МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Тольяттинский государственный университет»

Кафедра «Прикладная математика и информатика»
(наименование)

01.03.02 Прикладная математика и информатика
(код и наименование направления подготовки / специальности)

Компьютерные технологии и математическое моделирование
(направленность (профиль) / специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему Исследование и реализация алгоритма для решения задачи оптимизации выпуска готовой продукции

Обучающийся Д.А.Спирин
(И.О. Фамилия) (личная подпись)

Руководитель д.т.н., доцент, С.В. Мкртычев
(ученая степень(при наличии), ученое звание(при наличии), Инициалы Фамилия)

Консультант к.п.н., доцент, А.В.Егорова
(ученая степень(при наличии), ученое звание(при наличии), Инициалы Фамилия)

Тольятти 2024

Аннотация

Тема выпускной квалификационной работы: «Исследование и реализация алгоритма для решения задачи оптимизации выпуска готовой продукции».

Выпускная квалификационная работа посвящена решению задачи оптимизации выпуска готовой продукции.
Для обеспечения эффективного распределения ресурсов и получении максимальной прибыли необходимо оптимизировать процессы производства.

Объектом исследования бакалаврской работы является задача оптимизации выпуска готовой продукции.

Предметом исследования является алгоритм для решения задачи оптимизации выпуска готовой продукции.

Цель бакалаврской работы – исследование и реализация алгоритма для решения задачи оптимизации выпуска готовой продукции.

Методы исследования – методы и алгоритмы решения задачи выпуска готовой продукции, технологии реализации алгоритмов на языках высокого уровня.

Практическая значимость бакалаврской работы заключается в разработке и тестировании программы, реализующей эффективные алгоритмы решения задачи выпуска готовой продукции.

Результаты бакалаврской работы представляют научно-практический интерес и могут быть рекомендованы производственным предприятиям для увеличения прибыли и оптимизации производства на основе методов оптимизации выпуска готовой продукции.

Бакалаврская работа состоит из 43 страниц текста, 3 таблиц, 9 рисунков и 25 источников.

Abstract

Theme of the graduate qualification work: "Research and implementation of the algorithm to solve the problem of optimization of the output of finished products".

The graduate qualification work is devoted to solving the problem of optimizing the output of finished products.

To ensure the effective allocation of resources and obtain maximum profit it is necessary to optimize production processes.

The object of research bachelor's work is the problem of optimizing the output of finished products.

The purpose of the bachelor's work - research and implementation of algorithms to solve the problem of optimizing the output of finished products.

The subject of the work are algorithms to solve the problem of optimizing the output of finished products.

Objectives of the bachelor's work - analysis of requirements and constraints, implementation of the algorithm, testing and performance evaluation.

Methods of research - methods and algorithms for solving the problem of output of finished products, technologies for implementing algorithms in high-level languages.

Practical significance of the bachelor's work is the development and testing of a program that implements effective algorithms for solving the problem of finished product release.

The results of the bachelor's work are of scientific and practical interest and can be recommended to manufacturing enterprises to increase profits and optimize production on the basis of methods for optimizing the output of finished products.

The graduate qualification work consists of 43 pages of text, 3 tables, 9 figures and 25 sources.

Оглавление

Введение

Исследование и реализация алгоритма для решения задачи оптимизации выпуска готовой продукции является актуальной задачей в области производства и бизнеса. В современном мире компании стремятся максимизировать свою производительность, минимизировать издержки и увеличить эффективность процессов производства. Оптимизация выпуска готовой продукции играет важную роль в достижении этих целей.

Задача оптимизации выпуска готовой продукции заключается в поиске оптимального производства, которое позволит достичь максимальной производительности при минимальных издержках и ресурсах. Для этого необходимо учитывать различные ограничения и условия, такие как ограничения по производственным мощностям, поставкам сырья, технологическим процессам и временным рамкам.

Исследование и разработка алгоритмов для решения данной задачи имеет большое значение, поскольку позволяет компаниям повысить свою конкурентоспособность, улучшить качество продукции, снизить издержки и повысить общую эффективность. Разработка эффективного алгоритма для оптимизации выпуска готовой продукции требует глубоких знаний в области математики, оптимизации, технологических процессов и производства.

Объектом исследования бакалаврской работы является задача оптимизации выпуска готовой продукции.

Предметом исследования является алгоритм для решения задачи оптимизации выпуска готовой продукции.

Цель бакалаврской работы – исследование и реализация алгоритма для решения задачи оптимизации выпуска готовой продукции.

Для достижения поставленной цели необходимо решить следующие задачи:

– выполнить постановку задачи исследования и проанализировать методы оптимизации выпуска готовой продукции;

– проанализировать алгоритмы для решения задачи оптимизации выпуска готовой продукции;
– выполнить реализацию и тестирование алгоритма для решения задачи оптимизации выпуска готовой продукции.

Методы исследования – методы и алгоритмы решения задачи выпуска готовой продукции, технологии реализации алгоритмов на языках высокого уровня.

Практическая значимость бакалаврской работы заключается в разработке и тестировании программы, которая может эффективно реализовывать алгоритмы решения задачи выпуска готовой продукции.

