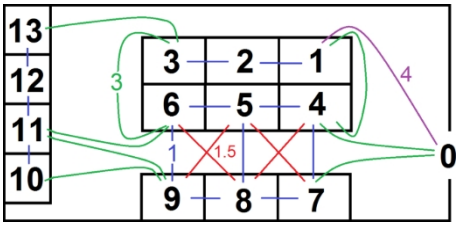

Рисунок 1. Примерная структура хранилища, используемая для объяснения формата данных и кодирования задач. Расстояния между соседними местами: синим цветом обозначены расстояния в 1 единицу, красным - в 1,5 единицы, зеленым - в 3 единицы, фиолетовым - в 4 единицы.

В хромосомах закодированы планировки складов с размещением товаров и маршруты комплектации заказов. Предположим, что количество доступных товаров равно количеству мест на складе, то есть 13 для данного примера склада, а названия товаров следующие: A,B,C,D,E,F,G,H,I,J,K,L,M. Предположим также, что существует три различных заказа, которые поступают с одинаковой частотой и состоят из следующих продуктов:

Заказ1: A,B,C,D,E,F,G
Заказ2: G,H,I,J,K
Заказ3: A,B,K

Для оптимизации размещения товаров мы закодируем задачу в хромосоме с 13 позициями. Вначале мы генерируем популяцию случайных особей хромосом, представляющих размещение товаров (см. Алгоритм 1). Предположим, что 15-я случайно сгенерированная особь выглядит следующим образом:

Layout15= [G H E F I J A C D K B L M]

В данном случае продукт G находится в месте 1 на рисунке 1, продукт H - в месте 2, и так далее. Нам нужно найти кратчайшие маршруты комплектации заказов для каждой особи (каждого размещения товара), чтобы вычислить ее пригодность. Для этого мы генерируем популяцию случайных хромосом, представляющих маршруты. На первой позиции всегда стоит 0 (обозначающий вход), а остальные позиции заняты случайным образом заказанными товарами из данного заказа. Допустим, 12-я случайно сгенерированная особь для заказа3 выглядит следующим образом:

Order3-Route12= [0 A K B 0]

Длина этого *маршрутаR* определяется по формуле:

$$lengthR = dist(loc0, \text{loc7}) + dist(loc7, \text{loc10}) + dist(loc10, \text{loc11}) + dist(loc11, \text{loc0})$$

как в Layout15 *loc0* - это вход, продукт A на месте 7, продукт K на месте 10 и продукт B на месте 11.

Таблица 1. Исходная матрица расстояний, соответствующая структуре склада, показанной на рисунке 1, содержащая только те значения, которые необходимы алгоритму.

.	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0		4			3			3						
1			1		3									
2				1										
3							3							3
4						1		1	1.5					
5							1	1.5	1	1.5				
6									1.5	1	3			
7									1					
8										1				
9											3	3		
10												1		
11													1	
12														1
13														

Этот формат входных данных был специально разработан для того, чтобы потребовать минимальных усилий от пользователя, вводящего данные, и в то же время обеспечить максимальную точность расчетов. Во входных данных требуется указать только стоимость переходов между соседними местами. Однако если пользователь хочет ввести также стоимость перехода между соседними локациями, он может это сделать. Программа сохраняет все введенные пользователем расстояния и рассчитывает только оставшиеся расстояния.

Стоимость перехода между локациями можно вводить как расстояние в метрах, но также и в секундах - как время, необходимое для преодоления этого расстояния. Это учитывает, что, например, стоимость преодоления одного и того же расстояния по вертикали выше, чем по горизонтали, или что поворот за угол требует больше времени, чем преодоление того же расстояния по прямой линии, и таким образом позволяет получить более высокую точность времени комплектации заказов. Однако эти единицы не имеют никакого значения для предлагаемого метода, который просто рассматривает их как единицы стоимости.

Поскольку имеющиеся планы различных складов могут быть в разных более или менее удобных форматах, создание отдельной программы для подготовки исходных данных для каждого отдельного склада нецелесообразно, так как это займет больше времени, чем ввод переходных затрат вручную. Программе не нужно знать геометрическую структуру склада. Это дополнительное преимущество, так как в этом случае требуется гораздо меньше работы по подготовке исходных данных.

Алгоритм 1 Процесс оптимизации Product Placement (PP)

Исходные данные 1: схема склада в виде стоимости перехода между соседними местами (см. таблицу 1)

Вход 2: Список заказов

Выход 1: Оптимизированное размещение товара (PP) на складе

Выход 2: Кратчайшие маршруты комплектации заказов для каждого заказа для оптимизированного ПП

1: С помощью алгоритма Дейкстры вычислите полную матрицу **D** стоимости перехода между местами расположения продуктов
2: **для** *k*= от 1 **до** *numberO f ProcessRestarts* **do**
3: Сгенерируйте случайную популяцию *W* PPs
4: **if** *numberO f ProcessRestarts*> 1 **then**
5: *ma xIterationPP*= 10 else *maxIterationP P*= 1000
6: **конец if**
7: **для** *n*= от 1 **до** *maxIterationPP* **сделайте**
8: (Вычисление пригодности w-го PP)
9: **for** *i*= 1 **to** *popSizePP* **do**
10: **если** i-й ПП уже находится в кэше**, то**
11: Извлеките *costPP*(*i*) из кэша
12: Вычислить пригодность *f itnessPP*(*i*) i-го ПП в соответствии с уравнением (3)
13: **else**
14: **для** *j*= от 1 **до** *numOrdersWithDi f f erentItems* **do**
15: Запустите алгоритм 2, чтобы найти кратчайший маршрут для j-го заказа и его *длинуR*(*i*, *j*) 16: Умножить *длинуR*(*i*, *j*) на количество заказов, содержащих одинаковые предметы $N_{rep}(j)$ 17: **конец для**
18: Рассчитайте стоимость i-го ПП как сумму *длинR* лучших маршрутов всех заказов
19: Вычислите пригодность *f itnessPP*(*i*) i-го ПП в соответствии с уравнением (3)
20: Обновление кэша
21: **конец if**
22: **конец для**
23:
24: **для** *p*= **1 to** *popSizePP* **do**
25: Выберите родителей для каждого ребенка PP
26: Создайте дочерний PP с помощью оператора кроссинговера AEX
27: **конец для**
28: **если** за NBI_{PP} итераций *f itnessPP* лучшего PP не улучшился**, то**
29: break
30: **конец if**
31: Сортировка родительского и детского населения ПП
32: продвигать *popSizePP* PPs от лучших родителей и лучших детей в следующее поколение.
33: Применить мутацию и обновить кэш для каждого мутированного PP, которого еще нет в кэше
34: **конец для**
35: **if** *numberO f ProcessRestarts*> 1 **then**
36: Сохранение текущей популяции ПП и их *жизнеспособностиПП*
37: **конец if**
38: **end for**
39: **if** *numberO f ProcessRestarts*> 1 **then**
40: установить *числоO f ProcessRestarts*= 1
41: Восстановите популяцию лучших PP и продолжите оптимизацию со строки 4
42: **end if**
43: Возврат лучшего ПП и соответствующего набора кратчайших маршрутов комплектации заказов

Алгоритм 2 Процесс оптимизации маршрута комплектования заказов

Вход 1: Матрица **D** затрат на переход от одного местоположения продукта к другому

Вход 2: Список товаров в j-ом заказе

Вход 3: размещение i-го продукта

Выход 1: Кратчайший маршрут для завершения j-го заказа $route_{min}(i, j)$

Выход 2: Длина кратчайшего маршрута для j-го порядка завершения $lengthR_{min}(i, j)$

если количество товаров в j-м заказе<= Threshold1**, то**
    Определите $маршрут_{min}(i, j)$, оценивая перестановки
else if number products in j-th order<= Threshold2 **then**
    если $маршрут_{min}(i, j)$ находится в кэше**, то**
        Получение $маршрута_{min}(i, j)$ и $длиныR_{min}(i, j)$ из кэша
    else
        Сгенерируйте случайную популяцию маршрутов *popSizeR*
        для *k*= от 1 **до** *maxIterationR***.**
            для *m*= 1 **to** *popSizeR* **do**
                Рассчитайте длину каждого маршрута *lengthR*(*i*, *m*)
            конец для
            если длина кратчайшего маршрута не уменьшилась за NBI_{route} итераций**, то**
                Верните $маршрут_{min}(i, j)$ и $длинуR_{min}(i, j)$
            конец if
            для *m*= 1 **to** *popSizeR* **do**
                Рассчитайте пригодность маршрутов *f itnessR*(*i*) для оператора выбора в соответствии с уравнением (5)
                Выберите родителей для каждого маршрута ребенка
                Создайте дочерний маршрут с помощью оператора кроссовера
            конец для
            Примените мутацию к маршрутам
            Сортировка родительских и дочерних маршрутов
            если за $NBI_{маршрут}$ итераций наилучшее соответствие маршрута не улучшилось**, то**
                перерыв
            конец if
            продвижение маршрутов *popSizeR* от лучших родителей и детей к следующему поколению
        конец для
        Обновление кэша
    конец if
else
    Определите $маршрут_{min}(i, j)$ с помощью алгоритма ближайших соседей
end if
Возврат $маршрута_{min}(i, j)$ и $длиныR_{min}(i, j)$

Для вычисления оставшихся затрат на переход между каждой парой местоположений (строка 1 в Алгоритме 1) можно использовать любой алгоритм, который может это сделать, например алгоритм Дейкстры [20], алгоритм Флойда Уоршалла [21] или алгоритм Беллмана-Форда [22]. Мы используем алгоритм Дейкстры, потому что он самый быстрый, особенно для разреженных графов (как в нашем случае), где каждая вершина связана только с несколькими другими вершинами. Для графа из v вершин (местоположений) и e соединяющих ребер (стоимости перехода) вычисление всех расстояний с помощью алгоритма Дейкстры с приоритетной очередью имеет сложность $O(v(e+ vlogv))$, то время как сложность двух других алгоритмов составляет $O(^{v3})$ и $O(^{ev2})$.

