Автономная некоммерческая организация высшего образования   
«Русский университет метатехнологий»

Кафедра программных систем

Направление подготовки: 09.03.04 Программная инженерия

Профиль подготовки: Проектирование и разработка программных систем

на тему:

**«Оптимальное распределение товаров на складе»**

|  |
| --- |
| Выполнил:  студент группы ПС–21  очной формы обучения  Яшметов Кирилл Романович  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / Яшметов К.Р.  подпись расшифровка |
| Руководитель:  доцент, канд. физ-мат. наук  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / Козлов А.И.  подпись расшифровка |

Йошкар-Ола

2025

Оглавление

[Введение 3](#_Toc191025860)

[1. Аналитическая часть 4](#_Toc191025861)

[1.1. Обзор существующих алгоритмов 4](#_Toc191025862)

[1.1.1. Название Алгоритма 1 4](#_Toc191025863)

[1.1.2. Название Алгоритма 2 4](#_Toc191025864)

[1.1.3. Название Алгоритма 3 4](#_Toc191025865)

[1.2. Название выбранного алгоритма 5](#_Toc191025866)

[1.2.1. Доказательство корректности алгоритма. 5](#_Toc191025867)

[1.2.2. Трудоёмкость алгоритма. 5](#_Toc191025868)

[1.2.3. Блок-схема алгоритма 5](#_Toc191025869)

[2. Конструкторская часть 6](#_Toc191025870)

[2.1. Выбор языка программирования 6](#_Toc191025871)

[2.2. Описание основной функции 6](#_Toc191025872)

[2.3. Использованные технологии 6](#_Toc191025873)

[2.4. База данных 6](#_Toc191025874)

[Заключение 8](#_Toc191025875)

[Cписок использованных источников 9](#_Toc191025876)

[ПРИЛОЖЕНИЕ А 10](#_Toc191025877)

[ПРИЛОЖЕНИЕ Б 11](#_Toc191025878)

# Введение

В современном мире оптимизация логистических процессов играет ключевую роль в повышении эффективности и конкурентоспособности предприятий. Одним из важнейших аспектов логистики является работа склада, где значительная часть затрат связана с комплектацией заказов. Задача сокращения времени и стоимости комплектации заказов является актуальной и требует применения эффективных методов оптимизации.

Имеется склад, который представляется в виде неориентированного связного графа. Вход, выход со склада и места хранения товаров – вершины графа. Ребра графа имеют стоимость, которую можно представить как расстояние или время для достижения этого места на складе. Необходимо собрать заказ за минимальное время. Заказ состоит из товаров, которые мы размещаем на складе. Для каждого товара отводится отдельное место. До каждого места определена длина пути через матрицу расстояний.

Целью работы является нахождение оптимального расположения товаров в зависимости от частоты заказов и оптимального маршрута их размещения на складе(графе).

Задачами данной работы являются:

1. Обзор существующих алгоритмов для нахождения оптимального маршрута при обходе графа для комплектации заказов на складе.
2. Определение оптимального алгоритма для размещения товаров на складе(в вершинах графа) для наиболее быстрого сбора заказов.
3. Написание программы для реализации алгоритма.
4. Экспериментальная проверка алгоритма на реальных данных.
5. Разработка пользовательского приложения для использования алгоритма.

# 1. Аналитическая часть

## 1.1. Обзор существующих алгоритмов

Для решения задачи поиска оптимального распределения товаров на складе будем представлять склад в виде неориентированного связного графа. Ребра графа будут иметь вес равный расстоянию или времени перемещения между вершинами, которые выполняют функцию мест для расположения товаров. В рамках данной темы мы будем рассматривать алгоритмы для достижения двух целей:

1. Оптимальное размещение товаров
2. Кратчайшие маршруты для комплектования заказов. В размещении товаров