Интеграция разработанного алгоритма в производственный процесс позволит компаниям автоматизировать и оптимизировать процессы планирования и управления производством, что приведёт к улучшению качества продукции, снижению издержек и повышению общей эффективности производства. Таким образом, исследование и реализация алгоритма для решения задачи оптимизации выпуска готовой продукции имеет большое практическое значение для компаний и организаций, занятых в производственной сфере.

Данная работа состоит из введения, трех глав, заключения и списка используемой литературы.

Первая главе работы посвящена постановке задачи исследования и анализу методов оптимизации выпуска готовой продукции.

Вторая глава работы посвящена обзору и анализу алгоритмов для решения задачи оптимизации выпуска готовой продукции.

В третьей главе рассматривается программная реализация и тестирование алгоритма для решения задачи оптимизации выпуска готовой продукции.

В заключении описываются результаты выполнения выпускной квалификационной работы.

Глава 1 Постановка задачи исследования и анализ методов оптимизации выпуска готовой продукции

Постановка задачи:

- описание задач выпуска готовой продукции, выбор и анализ примера;
- изучение существующих методов решения задач выпуска готовой продукции;
- анализ теоретических основ и алгоритмов решения задач выпуска готовой продукции;
- реализация алгоритма для решения задач выпуска готовой продукции;
- оценка эффективности алгоритма на примере задачи;
- тестирование программной реализации с использованием разных начальных данных;
- сравнительный анализ результатов работы программной реализации алгоритма.

Оптимизация производственных процессов является ключевым аспектом управления, который направлен на улучшение эффективности, снижение издержек и увеличение прибыльности предприятия [4].

В контексте оптимизации производственных процессов, понимание и управление ограничениями и параметрами системы являются критически важными аспектами. Они задают рамки, в которых возможно функционирование и оптимизация процессов, и оказывают значительное влияние на выбор стратегий и методов оптимизации [18].

Рассмотрим подробнее, какие бывают ограничения и параметры, и как они влияют на процесс оптимизации.

Рассмотрим определение ограничений.

Ограничения – это условия или переменные, которые ограничивают возможности производства или процесса [8]. Они могут быть как внешними, так и внутренними:

- физические ограничения: включают в себя мощность оборудования, производственные площади, доступность ресурсов и технологические возможности. Например, максимальная производительность станка или максимальная загрузка производственной линии [16];
- ресурсные ограничения: связаны с доступностью сырья, материалов, финансовых и человеческих ресурсов [23]. Например, ограниченный бюджет на закупку материалов или ограниченное количество квалифицированных работников;
- операционные ограничения: включают в себя рабочие графики, соблюдение стандартов качества и безопасности, а также логистические ограничения;

Процесс производства обязательно будет учитывать все поставленные ограничения в решении задачи оптимизации выпуска готовой продукции.

Задача: малое предприятие, специализирующееся на производстве и продаже крафтовых продуктов, сталкивается с задачей оптимизации своего производственного плана. Предприятие производит два типа продуктов: ручные часы и кожаные сумки. Из-за ограниченного бюджета и необходимости максимизации прибыли, компания хочет определить оптимальное количество каждого продукта для производства, учитывая различные затраты, цены продажи и минимально необходимые объемы производства.

1.1 Процесс и основная задача оптимизации

Типичный процесс оптимизации инженерного проектирования показан на рисунке 1. Задача конструктора включает разработку технического задания, в котором описываются характеристики, константы, цели и ограничения проекта. Конструктор несет ответственность за формулировку задач и оценку

преимуществ возможных проектов с количественной точки зрения. Он также обычно предоставляет базовый проект или начальную точку проектирования для алгоритма оптимизации [22].

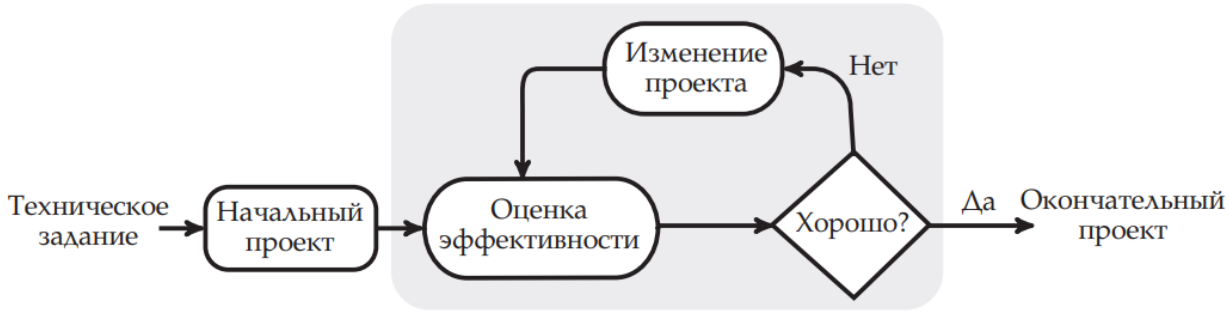

Рисунок 1 – Процесс оптимизации проекта

Алгоритм оптимизации используется для постепенного улучшения проекта до тех пор, пока проект больше не может быть улучшен или пока не будет затрачено запланированное время либо превышена предельно допустимая стоимость. Конструктор несет ответственность за анализ результатов процесса оптимизации, чтобы обеспечить его пригодность для конечного применения. Неправильные спецификации в постановке задачи, плохой начальный проект и неправильно реализованные или неподходящие алгоритмы оптимизации могут привести к неоптимальным или опасным проектам.