Вычисление матрицы затрат с помощью алгоритма Дейкстры занимает лишь очень малую, практически пренебрежимо малую долю времени от всего процесса оптимизации размещения товаров. Более того, однажды рассчитанная матрица затрат может быть повторно использована для других вариантов размещения товаров, если физическая планировка склада не меняется. Алгоритм A* [23] может быть быстрее алгоритма Дейкстры только в том случае, если

когда можно оценить приблизительную стоимость перехода от текущего узла к целевому. Однако в данной задаче мы не можем оценить приблизительную стоимость, так как не знаем координат конкретных мест, а только стоимость перехода между соседними местами. Учитывая эти два фактора, требовать от пользователя подготовки дополнительных данных с координатами каждого местоположения для использования алгоритма A* неоправданно, так как в этом случае выигрыш процессорного времени (обычно менее секунды) не компенсирует потерю времени пользователя (обычно несколько часов), потраченного на подготовку дополнительных данных.

5.2. Оптимизация размещения продукции

В этом подразделе представлен алгоритм, используемый для оптимизации размещения конкретных товаров на складе с целью минимизации общего времени выполнения заказов из списка заказов. Основной процесс оптимизации показан на псевдокоде в Алгоритме 1 и в виде диаграмм на рисунках 2 и 3. Подпроцесс, определяющий кратчайшие маршруты комплектации заказов, рассматривается в следующем разделе.

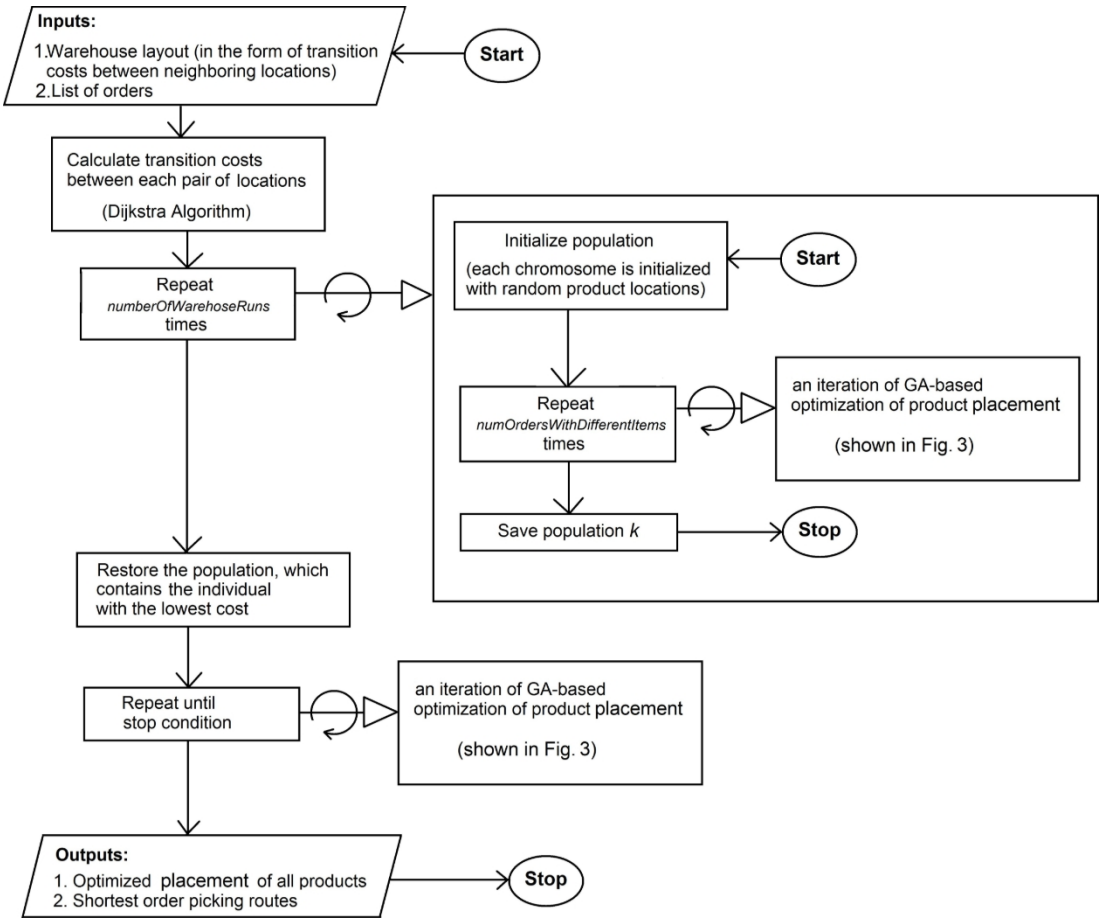

Рисунок 2. Оптимизация размещения продукта (основной процесс).

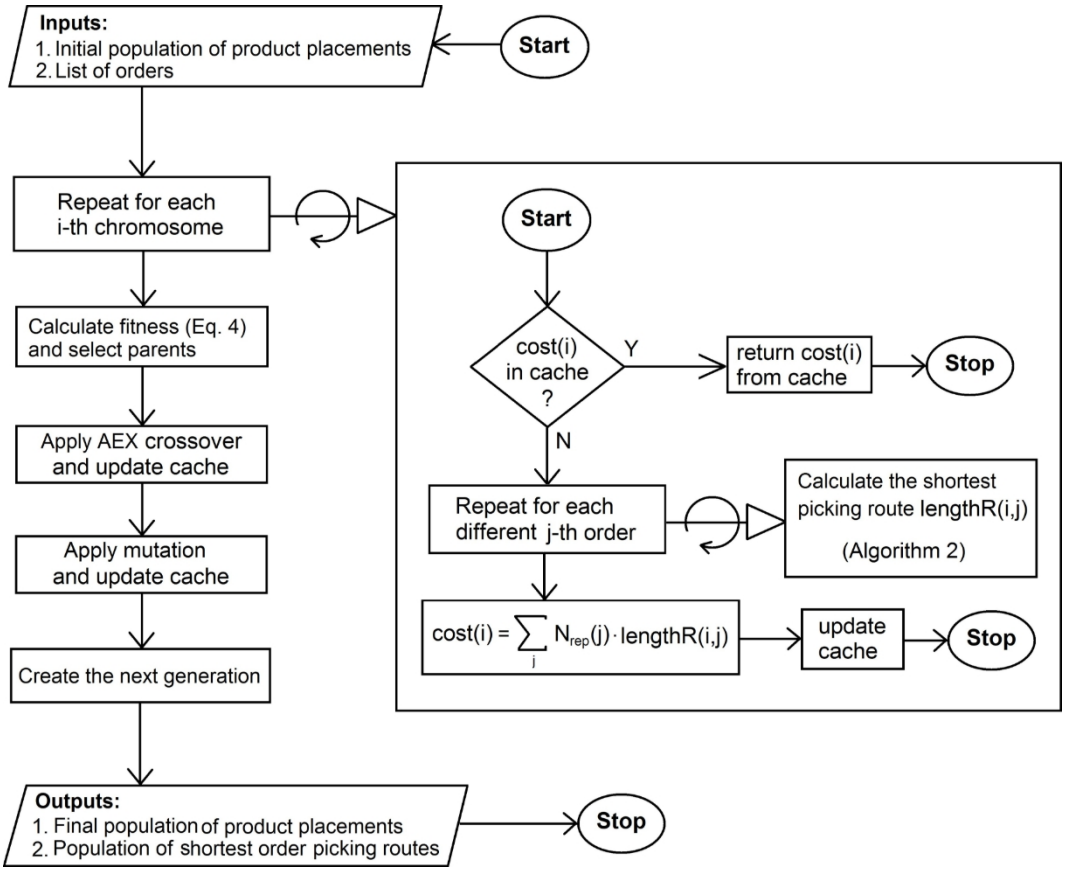

Рисунок 3. Оптимизация размещения товара (внутренний блок алгоритма, показанного на рисунке 2).

Сейчас мы объясним базовую версию алгоритма с *числомO f ProcessRestarts*= 1 (строка 2 в алгоритме 1), а в разделе 5.4.3 мы объясним использование и назначение множественных перезапусков процессов (*числоO f ProcessRestarts*> 1).

Процесс начинается в строке 1, где алгоритм Дейкстры вычисляет полную матрицу **D** стоимости перехода (или расстояния) между местами расположения продуктов (подробнее см. раздел 5.1). В строке 3 генерируется начальная случайная популяция мест размещения товаров (PP). В строке 7 генетический алгоритм начинает оптимизацию. В строке 10 алгоритм проверяет, существовала ли какая-либо из текущих особей ранее в текущей или любой прошлой эпохе. Если да, то фитнес такой особи не вычисляется, а извлекается из кэша. В строке 15 с помощью алгоритма 2 вычисляется кратчайший маршрут для выполнения каждого из различных заказов для текущего ПП. Поскольку нам нужно минимизировать сумму времен выполнения заказов, для каждого рассматриваемого размещения товаров на складе необходимо вычислить время выполнения каждого заказа, а затем сложить эти времена. Для минимизации вычислительной сложности этого шага мы группируем заказы, состоящие из одинаковых продуктов, и присваиваем такому агрегированному заказу больший вес, равный количеству единичных заказов, из которых состоит агрегированный заказ. В строке 18 вычисляется стоимость текущего ПП, затем из стоимости вычисляется пригодность и обновляется кэш. В строке 25 выбираются родители для каждого ребенка, а в следующей строке создается ребенок с помощью оператора кроссинговера AEX. Если фитнес лучшего PP не улучшился за NBI_{PP} итераций (строка 28), то оптимизация завершается. В строке 32 происходит переход детей и лучших родителей в следующее поколение. В следующей строке применяется мутация и обновляется кэш. Наконец, в последней строке алгоритм возвращает наилучшее размещение товара и соответствующий набор кратчайших маршрутов комплектации заказов.

Для того чтобы давление отбора было постоянным и, следовательно, исследование пространства поиска было более интенсивным на ранних стадиях, а сходимость - более быстрой на поздних [24], как и в оптимизации размещения товаров, так и в подпроцессе оптимизации маршрута комплектации заказов, мы используем фитнес

нормализация функции. Таким образом, стоимость данного размещения товара и длина маршрутов комплектации заказов не используются напрямую в качестве фитнес-значений, а пересчитываются. Мы использовали выбор колеса рулетки как для оптимизации размещения товаров, так и для оптимизации маршрутов комплектации заказов. Согласно Разали [25] и нашим экспериментам, выбор с помощью колеса рулетки и турнирный выбор дают сопоставимые результаты. Кроме того, в обоих случаях можно регулировать давление отбора (количество кандидатов на родителя в турнирном отборе и форму фитнес-функции в отборе с помощью колеса рулетки). Мы выбрали отбор с помощью колеса рулетки для отладки, так как с его помощью было проще проанализировать детальное поведение алгоритма и произвести его тонкую настройку.