Проблема поиска кратчайшего маршрута эквивалентна задаче о коммивояжере. Оптимальное размещение товаров более сложная задача. Основное отличие одной от другой в том, что при размещении товаров целевая функция – это сумма оптимальных времен (или длин) маршрутов для каждого заказа при данном размещении товаров. В TSP(Traveling Salesman Problem) целевая функция – просто длина одного маршрута. Из-за этого задача размещения двухуровневая. На внешнем уровне мы оптимизируем расположение товаров. А на внутреннем уровне, чтобы оценить каждое возможное размещение, мы должны решить задачу TSP (или использовать другой алгоритм маршрутизации) для каждого заказа и просуммировать результаты. TSP – это одноуровневая задача. Для решения этих задач существует множество алгоритмов. Мы будем использовать эвристические, потому что они дают приемлемое время выполнения. Среди эвристических алгоритмов можно обратить внимание на генетические алгоритмы, алгоритм имитации отжига, алгоритм муравьиной колонии. В данной работе для решения задачи оптимального размещения товаров и поиска кратчайших маршрутов комплектования заказов выбран генетический алгоритм. Этот выбор обусловлен следующими факторами:

* генетические алгоритмы могут быть адаптированы к различным типам складов, заказов и критериям оптимизации. Операторы кроссовера и мутации могут быть настроены для учета специфических особенностей задачи.
* в статье[[1](#_Cписок_использованных_источников)] демонстрируется успешное применение генетических алгоритмов для решения данной задачи.

## 1.2. Описание выбранного алгоритма

Генетический алгоритм, выбранный для решения задачи оптимального размещения товаров на складе, представляет собой итеративный процесс, имитирующий биологическую эволюцию для поиска наилучшего решения. Алгоритм начинается с инициализации популяции, состоящей из N хромосом, где каждая хромосома кодирует возможное размещение товаров на складе. Хромосома, как правило, представляется в виде массива, где позиция элемента соответствует позиции товара на складе. Начальные хромосомы генерируются случайным образом. Далее, для каждой хромосомы в популяции вычисляется целевая функция, определяющая приспособленность решения. В контексте складской логистики целевая функция представляет собой суммарную длину (или время) оптимальных маршрутов комплектации всех заказов для данного размещения товаров. После оценки приспособленности выполняется селекция, в ходе которой отбираются хромосомы для участия в процессе кроссовера. Хромосомы с более высокой приспособленностью имеют больший шанс быть выбранными. Существуют различные методы селекции, такие как рулетка, турнирная селекция и ранговая селекция. Отобранные хромосомы подвергаются кроссоверу, в ходе которого происходит обмен генетической информацией между родительскими хромосомами для создания новых хромосом (потомков). Существуют различные операторы кроссовера, такие как одноточечный, двухточечный, равномерный кроссовер, оператор HGreX и прочие. После кроссовера выполняется мутация, вносящая случайные изменения в хромосомы потомков. Мутация позволяет привнести разнообразие в популяцию и избежать застревания в локальных оптимумах. Существуют различные операторы мутации, такие как перестановка, инверсия и вставка. Новые хромосомы (потомки) заменяют часть старых хромосом в популяции. Существуют разные стратегии замены, такие как замена худших, элитизм и турнирная замена. Алгоритм останавливается, когда выполнено определенное условие останова, например, достигнуто максимальное количество поколений, приспособленность лучшей хромосомы не улучшается в течение определенного количества итераций.

Алгоритм для поиска оптимального размещения продуктов на складе имеет следующую последовательность:

1. На основе полученной матрицы смежности, описывающей расположение склада (графа), вычисляется матрица достижимости с помощью выполнения алгоритма Дейкстры для каждой вершины графа. Эта матрица хранит кратчайшие расстояния между всеми парами позиций на складе и используется для быстрой оценки целевой функции на следующих шагах.

2. Создается начальная популяция(набор хромосом), состоящая из хромосом, где – размер популяции для внешнего ГА. Каждая хромосома кодирует возможное размещение товаров на складе. Хромосома представляется в виде массива, где позиция элемента соответствует позиции товара на складе.

3. Для каждой хромосомы (размещения) в популяции вычисляется целевая функция, определяющая приспособленность решения. Целевая функция представляет собой суммарную длину (или время) оптимальных маршрутов комплектации всех заказов для данного размещения товаров. Для каждого заказа вычисление целевой функции включает в себя:

а. Определение позиций товаров из заказа на складе, используя текущее размещение товаров (хромосому).

б. Поиск близкого к оптимальному маршрута комплектации заказа. Для поиска маршрута используется генетический алгоритм для TSP (внутренний ГА).

i. Создается начальная популяция, состоящая из K хромосом, где каждая хромосома кодирует последовательность посещения позиций товаров для данного заказа

ii. Для каждой хромосомы в популяции вычисляется длина маршрута. Путем суммирования расстояний между последовательными позициями, используя матрицу достижимости

iii. Отбираются хромосомы с лучшей длиной для участия в кроссовере.

iv. Кроссовер HGreX мутации для создания новых маршрутов

в. Вычисление длины (или времени) маршрута на основе матрицы расстояний

4. Суммирование длин (или времен) маршрутов для всех заказов.