Есть несколько преимуществ оптимизации подхода к проектированию. В первую очередь, процесс оптимизации предоставляет методичную и логическую методику проектирования [21]. При ее адекватном применении алгоритмы оптимизации способны снизить риск человеческих ошибок в процессе проектирования. Полагаться на интуицию в инженерии может привести к ошибкам; гораздо эффективнее использовать оптимизацию данных. Оптимизация также может ускорить проектирование, особенно когда процедуру можно разработать один раз и повторно использовать для

различных задач.

Традиционные инженерные методы часто визуализируются и обосновываются людьми в двух или трех измерениях. В то же время современные методы оптимизации могут применяться к задачам с миллионами переменных и ограничений [19].

Основная задача оптимизации формулируется следующим образом (1):

$$\min f(x) \text{ при условии, что } x \in X. \quad (1)$$

Здесь x – расчетная точка (design point). Расчетная точка может быть представлена как вектор значений, соответствующих различным расчетным переменным (design variables). Расчетная точка в n-мерном пространстве записывается как $[x_1, x_2, \dots, x_n]$, где i-я расчетная переменная обозначена x.

Элементы в этом векторе можно регулировать, чтобы минимизировать целевую функцию f. Любое значение x из всех точек в допустимом множестве X, которое минимизирует целевую функцию, называется решением или точкой минимума. Конкретное решение записывается как x*. Пример задачи одномерной оптимизации показан на рисунке 2.

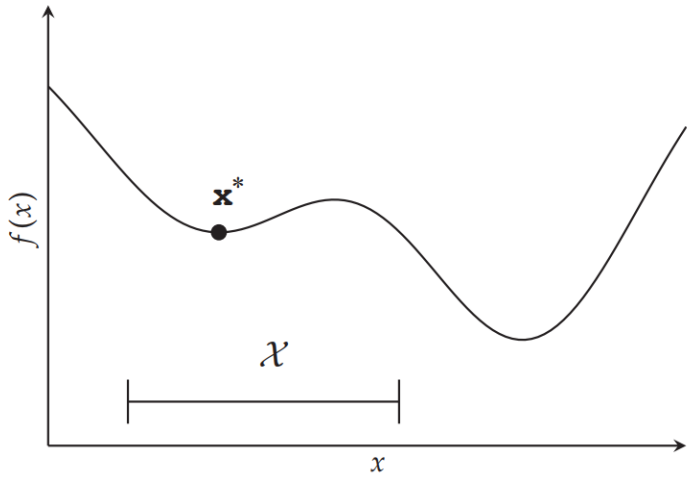

Рисунок 2 – Задача одномерной оптимизации

Эта формулировка является общей, т.е. любая задача оптимизации может быть переформулирована так (2):

$$\max -f(x) \text{ при условии, что } x \in X. \quad (2)$$

Задача в новой формулировке имеет тот же самый набор решений.

Моделирование инженерных задач в рамках этой математической формулировки может быть сложной задачей. То, как мы формулируем задачу оптимизации, может сделать процесс решения простым или сложным. Сосредоточимся на алгоритмических аспектах оптимизации, возникающих после того, как проблема была правильно сформулирована.

Поскольку в этой работе обсуждается множество различных алгоритмов оптимизации, можно задаться вопросом, какой алгоритм лучше. Как утверждает теорема об отсутствии бесплатного завтрака Вольперта и Макриди, нет причин предпочитать один алгоритм другому, если только мы не сделаем предположений о распределении вероятностей по пространству возможных целевых функций [24]. Если один алгоритм работает лучше, чем другой алгоритм для одного класса задач, то он будет работать хуже для другого класса задач. Чтобы многие алгоритмы оптимизации работали эффективно, в целевой функции должна быть некоторая регулярность, например липшиц-непрерывность или выпуклость. Обсуждая различные алгоритмы, мы опишем их предположения, обоснование их работы, а также их преимущества и недостатки.

1.2 Анализ методов оптимизации выпуска готовой продукции

Для решения задач оптимизации разберем основные методы:

− графический метод;

− симплекс-метод;

− метод Гомори.

Рассмотрим графический метод.

«Этот метод использует геометрическое изображение возможных решений и целевой функции задачи. В контексте задачи линейного программирования, каждое неравенство определяет на плоскости с координатами (x1, x2) отдельную полуплоскость [10]. Область, где эти полуплоскости пересекаются, образует область допустимых решений (ОДР), что означает, что любая точка внутри этой области удовлетворяет всем ограничениям задачи.

В зависимости от специфики задачи, область допустимых решений может принимать различные формы, такие как выпуклый многоугольник, неограниченная многоугольная область, луч, отрезок, отдельная точка или даже оказаться пустой, если ограничения задачи не совместимы друг с другом» [6]. Всегда стоит учитывать, что область допустимых решений представляет собой выпуклую геометрическую фигуру (рисунок 3).