Стоимость размещения *i-го* продукта *costPP*(*i*) выражает сумму кратчайших маршрутов, найденных для каждого комплектования заказа, и задается уравнением (2):

$$CostPP(i = \sum_{j\,1=}^{Нои\,f\,f} N_{rep}(j) \text{ - } длинаR_{min}(j) \quad (2)$$

где $N_{di\,f\,f}$ - количество различных заказов, $N_{rep}(j)$ - число, выражающее, сколько раз *j-й* заказ повторяется в списке заказов, а $lengthR_{min}(j)$ - наилучший (кратчайший) маршрут, найденный для выполнения *j-го* заказа. Для оценки хода и результатов оптимизации используется значение *costPP* (см. рис. 3).

Уравнение (3) определяет пригодность *f itnessPP*(*i*) *i-го* размещения продукта, используемого при выборе колеса рулетки:

$$f\,itnessPP(i) = c_1 + \frac{costPP_{max} - costPP(i)}{стоимостьPP_{max} - стоимостьPP_{min}}, \quad (3)$$

где c_1 - коэффициент (чем меньше c_1, тем большее предпочтение отдается особям с меньшей стоимостью), $costPP_{max}$ - максимальная стоимость, $costPP_{min}$ - минимальная стоимость, а *costPP*(*i*) - стоимость размещения i-го продукта в популяции.

Переменная *f itnessPP* принимает большие значения для лучших особей, чтобы гарантировать, что лучшие особи имеют большую вероятность быть выбранными в качестве родителей, а переменная *costPP* принимает меньшие значения для лучших особей, поскольку выражает стоимость размещения продукта, которая равна сумме длин кратчайших маршрутов.

Мы используем динамическую вероятность мутации (вероятность замены некоторых мест в хромосоме), которая постепенно увеличивается в процессе оптимизации. Кроме тоговероятность мутации выше для особей с более низкой приспособленностью. Это минимизирует вероятность разрушения особи с высокой приспособленностью и усиливает исследовательскую роль особей с низкой приспособленностью. Более низкая вероятность мутации также позволяет эффективнее кэшировать значения пригодности (см. алгоритм 1). Эффективность этого подхода была основана на различных наблюдениях [26]. Мы используем два различных оператора мутации - обратную мутацию последовательности (RSM) и частичную мутацию перестановкой (PSM), причем вероятность применения RSM в три раза выше. Выбор этих операторов мутации основан на экспериментальном исследовании Отмана и других [27]. Общая вероятность *мутацииProb*(*i*) применения мутации к *i-й* хромосоме выражается уравнением (4).

$$mutationProb(i) = (c \sqrt{}_{i} iter + \overline{c} \quad _{n} - iter_{NBI}) \frac{Fitness_{max}}{Фитнес\,(i) + c_f}, \quad (4)$$

где c_i, c_n и c_f - коэффициенты, *iter* - текущая итерация (эпоха) генетического алгоритма, $iter_{NBI}$ - количество итераций без улучшения лучшей особи. Универсальные значения коэффициентов по умолчанию для наших целей были экспериментально установлены на c_i= 0.00001, c_n= 0.00001 и c_f= 0.3. Дальнейшее совершенствование схемы мутации, а также взаимодействия мутации и кроссинговера - довольно сложный вопрос, и это будет одной из тем наших будущих исследований, когда мы попытаемся найти оптимальные схемы для разных ситуаций.

5.3. Оптимизация маршрутов комплектации заказов

Как уже говорилось ранее, задача оптимизации маршрутов комплектации заказов является частью оптимизации размещения продукции на складе. Сумма длин маршрутов комплектации заказов для данного размещения товара является его *себестоимостьюPP* - чем меньше, тем лучше (см. уравнение (1)). Процесс представлен в псевдокоде в Алгоритме 2 и в диаграмме на рисунке 4.

Во время оптимизации маршрута комплектации заказов местоположение продуктов остается постоянным. Алгоритм 1 изменяет местоположение продуктов только перед каждым раундом оптимизации маршрута комплектации заказов. Как уже говорилось в разделе 5.1, маршрут комплектации заказов начинается от входа, затем посещает все места расположения продуктов, перечисленных в текущем заказе, и возвращается к входу на склад. Задача состоит в том, чтобы оптимизировать последовательность посещения мест для получения кратчайшего маршрута.

Существует два основных семейства подходов к поиску кратчайших маршрутов, соединяющих список местоположений: методы локального поиска (например, ближайшего соседа или k-opt [28]) и популяционные методы (например, генетические алгоритмы или оптимизация муравьиной колонии [28]). В алгоритме ближайшего соседа мы всегда переходим от текущего местоположения к ближайшему еще не посещенному местоположению. Таким образом, алгоритм реализует локальный поиск. Локальный поиск гарантирует нахождение ближайшего местоположения к текущему с вероятностью 100 %. Однако недостатком подходов локального поиска является то, что они не учитывают глобальный взгляд на ситуацию. Несмотря на то что были проведены исследования по улучшению этих методов, популяционные методы все еще имеют преимущество в применении глобального поиска. Пример маршрута, определенного с помощью ближайшего соседа, который демонстрирует проблемы этого метода, показан на рисунке 5.

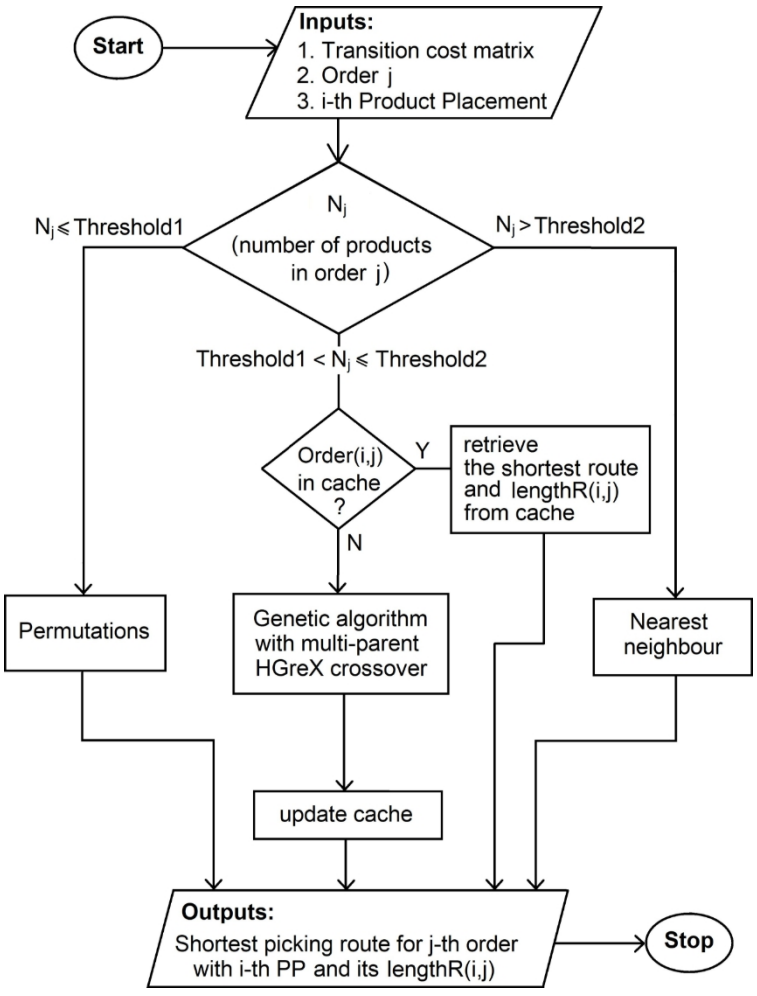

Рисунок 4. Оптимизация маршрута комплектации заказов.

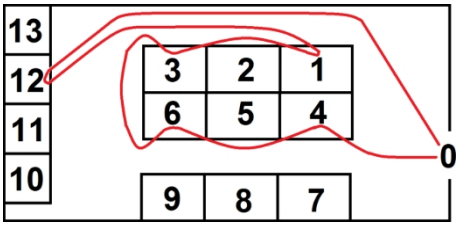

Рисунок 5. Пример маршрута (красным цветом), соединяющего точки 0, 4, 6, 2, 1, 12, определенный с помощью алгоритма ближайших соседей.

Как будет показано в экспериментальных оценках в разделе 6 и как известно из предыдущих исследований [29], при использовании алгоритма ближайших соседей вместо генетических алгоритмов для такого рода задач время вычислений может быть значительно сокращено, однако ценой худших результатов (в среднем на 10 % длиннее маршруты).

Чтобы определить кратчайший маршрут для выполнения каждого заказа, наш метод может использовать три различных алгоритма:

- Итерация по половине возможных перестановок (поскольку матрица стоимости перехода симметрична, мы можем начать выполнение заказа с конца или с начала определенного маршрута, поэтому нам не нужно проверять все перестановки, а только половину из них). Этот метод на 100% точен, но является самым медленным для очень коротких заказов. Поскольку его сложность составляет $O(n!)$, он непрактичен для заказов свыше 10 позиций и технически невозможен для заказов свыше 15 позиций.
- Генетический алгоритм с многородительским оператором кроссинговера HGreX. Этот метод быстрее, чем перебор перестановок, но не гарантирует нахождения наилучшего решения, а только решения, близкого к наилучшему. (Базовая версия кроссинговера HGreX представлена в разделе 4.3, а многородительское расширение - в разделе 5.4.1). Пригодность i-го маршрута комплектации заказов уравнением (5). $f\ itnessR(i)$ принимает большие значения для лучших особей, чтобы гарантировать, что они имеют большую вероятность быть выбранными в качестве родителей, в то время как *lengthR* принимает меньшие значения для лучших особей, так как эта величина выражает длину маршрута.

$$FitnessR(i) = c_2 + \frac{lengthR_{max} - lengthR(i)}{длинаR_{max} - длинаR_{min}} \quad (5)$$

где c_2 - коэффициент, определяющий силу отбора (чем меньше c_2, тем большее предпочтение отдается особям, представляющим более короткие маршруты) $lengthR_{max}$ - максимальная, а $lengthR_{min}$ - минимальная длина маршрута комплектования заказов в популяции. Это гарантирует, что перемасштабированная пропорция между максимальной и минимальной пригодностью будет постоянной во время оптимизации (пояснения см. разделе 5.2).