5. Селекция размещений. Выбираются хромосомы (размещения) из популяции для участия в кроссовере. Хромосомы с более высокой приспособленностью (меньшим значением целевой функции) имеют больше шансов быть выбранными. Используются методы селекции, такие как рулетка, турнирная селекция или ранговая селекция.

6. Кроссовер размещений. Выбранные хромосомы (родители) скрещиваются для создания новых хромосом (потомков). Используются операторы кроссовера AEX и HGreX.

7. Мутация размещений. В хромосомы потомков вносятся небольшие случайные изменения. Используются операторы мутации, такие как перестановка, инверсия или вставка.

8. Замена популяции. Новые хромосомы заменяют часть старых хромосом в популяции. Используются стратегии замены, такие как замена худших, элитизм или турнирная замена.

9. Алгоритм останавливается, когда выполнено определенное условие останова, например, достигнуто максимальное количество поколений (), приспособленность лучшей хромосомы не улучшается в течение определенного количества поколений.

### 1.2.1. Доказательство корректности алгоритма.

Нельзя сказать, что генетический алгоритм находит оптимальный вариант, но он сходится к оптимальному. Работоспособность ГА доказывает теорема о схемах.

### 1.2.2. Трудоёмкость алгоритма.

Для получения матрицы достижимости с расстояниями запускаем алгоритм Дейкстры для каждой вершины и для разряженного графа получаем сложность .

Инициализация популяции внешнего ГА имеет сложность , где – количество хромосом в популяции, – количество товаров на складе.

Дальше для каждого заказа, сложность поиска близкого к оптимальному маршрута, в нашем случае генетического алгоритма для TSP , где

- количество поколений для внутреннего ГА (TSP).

- размер популяции для внутреннего ГА.

- среднее количество товаров в заказе (длина хромосомы для TSP).

- сложность оператора кроссовера для TSP.

- сложность оператора мутации для TSP.

Она сложилась из вычисления длины маршрута, операции кроссовера и мутации для внутреннего ГА. Так как обычно операторы и не превышают длину хромосомы, можно записать сложность . Учитывая количество заказов М имеем сложность для одной хромосомы .

### Селекция требует сравнения приспособленности хромосом и выбора наиболее подходящих. Сложность зависит от метода селекции:

### Рулетка: . Нужно вычислить вероятности для каждой хромосомы и выбрать раз.

### Турнирная селекция: , где - размер турнира (обычно небольшое число, например, 2 или 3). Поэтому можно считать .

### Ранговая селекция: (для сортировки) + (для выбора).

### В большинстве случаев можно считать .

Скрещивание выбранных хромосом для создания потомков имеет сложность как для оператора AEX так и для HGreX.

Сложность мутации также зависит от оператора и в худшем случае может быть , если мы выполняем мутацию для каждой хромосомы и проходимся по всем элементам.

Замена популяции также зависит от способа:

* Замена худших: нужно найти худшую хромосому . Может потребоваться , если нужно сортировать всю популяцию.
* Элитизм: просто копируем лучшие хромосомы .
* Турнирная замена: , где - размер турнира. Поэтому можно считать .

В большинстве случаев можно считать

Общая сложность получается

Преобразуем и получаем

### 1.2.3. Блок-схема алгоритма

# 2. Конструкторская часть

## 2.1. Выбор языка программирования

Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст.

Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст.

## 2.2. Описание основной функции

Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст.

Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст.

## 2.3. Использованные технологии

Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст.

Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст.

## 2.4. База данных

Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст.

Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст.

# Заключение

Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст.

Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст.

# Cписок использованных источников

1. Кордос, М., Боричко, Я., Блахник, М., Голак, С. Optimization of Warehouse Operations with Genetic Algorithms // Applied Sciences. – 2020. – Т. 10, № 14. – С. 4817.
2. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст.

# ПРИЛОЖЕНИЕ А

Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст.

Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст.

# ПРИЛОЖЕНИЕ Б

Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст.

Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст. Текст.