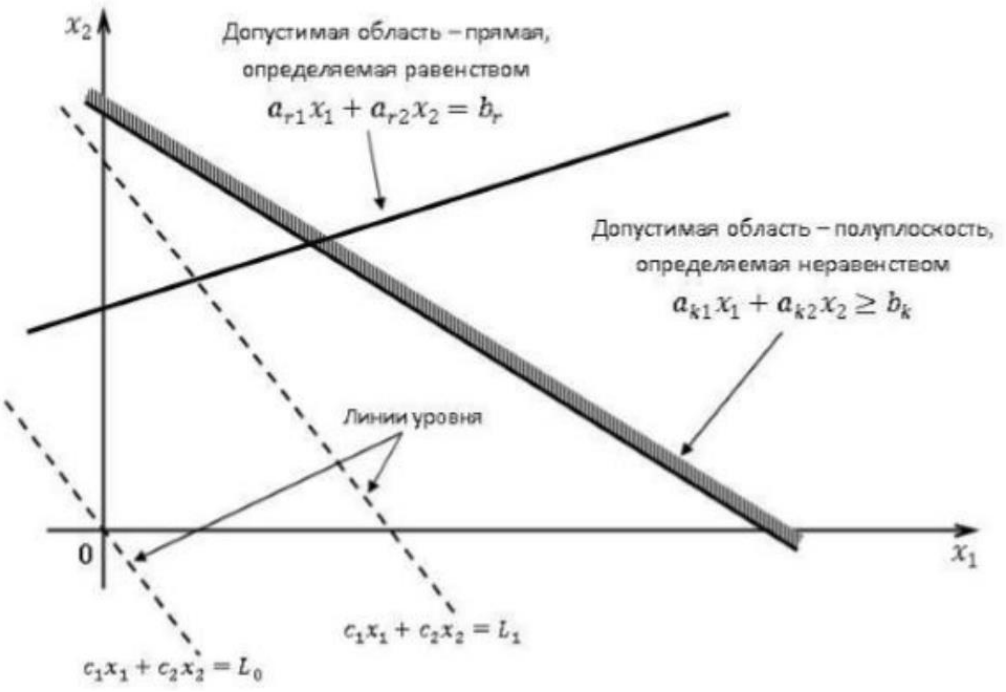

Рисунок 3 – Геометрическая интерпретация ограничений и целевой функции

«Из геометрической перспективы в задаче оптимизации осуществляется поиск такой вершины или группы вершин из области допустимых решений, где достигается наивысшее (или наименьшее) значение линии уровня, находящейся наиболее далеко (или близко) от остальных в направлении наибыстрейшего возрастания» [14].

Целевая функция при заданном значении L определяет на плоскости прямую $L = c_1 * x_1 + c_2 * x_2$. При изменении L формируется ряд параллельных прямых, которые называются линиями уровня.

Для определения экстремального значения целевой функции применяется вектор градиента L = C на плоскости X_1OX_2, указывающий направление наибыстрейших изменений целевой функции. Вектор градиента L = C выражается как (3):

$$\overrightarrow{grad}\, L = C = \frac{\partial L}{\partial x_1}\overrightarrow{e_1} + \frac{\partial L}{\partial x_2}\overrightarrow{e_2}, \quad (3)$$

где $\bar{e}_1$ и $\bar{e}_2$ являются единичными векторами вдоль осей OX_1 и OX_2 соответственно.

Таким образом, получим (4):

$$\vec{C} = \left(\frac{\partial L}{\partial x_1}, \frac{\partial L}{\partial x_2}\right). \quad (4)$$

Координатами вектора $\vec{C}$ соответствуют коэффициентам целевой функции L(x).

«Рассмотрим симплекс-метод.

Суть симплекс-метода заключается в том, что если задача оптимизации имеет решение, то экстремум функции f(x) достигается хотя бы в одной из вершин допустимого множества S задачи.

Так как разные базисные решения системы соответствуют выбору m переменных из общего числа n переменных x_j, количество возможных базисных решений (вершин) не превышает (5):

$$C_n^m = \frac{n!}{m! * (n-m)!}. \quad (5)$$

Следовательно, задачу можно решить путем перебора конечного числа вершин допустимого множества S, сравнивая значения целевой функции в каждой из них.

Симплекс-метод, разработанный Джорджем Данцигом, предполагает целенаправленный перебор вершин с последовательным уменьшением значения целевой функции f(x). В этом методе начало координат переносится в вершину допустимой области, и изучается изменение целевой функции вдоль линий, соединяющих эту вершину с соседними вершинами. Если какая-либо из линий приводит к улучшению, начало координат перемещается в вершину, к которой ведет эта линия, и процесс повторяется до достижения вершины, где функция f(x) не увеличивается (или не уменьшается) по любой из линий» [5].

Рассмотрим метод Гомори.

«Данный метод предназначен для решения целочисленных задач оптимизации, который включает построение дополнительных ограничений и использование модифицированного симплекс-метода» [20].

«В методе Гомори первый этап решения аналогичен обычному расчету по симплекс-методу. Если среди значений переменных в оптимальном плане есть дробные, то формулируется дополнительное ограничение, которое отсекает дробную часть решения, сохраняя при этом все остальные условия. Это ограничение добавляется к исходным, и снова применяется симплекс-метод, позволяя достичь оптимального целочисленного решения за конечное число шагов» [1].