- Алгоритм ближайших соседей. Это самый быстрый метод. Он также не гарантирует нахождения наилучшего решения, и в применении к нашей задаче обычно находит худшие решения, чем генетические алгоритмы.

Чтобы обеспечить оптимальный компромисс между точностью и скоростью оптимизации маршрута, можно применить три вышеуказанных алгоритма, а значения *Threshold1* и *Threshold2* используются для определения того, какой именно алгоритм будет использоваться для данного заказа, в зависимости от количества товаров в заказе (см. Алгоритм 2 и Рисунок 4).

Число возможных маршрутов комплектации заказов равно числу перестановок k-элементного множества, которое *равно k*! Если в данном заказе меньше k= 7 товаров, то быстрее оценить половину возможных перестановок (для k= 6 нужно рассмотреть 6!/2= 360 различных маршрутов), чем использовать генетические алгоритмы. Для k= 7 нам нужно оценить 7!/2= 2520 перестановок. С другой стороны, генетические алгоритмы обычно способны найти решение, оценивая меньшее количество маршрутов (например, при популяции в 50 особей и 10 итерациях, что дает всего 500 оценок). Однако генетические алгоритмы

имеют дополнительные временные затраты на такие операции, как отбор, кроссинговер, генерация случайных чисел и так далее. Таким образом, при 7 товарах в заказе время расчета сопоставимо, а при более чем семи товарах генетические алгоритмы работают быстрее. Таким образом, мы предлагаем установить *порог1*= 7.

Как видно из раздела 6, результаты, полученные с помощью генетических алгоритмов с многородительским кроссинговером HGreX, превосходят результаты, полученные с помощью алгоритмов ближайших соседей. С другой стороны, алгоритм ближайшего соседа быстрее генетических алгоритмов. Однако его преимущество в скорости гораздо выше при оптимизации одного маршрута, где он может быть на два порядка быстрее. В нашей системе, когда оптимизация маршрута является итеративно выполняемым подпроцессом оптимизации размещения товара, разница в скорости между этими двумя методами гораздо ниже, менее одного величины. Это объясняется двумя причинами. Первая заключается в том, что реализовано кэширование уже рассчитанных маршрутов (см. подробности в разделе 5.4.2). Накладные расходы на кэширование сопоставимы с вычислительными усилиями алгоритма ближайших соседей, поэтому оно может лишь ускорить расчет маршрута на основе генетического алгоритма. Вторая причина заключается в том, что время выполнения основного процесса (оптимизации размещения товара) в обоих случаях одинаково.

Порог2 указывает, при превышении какого количества товаров в заказе следует использовать алгоритм ближайших соседей для оптимизации данного маршрута комплектации заказа. Для получения наилучших результатов рекомендуется установить значение Threshold2 настолько высоким, чтобы алгоритм ближайших соседей не использовался вообще (например, *Threshold2*= 1000). Однако если наши данные очень велики, а вычислительные и временные ресурсы ограничены, мы можем установить Threshold1= 0 и Threshold2= 0, и таким образом для оптимизации маршрута будут использоваться только алгоритмы ближайших соседей. Иногда случается так, что длинных заказов очень мало (например, два заказа по 50 товаров и все остальные заказы менее 20 товаров), поэтому эти несколько заказов не окажут существенного влияния на конечное размещение товаров, и мы можем использовать алгоритм ближайших соседей для них, чтобы ускорить вычисления, и генетические алгоритмы для всех остальных заказов (в этом случае установив, например, *Threshold2*= 30).

5.4. Улучшения и ускорения процесса

Для получения лучших результатов и ускорения процесса мы используем следующие усовершенствования: многородительский оператор кроссинговера, группировка заказов, кэширование стоимости размещения товаров и маршрутов комплектования заказов оцениваемых особей, многократный перезапуск, переключение между перестановками/генетическими алгоритмами/ближайшими соседями для оптимизации маршрутов, а также распараллеливание процесса. В следующих подразделах мы представим конкретные улучшения. Влияние этих улучшений на полученные результаты оценивается экспериментально и представлено в таблицах и рисунках в разделе 6.

5.4.1. Операторы кроссовера с несколькими родителями

Поскольку использование многородительских операторов кроссинговера позволяет значительно (до трех раз) ускорить скорость сходимости классических генетических алгоритмов [30,31] (а также их одноцелевых и многоцелевых версий, построенных на основе алгоритма NSGA-II [32]), одной из идей данной работы применение многородительского подхода к операторам кроссинговера в задачах оптимизации маршрутов и размещения товаров в надежде, что он позволит получить лучшие результаты.

Существует и другое обоснование увеличения числа родителей в операторе кроссинговера HGreX. В алгоритме ближайшего соседа позиции добавляются к маршруту по одной; каждый раз ближайшая добавляется к последней. В крайнем случае, когда популяция велика настолько, что в ней существует почти каждая возможная двухэлементная последовательность, а число родителей в многородительском кроссинговере HGreX равно размеру популяции - построенный таким образом генетический алгоритм становится эквивалентным поиску ближайшего соседа. Но, с другой стороны, небольшое увеличение числа родителей может добавить в генетические алгоритмы компонент локального поиска и тем самым улучшить результаты.

Предположим, что для создания каждого ребенка мы будем использовать четырех родителей.

P1= [A B C D E F G H] P2=
[E G F H A C B D] P3= [G
H A E B F C D] P4= [E F H
D B A G C]

Начнем с первой позиции в P1, это позиция A. Предположим, что существуют следующие расстояния d(A,B)= 12, d(A,C)= 15, d(A,E)= 18, d(A,G)= 11. Так как в данном случае расстояние d(A,G) наименьшее, то следующей позицией ребенка будет G.

Ch = [A G _____]
и значения, оставшиеся у родителей:
P1= [~~A~~ B C D E F ~~G~~ H] P2=
[E ~~G~~ F H ~~A~~ C B D] P3= [~~G~~
H ~~A~~ E B F C D] P4= [E F H
D B ~~A G~~ C]

Тогда предположим, что существуют следующие расстояния d(G,H) = 12, d(G,F) = 8, d(G,H) = 7, Так как в этом случае расстояние d(G,H) наименьшее, то следующей позицией у ребенка будет H, и так далее. Разрешение конфликтов реализуется так же, как и в двухродительской версии оператора. В случае AEX мы добавляли в ребенка последовательные позиции от последовательных родителей.

Мы провели эксперименты с различным количеством родителей и пришли к выводу, что для данной задачи оптимальным количеством родителей для модифицированного оператора кроссинговера HGreX является около 8. В результате применения нескольких родителей для кроссовера HGreX наблюдалось примерно двукратное сокращение числа итераций, но, что более важно, также были получены более короткие маршруты комплектации заказов (подробнее см. результаты экспериментов в разделе 6). С другой стороны, увеличение числа родителей для кроссовера AEX не привело к существенному изменению результатов.

5.4.2. Кэширование стоимости размещения товаров и длины маршрутов комплектации заказов

При оптимизации размещения продукции вычислительные усилия по расчету *себестоимости* (см. алгоритм 1) заданного размещения продукции и последующему определению на ее основе значения пригодности особи велики, так как требуется найти кратчайшие маршруты комплектации для всех заказов. Практически всегда либо некоторые родители переходят в следующее поколение, либо некоторые дети идентичны некоторым родителям (тем более на последних этапах оптимизации). В этом случае мы не вычисляем стоимость такой особи, а напрямую присваиваем уже вычисленную и кэшированную стоимость предыдущей идентичной особи (см. Алгоритм 1 и Рисунок 2). Мы также проверяем кэш после мутации.

С оптимизацией маршрутов комплектации заказов с помощью генетических алгоритмов дело обстоит иначе. Здесь вычислительные затраты на вычисление длины маршрута и определение пригодности особи невелики, и нет смысла реализовывать для этого кэширование. Однако стоимость вычисления кратчайшего маршрута для заказа (что может потребовать нескольких тысяч вычислений длин маршрутов, представленных всеми особями во всех итерациях оптимизации) гораздо выше, и здесь имеет смысл реализовать кэширование. Таким образом, перед вычислением кратчайшего маршрута для данного заказа $route_{min}(i, j)$ (см. алгоритм 2 и рисунок 4) проверяется, не содержал ли кэш уже маршрут для текущего заказа j, где все позиции товаров на складе были . Поясним, что если в кэше можно найти всю расстановку товаров, то подпроцесс расчета маршрута не вызывается из основного процесса, так как стоимость этой расстановки товаров *costPP* извлекается из кэша. Однако если размещение продукции отличается по некоторым позициям, то вызывается подпроцесс и проверяется для каждого заказа, существуют ли в кэше позиции в размещении продукции, занятые продуктами, содержащимися в текущих заказах. Если да, то из кэша извлекается длина $маршрутаR_{min}(i, j)$. В противном случае она вычисляется, а кэш обновляется. Кэш заказов используется только для расчета маршрута с помощью генетических алгоритмов. (См. Алгоритм 1 и Рисунок 2).

Кэширование не используется для алгоритма ближайшего соседа, поскольку затраты времени на кэш сопоставимы с затратами времени ближайшего соседа. По той же причине кэширование не используется для очень коротких заказов, где кратчайший маршрут определяется перестановками.

Очевидно, что кэширование не влияет на результаты оптимизации и лишь ускорить ее. Также стоит отметить, что кэширование более эффективно на поздних этапах оптимизации, так как в начале индивиды быстро меняются. В наших экспериментах кэширование позволило ускорить оптимизацию размещения товара в несколько раз (см. раздел 6).

5.4.3. Многократный перезапуск

Генетическим алгоритмам может потребоваться много итераций, чтобы прийти к оптимальному решению. Однако наиболее быстрый прогресс наблюдается в начале оптимизации. Генетические алгоритмы используют некоторые случайные числа и, таким образом, являются стохастическим процессом, в результате чего при последовательных запусках оптимизации могут быть найдены различные решения. Мы заметили, что при оптимизации размещения товара лучшим подходом является запуск оптимизации несколько раз только на несколько итераций и сохранение текущей популяции. Тогда оптимизация будет продолжаться только с популяцией наилучшего решения. Это разумный подход, так как чаще всего оптимизация, которая начинается как лучшая, заканчивается как лучшая. Таким образом, данный метод позволяет объединить эффективную по времени оптимизацию с хорошими результатами (см. экспериментальную проверку в разделе 6).