Для удобства сравнения и оценки характеристик методов решения задачи оптимизации выпуска готовой продукции используем таблицу 1.

Таблица 1 – Преимущества и недостатки основных методов решения задач оптимизации выпуска готовой продукции

Методы решения задач оптимизации	Преимущества	Недостатки
Симплекс-метод	Эффективен для решения большинства задач оптимизации любой размерности. Дает точное решение. Существует множество улучшений и вариаций.	Может быть вычислительно затратным для очень больших задач оптимизации.
Графический метод	Позволяет визуально представить решение задачи, что делает его интуитивно понятным и легким для понимания на базовом уровне. Для задач с двумя переменными прост в реализации и не требует сложных вычислений.	Не подходит для задач с большим количеством переменных. Возможны ошибки в точности решения при ручном определении оптимальной точки на графике.
Метод Гомори	Позволяет найти точное целочисленное решение.	Сложная реализация. Вычислительно затратный метод, так как требует многократного решения с добавлением новых ограничений на каждом шаге.

Выводы по главе 1

В первой главе ВКР были рассмотрены основные методы решения задач оптимизации.

Основное внимание уделено задаче о малом предприятие, которое сталкивается с задачей оптимизации своего производственного плана.

Глава 2 Обзор и анализ алгоритмов для решения задачи оптимизации выпуска готовой продукции

Алгоритм графического метода:

Алгоритм графического метода включает несколько шагов и предназначен для решения задач оптимизации (6, 7)

$$f(x) = c_0 + \sum_{j=1}^{n} c_j x_j \to extr, \tag{6}$$

$$\sum_{j=1}^{n} a_{ij} x_j \{\leq, =, \geq\} b_j, i = 1,2, \dots, m, \tag{7}$$

Шаг 1: если в задаче присутствуют только две переменные, то они обозначаются как $l_1 = 1, l_2 = 2$ и используются в графике для дальнейшей визуализации, следовательно переходим к шагу 3. Если переменных больше двух, переходим к шагу 2.

Шаг 2: преобразуем исходную задачу к форме с двумя переменными.

Это достигается путем выделения в ограничениях базиса (8):

$$x_{k_i} + \sum_{\substack{j=1 \\ j \neq k_1 \dots k_m}}^{n} \overrightarrow{a_{ij}} x_j = \overrightarrow{b_j}, x_{k_j} \geq 0, i = 1, 2, \dots, m, \tag{8}$$

где k_j – индексы базисных переменных. Перепишем (8) в виде (9):

$$x_{k_i} = \overrightarrow{b_i} - \sum_{\substack{j=1 \\ j \neq k_1 \dots k_m}}^{n} \overrightarrow{a_{ij}} x_j, i = 1, 2, \dots, m. \tag{9}$$

Учитывая условия, что количество независимых переменных задачи равно двум, то в правой части выражений (9) останутся только две неизвестные, то есть (10):

$$x_{k_i} = \overrightarrow{b_i} - \overrightarrow{a_{il_1}} x_{l_1} - \overrightarrow{a_{il_2}} x_{l_2}, i = 1, 2, \dots, m. \quad (10)$$

Здесь l_1 и l_2 индексы независимых переменных ($l_1 \neq l_2, l_1 \neq k_i, l_2 \neq k_i$).

С учетом ограничений на знак базисных переменных $x_{k_i} \geq 0$, получим (11):

$$\vec{b}_i - \vec{a}_{il_1} x_{l_1} - \vec{a}_{il_2} x_{l_2} \geq 0 \Rightarrow \vec{a}_{il_1} x_{l_1} + \vec{a}_{il_2} x_{l_2} \leq \vec{b}_i, i = 1, 2, \dots, m. \quad (11)$$

Также избавляемся от базисных переменных в целевой функции. Таким образом останутся две переменные – x_{l_1} и x_{l_2}.

Шаг 3: в декартовой системе координат пространства R^2 строим координатную сетку. Переменные x_{l_1} и x_{l_2} откладываем по осям сетки.

Шаг 4: определяем и изображаем область допустимых значений. Для каждого ограничения рисуем соответствующие геометрические фигуры.

Если ограничение представлено уравнением, то его геометрическим изображением будет прямая. Если ограничение представлено неравенством, то изображаем полуплоскость, ограниченную прямой, соответствующей уравнению.

Чтобы определить, какая именно полуплоскость соответствует ограничению, проверяем, удовлетворяет ли точка (0, 0) или другая контрольная точка данному ограничению.

Область допустимых решений (ОДР) задачи образует выпуклый многогранник (например, треугольник или четырехугольник), который является пересечением всех областей, определяемых ограничениями.

В случае несовместности системы ограничений, область допустимых

решений будет пуста, что указывает на отсутствие решения задачи оптимизации, и алгоритм завершает свою работу.

Если все штриховки в образовавшемся четырехугольнике направлены внутрь, то ОДР существует, и задача оптимизации является согласованной (рисунок 4).