5.4.4. Группировка заказов

Все заказы, содержащие одинаковый набор продуктов, объединяются в один заказ. Таким образом, оптимальный маршрут комплектации для этого заказа определяется только один раз. Для расчета стоимости и удобства размещения товара длина полученного маршрута умножается на количество заказов, состоящих из одинаковых товаров (см. алгоритм 1).

5.4.5. Три метода оптимизации маршрутов

Как описано в разделе 5.3, для достижения оптимального баланса между скоростью вычислений и точностью результатов маршрут комплектации заказов в алгоритме 2 может быть оптимизирован с помощью перестановок (только короткие маршруты), генетических алгоритмов или алгоритма ближайших соседей.

5.4.6. Распараллеливание процессов

Генетические алгоритмы хорошо масштабируются для параллельной реализации в случаях, когда стоимость вычисления фитнес-функции высока, поскольку в этих случаях нет необходимости в частом обмене данными между потоками. Это как раз случай оптимизации размещения товара, когда использование любого количества ядер процессора вплоть до количества особей в популяции приводит к практически линейному увеличению производительности в функции от количества ядер процессора. Более того, если доступных процессорных ядер больше, чем размер популяции, то имеет смысл увеличить размер популяции, по крайней мере, в три раза, чтобы использовать больше процессорных ядер. Хотя после превышения оптимального размера популяции масштабирование с ростом числа процессоров перестает быть линейным, такая реализация очень проста. Другой альтернативой при наличии нескольких сотен доступных процессорных ядер является распараллеливание расчета конкретных маршрутов комплектации заказов, но поскольку у нас не было доступа к таким вычислительным ресурсам, мы не смогли проверить эффективность этого подхода.

6. Экспериментальные результаты

В этом разделе мы экспериментально оцениваем метод и усовершенствования, представленные в предыдущих разделах.

Мы проводили эксперименты с помощью собственного программного обеспечения, созданного на языке C#. Исходный код и данные, использованные в экспериментах (планы складов со списками соответствующих заказов), доступны

с веб-страницыwww.kordos.com/appliedsciences2020 . На рисунках 6 и 7 дополнительно представлены три структуры (поэтажные планы) этих складов и несколько примеров заказов для склада *w3*.

Оценивались следующие алгоритмы оптимизации маршрута комплектования заказов - ближайший сосед, генетические алгоритмы с кроссовером HGreX, генетические алгоритмы с многородительским кроссовером HGreX.

Оцениваются следующие алгоритмы оптимизации размещения товаров: генетические алгоритмы с кроссовером AEX, генетические алгоритмы с многородительским кроссовером AEX, генетические алгоритмы с многородительским кроссовером AEX и множественным перезапуском.

Поскольку мы не смогли найти в литературе полного автоматического решения для оптимизации размещения товаров, учитывающего маршруты комплектации заказов, как представленное здесь решение (см. разделы 1 и 4.2), мы, очевидно, не можем численно сравнить наше решение с другими решениями на тех же данных.

Сначала мы оценили многородительские модификации оператора кроссинговера HGreX, чтобы определить оптимальное количество родителей (см. разделы 5.3 и 5.4.1). Результаты представлены в таблице 2 и на рисунке 8. По результатам наших тестов размеры популяций порядка *N*= 80-120 обеспечивали наиболее быструю сходимость процесса (наименьшее количество вычислений фитнес-функций). При больших популяциях для достижения одних и тех же результатов приходилось проводить оценку фитнес-функции большее количество раз, поэтому даже если требовалось меньше итераций, время достижения результатов было больше [33]. Однако если популяции были меньше, то требовалось и больше оценок фитнес-функции, а если популяции были слишком малы, то сходимость алгоритма была невозможна. Только для оптимизации маршрута, когда количество товаров в заказе составляло 20 или менее, использовались популяции меньшего размера, а популяции большего размера могут быть полезны для более длинных хромосом, чем те, которые мы использовали в экспериментальной оценке.

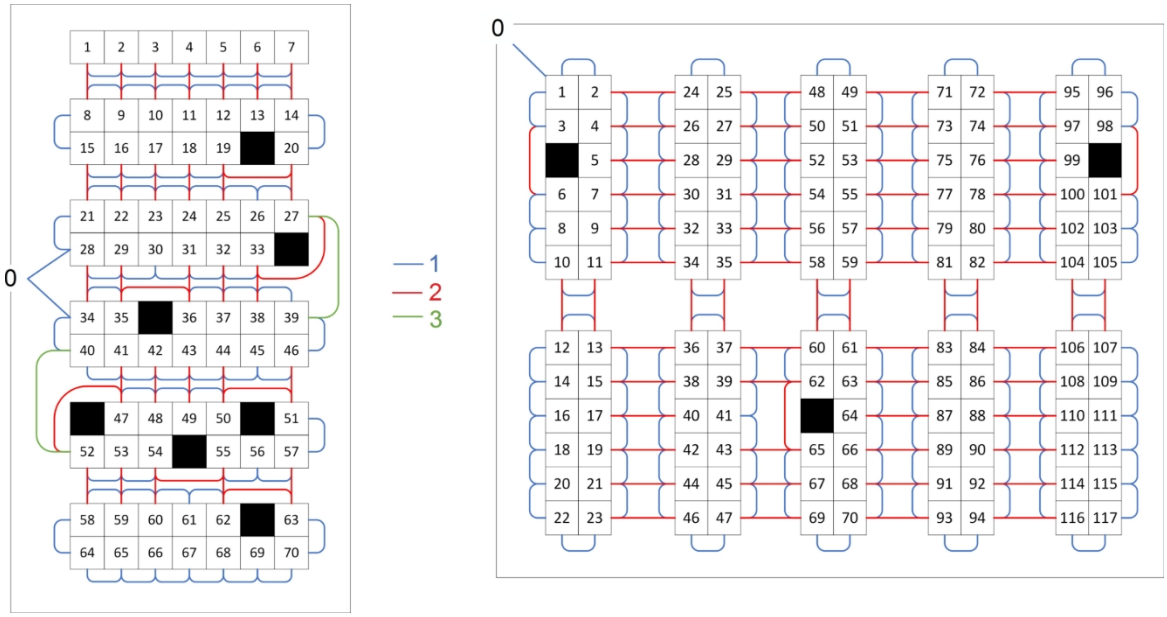

Рисунок 6. Образцы складских структур (поэтажные планы) складов w5 и w1, использованных в экспериментах. Каждая пронумерованная ячейка представляет собой одно местоположение товара. Синие линии показывают расстояние в 1 единицу, красные - в 2 единицы, зеленые - в 3 единицы.

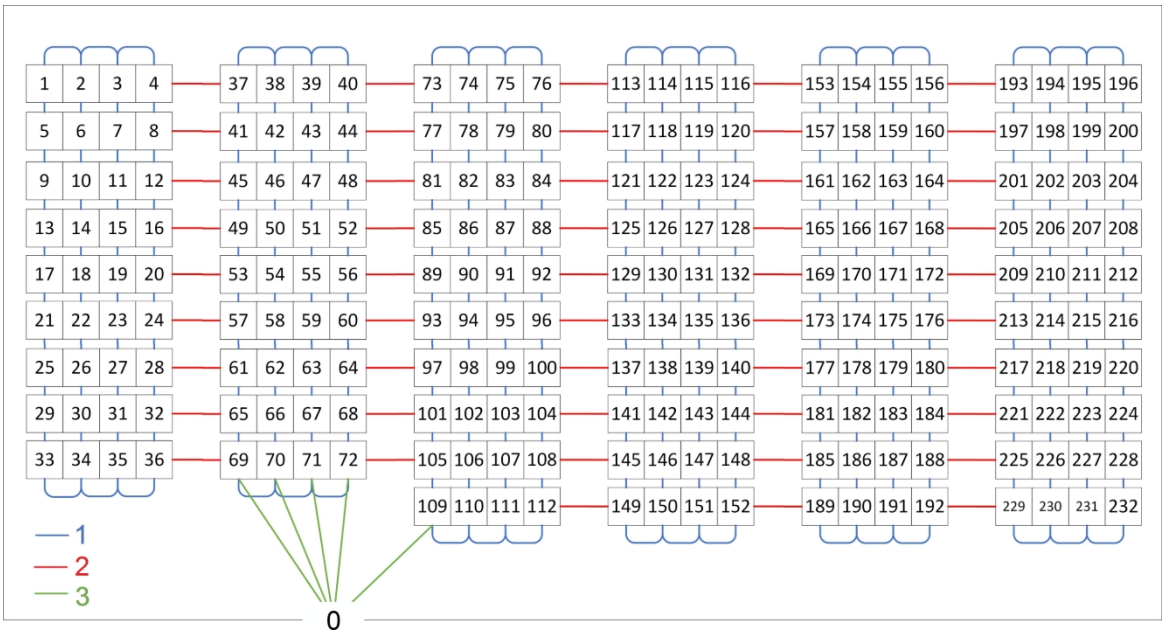

Рисунок 7. Пример структуры склада (план этажа) склада w2, использованного в экспериментах. Каждая пронумерованная ячейка представляет собой одно местоположение товара. Синие линии показывают расстояние в 1 единицу, красные - в 2 единицы, зеленые - в 3 единицы.