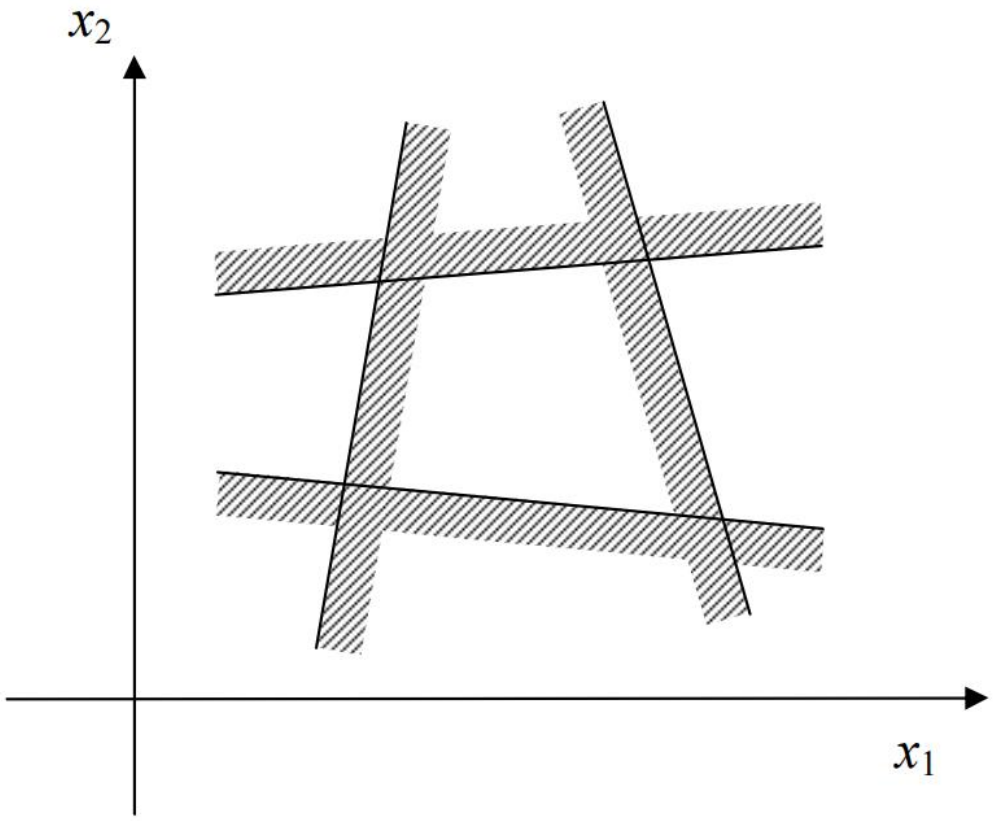

Рисунок 4 – Область допустимых значений – пустое множество

Шаг 5: определяем линию уровня целевой функции $f(x) = const$ так, чтобы она проходила через область допустимых решений [3].

Учитывая, что целевая функция имеет линейный характер, точки, соответствующие этой функции, располагаются на прямой [5]. При изменении значения const прямая испытывает параллельный сдвиг.

Для определения точки оптимального решения, эту линию уровня начинаем смещать параллельно в направлении, зависящем от типа задачи: в

сторону градиента, если задача заключается в поиске максимального значения, или в сторону антиградиента, если необходимо найти минимальное значение.

Процесс смещения продолжается до тех пор, пока линия уровня не достигнет точки, где она последний раз касается границы области допустимых решений [5].

Эта крайняя точка касания и представляет собой оптимальное решение задачи оптимизации. Значение const, при котором достигается эта точка, и будет оптимальным значением целевой функции.

Этот метод позволяет наглядно и точно определить, как изменение параметров влияет на результат, и выбрать наилучший возможный исход в рамках заданных ограничений.

Рассмотрим следующие случаи.

Случай 1. Когда происходит параллельный перенос, линия заданная уравнением $f(x) = const$ совпадает с одной из границ многогранника, содержащего возможные решения.

Это приводит к тому, что существует неограниченное количество решений.

На рисунке 5 видно, что каждая точка на отрезке AE представляет собой оптимальное решение, что говорит о бесконечном количестве возможных оптимумов в данной задаче.

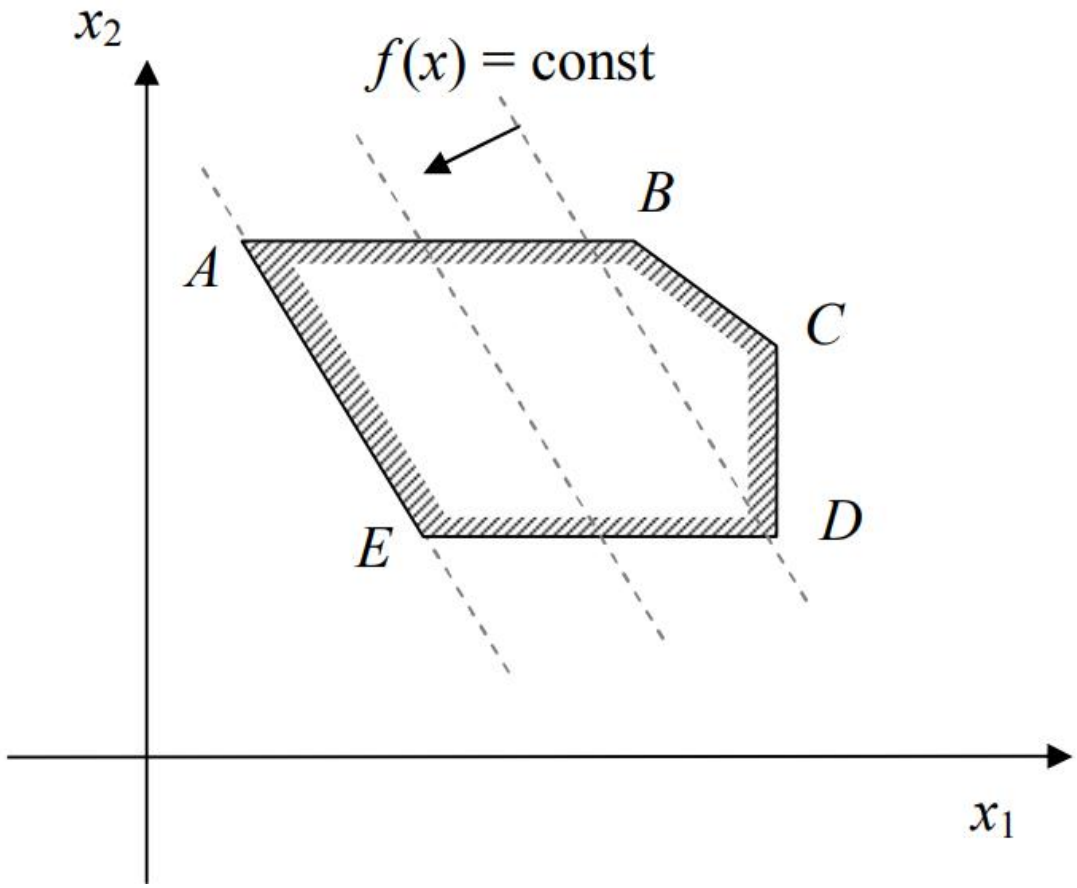

Рисунок 5 – Задача, имеющая бесконечное множество решений

Случай 2. Область допустимых решений (ОДР) не имеет верхней границы, что можно сравнить с формой "стакана", как показано на рисунке 6. Если направление антиградиента указывает в сторону "дна стакана" (вниз), то задача будет иметь решение, стремящееся к минимальному значению. В случае, когда антиградиент направлен в противоположную сторону, решение будет стремиться к максимальному значению.

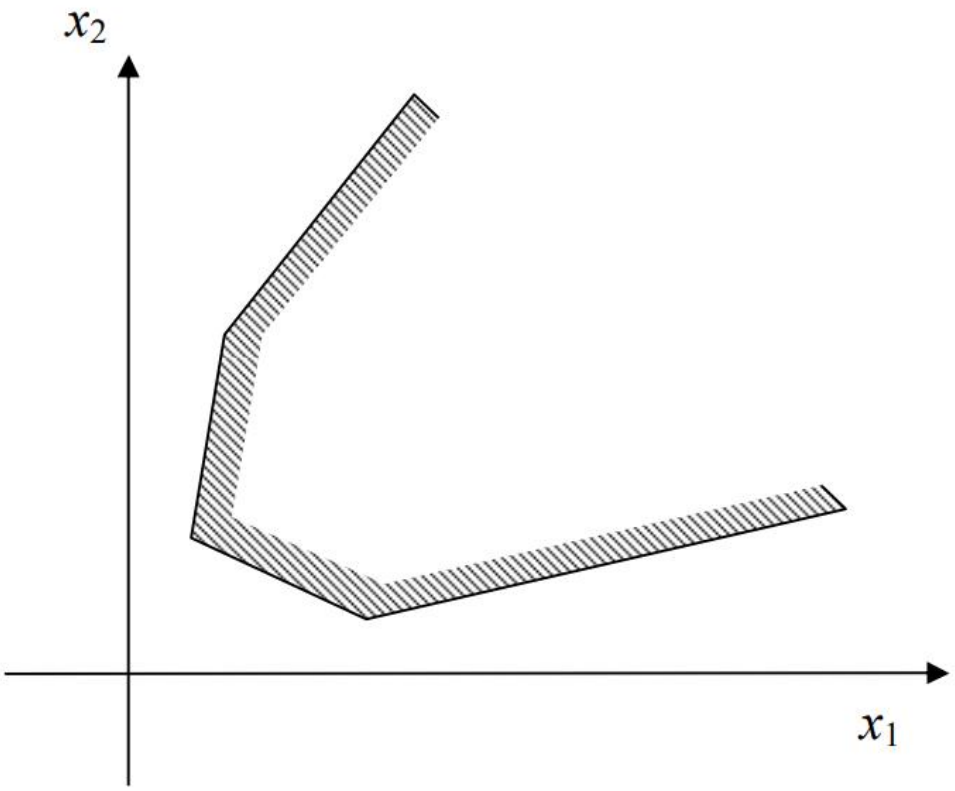

Рисунок 6 – Область допустимых решений задачи не ограничена сверху

Случай 3. Область допустимых решений также не ограничена, но в этот раз снизу, что напоминает "перевернутый стакан" (рисунок 7). По аналогии с предыдущим случаем, если направление антиградиента смотрит в сторону "дна стакана" (вверх), то решение будет стремиться к минимуму. Если направление антиградиента противоположное, задача будет иметь решение, стремящееся к достижению максимального значения.