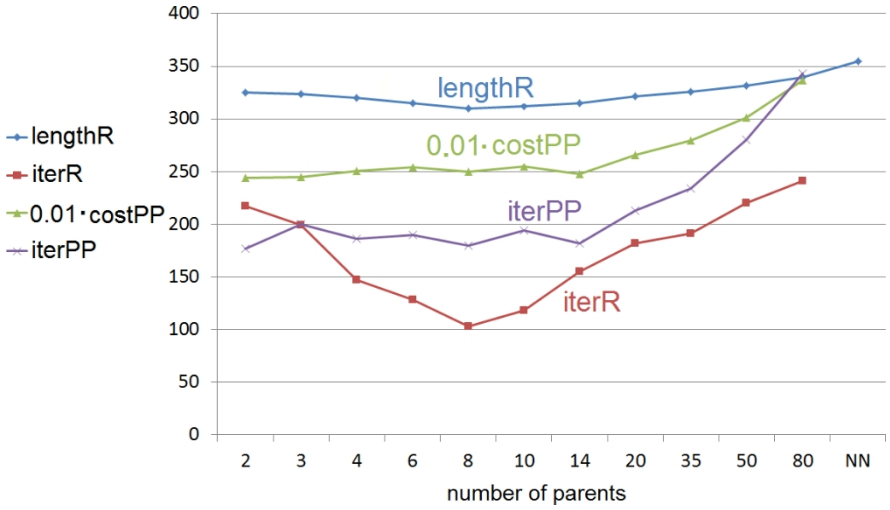

Рисунок 8. Полученные значения длины *маршрутаR* (чем меньше, тем лучше) и стоимости размещения *продукцииPP* (чем меньше, тем лучше), а также количество итераций для оптимизации маршрута *iterR* с помощью MP-HGreX и для оптимизации размещения продукции *iterP* с помощью MP-AEX (графическое представление данных из таблиц 2 и 3).

Критерием остановки для экспериментов, приведенных в таблице 2, было 20 итераций без улучшения лучшей особи. Количество итераций - это число итераций, после которых была найдена лучшая особь. Как видно, увеличение числа родителей в операторе кроссинговера однозначно снижает количество необходимых итераций (до двух раз для 8 родителей в данном) и, что более важно, позволяет получить лучшие значения фитнес-функции. Однако при использовании большего родителей, чем оптимальное, оптимизация снова замедляется, а при использовании более 20 родителей и полученные длины маршрутов начинают ухудшаться (увеличиваться). Как уже говорилось в разделе 5.3, при большом количестве родителей оператор HGreX ведет себя почти как метод поиска ближайших соседей, и его производительность стремится к тому же значению.

Таблица 2. Пример длины маршрута *lengthR* и количество итераций *iterR* для получения этой длины для модифицированного многородительского оператора кроссинговера HGreX для заказа из 60 продуктов с фиксированным размещением продуктов (среднее значение из 10 прогонов). *NN* в последнем столбце обозначает результат, полученный для алгоритма ближайшего соседа.

Количество родителей	**2**	**3**	**4**	**6**	**8**	**10**	**14**	**20**	**35**	**50**	**80**	**NN**
средняя *длинаR*	325	323	320	315	310	312	315	321	326	331	339	355
agv. *iterR*	217	199	147	128	103	124	155	182	191	220	243	-
стд. откл. *длинаR*	3.5	3.8	3.9	3.4	3.4	3.8	4.0	5.4	5.2	9.2	8.8	-
std. *iterR*	46	52	39	26	33	31	38	68	70	79	67	-

Количество итераций, используемых генетическим алгоритмом с кроссовером HGreX, сопоставимо с тем, которое требуется другим современным операторам кроссовера для поиска кратчайшего маршрута при сопоставимом размере популяции [34]. Даже если оптимизация в течение большего числа итераций может найти немного более короткий маршрут, обычно это не дает дополнительного выигрыша в стоимости размещения продукции, поскольку это лишь очень редко вызывает изменение расположения продукции. Для очень редко встречающихся заказов достаточно запустить оптимизацию на меньшее количество эпох или использовать алгоритм ближайших соседей, независимо от длины заказа, поскольку наиболее быстрое улучшение происходит в начале оптимизации, а влияние на оптимальное размещение продукции очень редких заказов также очень мало, поскольку они доминируют над более частыми заказами.

Далее мы проверили полезность многородительского кроссинговера AEX в оптимизации размещения товаров. Поскольку AEX не использует стоимость переходов между двумя элементами, родители для каждого элемента выбирались случайным образом. Результаты для склада с 232 местами (длина хромосомы составляла 232 элемента) представлены в таблице 3. На основании проведенных экспериментов мы пришли к выводу, при оптимизации размещения товаров достаточно использовать двухродительский кроссинговер AEX, так как увеличение числа родителей не приводило к выигрышу, а при числе 20 и более наблюдалось падение эффективности метода.

Таблица 3. Стоимость размещения продукции *costPP* как сумма длин маршрутов комплектации заказов (чем меньше, тем лучше - см. уравнение 2) и количество итераций *iterP* для получения этой стоимости для модифицированного многородительского оператора кроссовера AEX для склада размером 232 места и списка из 80 заказов, использующего 8-родительский HGreX для оптимизации маршрутов (средние значения из 10 прогонов).

Количество. Родители	**2**	**3**	**4**	**6**	**8**	**10**	**14**	**20**	**35**	**50**
средняя *стоимостьПП*	24,397	24,456	25,054	25,410	24,969	25,511	24,785	26,607	27,949	30,014
лучшая *стоимостьРР*	21,731	21,410	21,615	22,535	22,861	21,945	21,476	21,474	25,455	26,878
agv. *iterP*	177	200	186	190	180	202	182	213	234	280
стд. откл. *стоимостьПП*	2586	2777	2158	1831	3165	3110	2660	3106	3250	2996
std. *iterP*	44	79	76	86	101	104	79	98	112	106

Многократный перезапуск оптимизации размещения товаров с кроссовером AEX (MR-AEX) оказался весьма полезным (см. Алгоритм 1). В последней строке таблицы 4 оптимизация перезапускалась 5 раз, каждый раз выполнялось по 10 итераций, а затем мы продолжали только с популяцией лучшей особи, как описано в разделе 5.4.3. Это позволило не только получить меньшую стоимость, но и стандартное отклонение результатов оказалось примерно в два раза ниже.

В экспериментах, представленных в таблице 4, мы использовали размер популяции в 100 особей для оптимизации размещения товаров. Для оптимизации маршрута комплектации заказов мы использовали размер популяции в 100 особей, если количество товаров составляло 25 и более, и в четыре раза большее количество товаров для более коротких заказов. Причина выбора такого размера популяции заключается в том, что основная стоимость генетических алгоритмов - это оценка фитнес-функции (особенно для оптимизации размещения продукции). Количество оценок фитнес-функции может быть выражено умножением размера популяции на количество эпох. Используя большие популяции, мы можем получить те же результаты за меньшее количество эпох. Для меньших популяций также необходимо увеличить скорость мутации. Однако зависимость между размером популяции и количеством необходимых эпох не является линейной, и существует оптимальная популяция

размер, который позволяет получить наименьшее количество оценок фитнес-функции [33]. Это число также зависит от задачи и других параметров генетических алгоритмов. В наших экспериментах минимум обычно достигался при размере популяции от 80 до 120 особей при количестве мест на складе от 60 до 300, а затем он очень медленно рос, но гораздо медленнее, чем линейно, с увеличением склада. В районе минимума зависимость была очень плоской (изменение размера популяции, например, с 80 до 100 особей не давало статистически значимой разницы в количестве необходимых оценок фитнес-функции). Однако за пределами этого диапазона зависимость была более существенной, и, например, использование 1000 особей позволило уменьшить количество эпох только примерно в 3 раза, что фактически увеличило количество оценок фитнес-функции примерно в 3 раза. Более того, при слишком малом размере популяции не только увеличивалось время процесса, но и он становился нестабильным и часто не мог сходиться. По этой причине размер популяции в 100 особей был более безопасным выбором, чем, например, 80 особей.

Мы использовали коэффициенты мутации по умолчанию в уравнении (4): c_i= 0,00001, c_n= 0,00001, c_f= 0,3 как при размещении товаров, так и при оптимизации маршрутов.

Ниже мы приводим несколько примеров заказов для склада *w3*. Продукты в заказах кодируются числами, которые являются идентификаторами продуктов (мы не можем использовать буквы, как в примерах из предыдущих разделов, поскольку в алфавите недостаточно букв). Последнее число (N_{rep}) каждого заказа показывает, сколько раз такой заказ встречается в списке заказов, поэтому длина маршрута его выполнения может быть оценена только один раз и затем умножена на N_{rep} при расчете конечной стоимости размещения товара.

order1: 41 99 97 7 20 89 12 24 51 66 79 61 1 56 109 N_{rep}= 40
order2: 71 90 9 29 84 94 19 26 64 114 100 42 81 30 108 107 101 47 6 32 96 33 28 7 N_{rep}= 20
заказ3: 78 31 91 35 93 87 22 50 100 1 28 38 84 16 48 112 76 110 95 47 72 113 23 61 101 68 67 53 45 41 97 18 109 89 65 74 N_{rep}= 3
заказ4: 53 61 18 36 94 24 103 38 35 12 42 89 6 30 50 14 84 114 29 15 79 95 48 52 28 25 110 22 64 109 44 11 73 33 98 97 23 75 99 87 7 51 92 93 72 17 3 N_{rep}= 1

Таблица 4. Полученная стоимость размещения продукции (чем ниже, тем лучше) как сумма всех маршрутов комплектации заказов (см. уравнение (2)) для методов оптимизации размещения продукции и маршрутов комплектации заказов с различными улучшениями (см. раздел 5.4) для шести образцов складов: w1, w2, w3, w4, w5, w6 с соответствующими списками заказов, в среднем за 10 прогонов оптимизации. Время выполнения представлено в таблице 5.

Прод. Рlc. Оптимиз.	Маршрут Optimiz.	Стоимость	w1	w2	w3	w4	w5	w6	Стоимость по сравнению с рандами.	t-тест Вилкокса.
случайный	случайный	в среднем	29,346	49,374	42,942	14,156	16,934	40109	1.000	
		стд.откл.	5445	9447	9050	2872	3747	7792	0.207	0.0001
случайный	N.Neighbor	в среднем	16,931	32,879	27,501	9325	9102	20314	0.596	0.00001
		стд.откл.	2399	4444	3686	1354	1220	2915	0.083	0.0104
случайный	HGreX	в среднем	14,889	31,382	26,543	8950	8080	19060	0.575	0.00001
		стд.откл.	2238	4139	3246	1047	995	2328	0.071	0.0102
случайный	MP-HGreX	в среднем	14,342	30,660	26,004	8206	7917	18667	0.544	0.00001
		среднее отклонение.	1529	3074	2042	684	712	1529	0.048	0.0001
AEX	H.Сосед	в среднем	6398	11,198	10,039	7274	3278	7400	0.262	0.00001
		среднее отклонение.	418	747	619	525	212	498	0.018	0.0001
AEX	HGreX	в среднем	5989	10760	9516	6048	3160	6898	0.238	0.00001
		стд.откл.	550	809	826	453	253	544	0.019	0.0038
AEX	MP-HGreX	в среднем	5694	10,767	9003	6000	3164	6778	0.227	0.00026
		среднее отклонение.	382	649	533	403	221	441	0.015	0.0011
MR-AEX	MP-HGreX	в среднем	5394	9676	8543	5312	3023	6428	0.213	0.00001
		стд.откл.	171	339	289	174	101	226	0.007	

В таблице 4 представлены подробные результаты для шести примеров структуры складов и списков заказов (это максимальное количество складов, которое может поместиться в одной строке таблицы). MP-HGreX означает Multi-Parent HGreX с 8 родителями, MR-AEX - Multiple-Restart AEX с 5 перезапусками, сохранением популяции после 10 итераций, а затем продолжением популяции лучшей особи (см. раздел 5.4.3 и рисунок 2 для пояснений). В таблице 5 представлено реальное время работы процессов оптимизации (включая операции ввода-вывода и алгоритм Дейкстры).

Таблица 5. Реальное время работы процессов оптимизации в секундах на компьютере с двумя процессорами Xeon X5-2696-v2, усредненное за 10 прогонов оптимизации для экспериментальных данных, представленных в таблице 4, и дополнительно для AEX/HGreX без кэша (см. раздел 5.4.2).

Прод. Plc.	**Маршрут**	**w1**	**w2**	**w3**	**w4**	**w5**	**w6**
случайно	случайно	0	0	0	0	0	0
случайно	H.Сосед	1.4	2.0	2.0	1.0	1.0	1.8
случайно	HGreX	3.1	4.4	3.7	2.0	2.1	3.4
случайно	MP-HGreX	3.2	4.8	3.7	2.0	2.1	3.4
AEX	H.Сосед	130	204	169	59	77	186
AEX	HGreX	744	1329	1121	423	495	1077
AEX	MP-HGreX	762	1334	1077	385	450	1133
MR-AEX	MP-HGreX	1441	1989	1667	762	827	1998
AEX/HGreX, без кэша		3127	5678	4465	1710	2077	5225

Возможное снижение стоимости размещения продукции зависит от характера заказов. Наибольшее улучшение за счет оптимизации маршрута может быть получено для заказов, содержащих длинный список товаров. Наибольшее улучшение за счет оптимизации размещения продукции может быть достигнуто, если в заказах часто встречаются товары определенных групп и частота появления определенных товаров в заказах сильно различается. Таким образом, возможное улучшение определяется в основном свойствами заказов. Таким образом, конкретные методы должны сравниваться между собой для одной и той же структуры склада и для одного и того же списка заказов (это похоже на классификацию, где возможная точность зависит от свойств набора данных, и различные классификаторы должны сравниваться на одних и тех же данных).

Столбец *cost vs. rnd.* в таблице 4 содержит среднее относительное снижение стоимости размещения продукции, рассчитанное как $cost\ vs.\ rnd. = Average(Sum(F1p(w)/random(w))$, где w= 1...6 - номер склада. Последний столбец содержит тесты статистической значимости, рассчитанные на всех данных между двумя соседними методами, поэтому он выводится между строками сравниваемых методов. Поскольку одни предпочитают для таких Т-тест, а другие - ранговый тест с признаками Вилкоксона, мы использовали оба теста, чтобы удовлетворить всех. Поскольку все p-значения в последнем столбце таблицы 4 меньше 0,05, можно предположить, что все методы значительно отличаются друг от друга.

Как видно из рисунков 9 и 10, наилучшие результаты получены при многократном перезапуске генетических алгоритмов с оператором кроссинговера AEX (MR-AEX) для оптимизации размещения товаров (см. раздел 5.4.3 и алгоритм 1) и генетического алгоритма с многородительским оператором кроссинговера HGreX (MP-HGreX) для оптимизации маршрута комплектования заказов (см. рисунок 4).

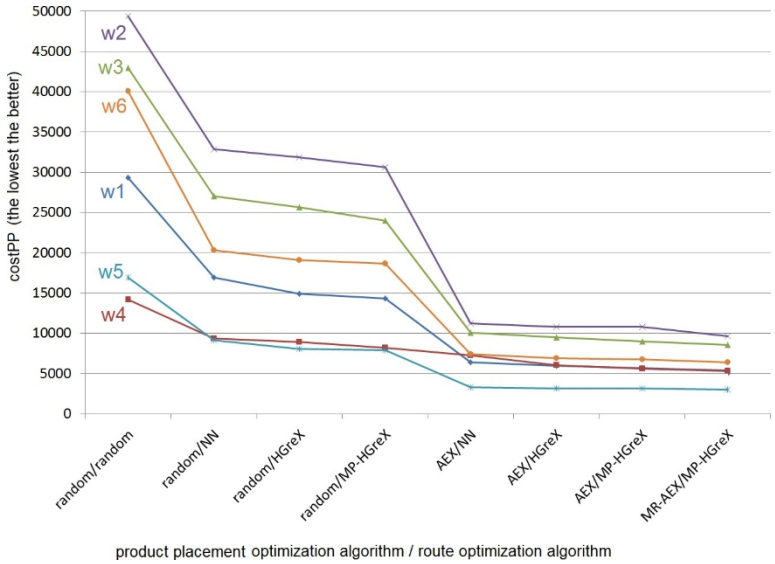

Рисунок 9. Графическое представление данных из таблицы 4). По горизонтальной оси: методы оптимизации с различными улучшениями (см. раздел 5.4). По вертикальной оси: стоимость размещения продукции *costPP* (чем ниже, тем лучше), полученная с помощью конкретных методов для складов w1, w2, w3, w4, w5, w6 с соответствующими списками заказов.

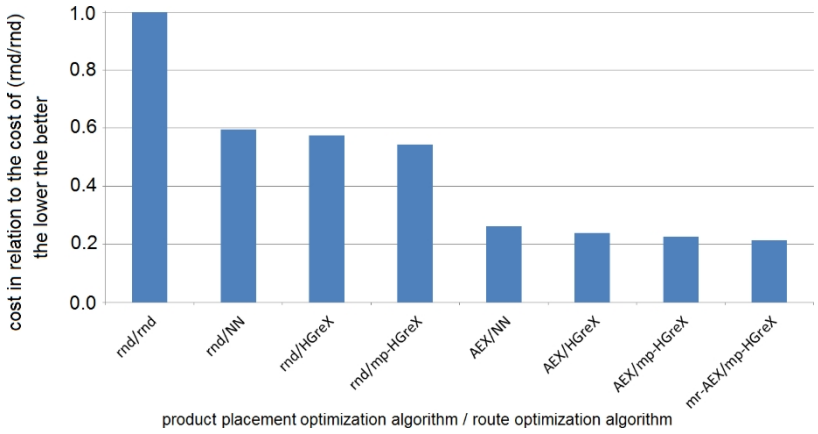

Рисунок 10. Сравнение производительности представленных методов с различными улучшениями (см. раздел 5.4). По горизонтальной оси: метод оптимизации. По вертикальной оси: средняя полученная стоимость размещения товара (чем ниже, тем лучше) в процентах от стоимости при случайном размещении товара и случайных маршрутах для шести складов со списками заказов, представленными в таблице 4.

7. Выводы

Сокращение времени комплектации заказов - важнейший и наиболее выгодный фактор снижения затрат на эксплуатацию склада (где, как правило, 60 % расходов приходится на комплектацию заказов [1]). Этого можно достичь без значительных инвестиций, оптимизировав расположение конкретных товаров на складе и определив наиболее быстрые маршруты комплектации заказов. Поскольку пространство поиска решений огромно ($9{,}3\times 10^{157}$ возможных размещений для 100 товаров, $3{,}1\times 10^{10614}$ для 300 товаров), задача не может быть проанализирована методами грубой силы. Поэтому мы представили полностью автоматическую систему на основе генетических алгоритмов, которая за счет интеллектуального поиска позволяет находить оптимальные варианты размещения товаров в течение нескольких минут или десятков минут для такого размера задачи (в зависимости от аппаратного обеспечения компьютера и параметров процесса). Несмотря на то что нахождение оптимального решения с помощью генетических алгоритмов не гарантировано, можно найти решение, очень близкое к оптимальному, так что на практике это не будет иметь существенного значения.

Представленная система принимает на вход структуру склада (в виде частичных затрат на переход) и список заказов и возвращает оптимальное размещение продукции и соответствующие кратчайшие маршруты комплектации заказов. Внедрение такой системы позволяет ускорить комплектацию заказов и тем самым снизить эксплуатационные расходы склада. Это позволяет обслуживать больше клиентов тем же количеством сотрудников за то же время, а значит, еще больше увеличить продажи и прибыль.

Эксперименты показали, что использование многородительского кроссовера HGreX улучшает результаты, в то время как для кроссовера AEX добавление большего числа родителей не меняет его эффективности. Наилучшие результаты были получены для генетического алгоритма с множественным перезапуском и оператором кроссовера AEX (MR-AEX) для оптимизации процесса размещения товаров и генетического алгоритма с многородительским оператором кроссовера HGreX (MP-HGreX) для оптимизации маршрута комплектации заказов. Для ускорения процесса может использоваться кэширование стоимости или длины маршрута. Кроме того, для оптимизации маршрута можно использовать алгоритм ближайших соседей, чтобы еще больше ускорить процесс, но это обычно происходит за счет немного худшего результата.

В дальнейших работах мы планируем реализовать другие модификации для дальнейшего повышения скорости оптимизации и качества получаемых решений. Прежде всего мы хотим оценить новые операторы кроссинговера и смеси различных операторов. В экспериментальном сравнении Puljic [19] смесь различных операторов кроссинговера показала себя немного лучше, чем HGreX. Также были предложены более продвинутые смеси генетических операторов [35,36]. Однако мы не стали использовать этот подход, так как его реализация, безусловно, сложнее. Вместо этого мы модифицировали HGreX для использования нескольких родителей, что значительно улучшило результаты. Лапа и др. [37] предложили использовать различные операторы (не только кроссинговеры, но и различные мутации и другие операторы) для разных особей в стандартных генетических алгоритмах. Мы собираемся применить эти подходы к оптимизации маршрутов и размещения товаров и исследовать различные варианты.

Другая ветвь наших будущих исследований связана с ограничениями в работе генетического алгоритма, поскольку на некоторых складах такие ограничения могут существовать и ограничивать возможное размещение определенных товаров. В литературе типичным подходом к ограничениям в генетических алгоритмах является использование штрафных функций [38]. Иногда также используются методы, основанные на доминировании [39]. Однако мы собираемся реализовать это иначе, встроив механизм непосредственно в операторы кроссинговера и мутации, специфичные для данной задачи, чтобы эффективно реализовать ограничение и ограничить вычислительную сложность оптимизации.

Авторский вклад: Концептуализация, М.К. и С.Г.; Формальный анализ, М.Б. и С.Г.; Получение финансирования, М.Б. и С.Г.; Исследование, М.К. и Дж.Б.; Методология, М.К., С.Г. и Дж.Б.; Программное обеспечение, М.К. и Дж.Б.; Валидация, М.Б. и С.Г.; Визуализация, М.Б. и С.Г.; Написание первоначального варианта, М.К., Дж.Б., М.Б. и С.Г. Все авторы прочитали и согласились с опубликованной версией рукописи.

Финансирование: Работа выполнена при поддержке проекта Силезского технологического университета: BK-204/2020/RM4.

Благодарности: Авторы выражают благодарность Михалу Кшизу˙ owski, Лукашу Мыслаеку, Антони Копецу и Якубу Гавееду за помощь в сборе и подготовке данных, использованных в данном исследовании.

Конфликты интересов: Авторы заявляют об отсутствии конфликта интересов. Спонсоры-учредители не принимали участия в разработке дизайна исследования, сборе, анализе и интерпретации данных, написании рукописи и принятии решения о публикации результатов.

Ссылки

1. Бартольди Ж.Ж., Хакман С.Т. Наука о складах и дистрибуции // Складское хозяйство. 2019. Доступно онлайн: https://www. warehouse-science.com/book/index.html (дата обращения: 30 мая 2020 г.).
2. Авдейкинс А., Саврасов М. Умная логистика склада с помощью эффективной стратегии размещения на основе генетических алгоритмов. *Transp. Telecommun.* **2019**, *20*, 318-324. [CrossRef].
3. Боланьос Зуньига Х., Сауседо Мартинес Х.А., Салаис Фьерро Т.Е., Мармолехо Сауседо Х.А. Оптимизация задачи назначения места хранения и маршрутизации комплектовщика с помощью математического программирования. *Appl. Sci.* **2020**, *10*, 534. [CrossRef].
4. Ван Гилс Т., Рамакерс К., Карис А., Де Костер Р. Проектирование эффективных систем комплектации заказов путем объединения задач планирования: Классификация и обзор современного состояния. *Eur. J. Oper. Res.* **2018**, *267*, 1-15. [CrossRef].

5. Wang, W.; Gao, J.; Gao, T.; Zhao, H. Optimization of Automated Warehouse Location Based on Genetic Algorithm. В материалах 2-й Международной конференции по управлению, автоматизации и искусственному интеллекту (CAAI 2017), Санья, Китай, 25-26 июня 2017 г.; стр. 309-313.
6. Гроссе Е.Х., Глок К.Х., Нойманн П.В. Человеческие факторы при комплектации заказов: Контент-анализ литературы . *Int. J. Prod. Res.* **2016**, *55*, 1260-1276. [CrossRef].
7. Dijkstra, A.; Roodbergen, K. Точные формулы длины маршрута и эвристика назначения места хранения для складов . *Transp. Res. Part E* **2017**, *102*, 38-59. [CrossRef].
8. Ракеш, В.; Кадил, Г. Оптимизация планировки трехмерного склада для комплектации заказов. *IFAC-PapersOnLine* **2017**, *48*, 1155-1160. [CrossRef].
9. Даварзани, Х.; Норрман, А. На пути к актуальной повестке дня для исследований в области складского хозяйства: обзор литературы и практиков. *Logist. Res.* **2015**, *8*, 1-18. [CrossRef].
10. Зунич Е., Беширевич А., Скробо Р., Хашич Х., Ходжич К., Джедович А. Разработка системы оптимизации для комплектации заказов на складе в реальной среде. В сборнике трудов XXVI Международной конференции по информационным, коммуникационным и автоматизированным технологиям, Сараево, Босния и Герцеговина, 26-28 октября 2017 г.; Том 26.
11. Dharmapriya, U.; Kulatunga, A. Новая стратегия оптимизации склада - бережливое складирование. В материалах Международной конференции по промышленной инженерии и операциям 2011 года, Куала-Лумпур, Малайзия, 22-24 января 2011 года.
12. Аффенцеллер М., Вагнер С., Винклер С., Бехам А. *Генетические алгоритмы и генетическое программирование: Modern Concepts and Practical Applications*; CRC Press: Boca Raton, FL, USA, 2018.
13. Саймон, Д. *Эволюционные алгоритмы оптимизации*; Wiley: New York, NY, USA, 2013.
14. Лавринович, А. *Генетические алгоритмы для перспективного планирования и составления графиков в сетях поставок*; Difin: Варшава, Польша, 2013.
15. Xu, W.; Jia, H. Исследование оптимизации расположения хранилища на основе генетических алгоритмов. *J. Phys. Conf. Series* **2019**, *1213*, 032020. [CrossRef].
16. Хассанат А.Б.А., Алькафавин Э. Об улучшении генетических алгоритмов с помощью новых кроссоверов. *Int. J. Comput. Appl. Technol.* **2017**, *55*. [CrossRef].
17. Хванг, Х. Модель улучшения для задачи маршрутизации транспортных средств с ограничением по времени на основе генетического алгоритма. *Comput. Ind. Eng.* **2002**, *42*, 361-369. [CrossRef].
18. Тан, Х.; Ли, Л.Х.; Жу, К.; Оу, К. Эвристические методы для задачи маршрутизации транспортных средств с временными окнами. *Artif. Intell. Eng.* **2001**, *16*, 281-295. [CrossRef].
19. Пулич, К.; Мангер, Р. Сравнение восьми эволюционных операторов кроссовера для задачи маршрутизации транспортных средств. *Математика. Commun.* **2013**, *18*, 359-375.
20. Даварзани, Х.; Норрман, А. Замечание о двух проблемах, связанных с графами. *Numer. Math.* **1959**, *1*, 269-271.
21. Флойд, Р.У. Алгоритм 97: Кратчайший путь. *Commun. ACM* **1962**, *5*. [CrossRef].
22. Беллман, Р. О проблеме маршрутизации. *Q. Appl. Math.* **1958**, *16*, 87-90. [CrossRef].
23. Харт П.Е., Нильссон Н.Дж., Рафаэль Б. Формальная основа для эвристического определения путей с минимальной стоимостью. *IEEE Trans. Syst. Sci. Cybern.* **1968**, *4*, 100-107. [CrossRef].
24. Крейнович В., Олак Л.Ф., Кинтана К. Генетические алгоритмы: Какое масштабирование фитнеса является оптимальным? *Cybern. Syst.* **2001**, *24*. [CrossRef].
25. Разали, Н.М.; Герагти, Дж. Производительность генетического алгоритма с различными стратегиями выбора при решении TSP. В материалах Всемирного инженерного конгресса WCE 2011, Лондон, Великобритания, 6-8 июля 2011 года.
26. Хассанат, А.Е.А. Выбор коэффициентов мутации и кроссовера для генетических алгоритмов - обзор с новым динамическим подходом. *Информация* **2019**, *10*, 390. [Электронный ресурс].
27. Отман, А.; Таяни, К.; Абучабака, Ж. Анализ эффективности операторов мутации для решения задачи о путешествующем коммивояжере . *arxiv* **2012**, arXiv:1203.3099.
28. Нильссон, К. *Эвристики для задачи о путешествующем продавце*; технический отчет; Университет Линчепинга: Linkoping, Sweden, 2003.
29. Кааби, Дж.; Харрат, Й. Правила перестановки и генетический алгоритм для решения задачи о коммивояжере. *Араб. J. Basic Appl. Sci.* **2019**, *26*, 283-291. [CrossRef].
30. Кордос, М.; , К. Многоцелевой эволюционный выбор экземпляров для задач регрессии. *Энтропия* **2018**, *20*, 746. [CrossRef].

31. Kordos, M.; Arnaiz-González, A.; García-Osorio, C. Multi-Objective Evolutionary Instance Selection for Regression Tasks. *Neurocomputing* **2019**, *358*, 309-320. [CrossRef].
32. Деб К., Пратап А., Агарвал С., Мейариван Т. Быстрый и элитарный многоцелевой генетический алгоритм: NSGA-II. *IEEE Trans. Evol. Comput.* **2002**, *6*, 182-197. [CrossRef].
33. Кордос, М. Оптимизация эволюционного выбора экземпляров. *Лекция. Notes Artif. Intell.* **2017**, *10245*, 359-369.
34. Xin, J.; Zhong, J.; Yang, F.; Cui, Y.; Sheng, J. An Improved Genetic Algorithm for Path-Planning of Unmanned Surface Vehicle. *Sensors* **2019**, *19*, 2640. [CrossRef].
35. Контрерас-Болтон, К.; Парада, В. Автоматическая комбинация операторов в генетическом алгоритме для решения задачи о путешествующем коммивояжере. *PLoS ONE* **2015**, *26*. [CrossRef].
36. Контрерас-Болтон, К.Э. Алгоритмы для вариантов задач маршрутизации. Ph.D. Thesis, Università di Bologna, Bologna, Italy, 2019.
37. Лапа, К.; Чпалка, К.; Ласковски, Л.; Кадер, А.; Зенг, З. Эволюционный алгоритм с настраиваемым механизмом поиска . *J. Artif. Intell. Soft Comput. Res.* **2020**, *10*, 151-157. [CrossRef].
38. Чехури А., Юнес Р., Перрон Ж., Илинка . Техника обработки ограничений для генетических алгоритмов с использованием коэффициента нарушения. *J. Comput. Sci.* **2016**. [CrossRef].
39. Ponsich, A.; Azzaro-Pantel, C.; Domenech, C.; Pibouleau, L. Стратегии обработки ограничений в генетических алгоритмах в применении к оптимальному проектированию установок периодического действия. *Chem. Eng. Процесс. Процесс. Intensif.* **2008**, *47*, 420-434. [CrossRef].