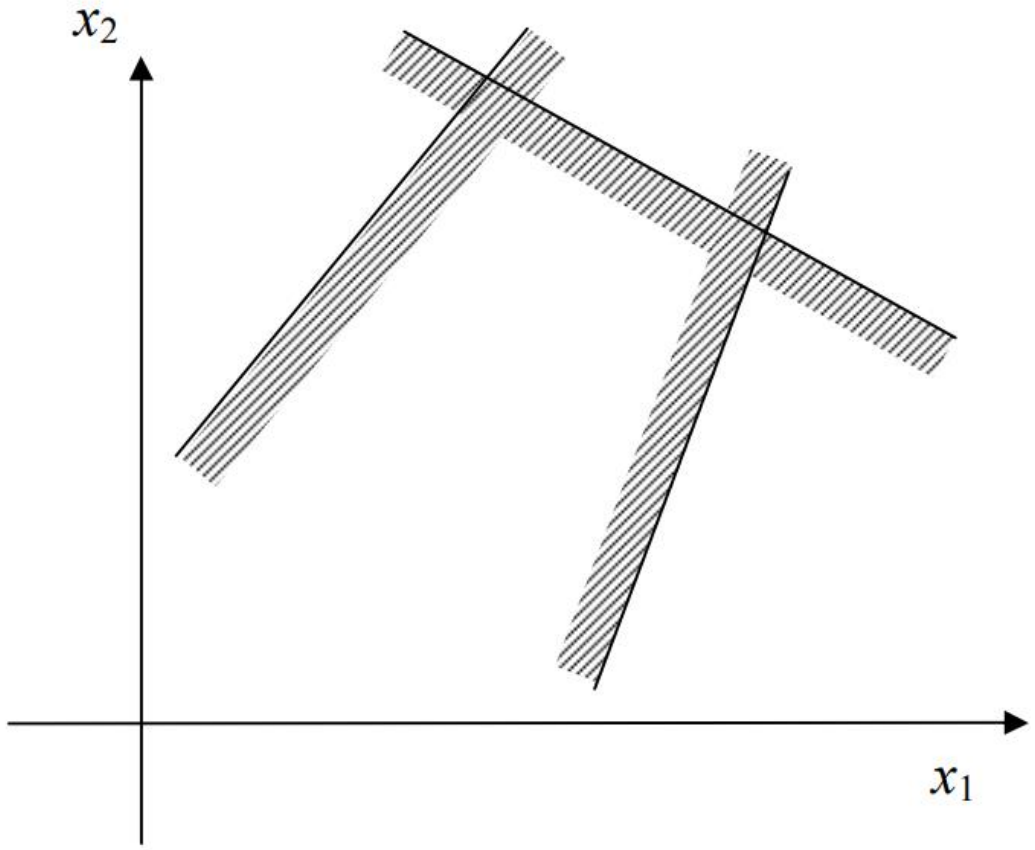

Рисунок 7 – Область допустимых решений задачи не ограничена снизу

Шаг 6: в процессе решения задачи могут быть выделены определенные базисные переменные. Для того чтобы определить их конкретные значения, необходимо обратиться к уравнению (10).

Алгоритм симплекс-метода:

Шаг 1: выделить в задаче допустимый базис.

Шаг 2: задать $k = 0, a_{ij}^0 = \bar{a}_{ij}, b_i^0 = \bar{b}_i, p_0^0 = p_0, p_j^0 = p_j,\ i = 1, 2, \dots, m,$ $j = 1, 2, \dots, n.$

Шаг 3: на данном шаге решения для упрощения будем полагать, что базисными являются переменные $x_1, \dots, x_m,$ а независимыми – переменные $x_{m+1}, \dots, x_n$. Таким образом, имеем задачу оптимизации (12-14):

$$f(x) = p_0^k + \sum_{j=m+1}^{n} p_j^k \, x_j, \tag{12}$$

$$x_i = b_i^k - \sum_{j=m+1}^{n} a_{ij}^k \, x_j, i = 1, 2, \dots, m, \tag{13}$$

$$x_j \geq 0, j = 1, 2, \dots, n. \tag{14}$$

Текущая угловая точка (15):

$$x^k = \left(b_1^k, b_2^k, \dots, b_m^k, 0, 0, \dots, 0\right)^T. \tag{15}$$

Рассмотрим следующие случаи.

Случай 1. Если в равенстве (12) все коэффициенты p_j^k неотрицательны, где $j = m + 1, \dots, n$, то можно утверждать, что в угловой точке (13) достигается минимальное значение целевой функции $f(x)$ в рамках допустимого множества S для данной задачи оптимизации. Этот минимум будет равен p_0^k.

Случай 2. Если среди коэффициентов p_j^k, где $j = m + 1, \dots, n$, есть отрицательные значения (например, p_l^k), и для всех коэффициентов a_{il}^k в равенстве (13) выполняется условие $a_{il}^k \leq 0$ для $i = 1, 2, \dots, m$, то целевая функция $f(x)$ не имеет ограничения снизу на множестве S, и, следовательно, задача не имеет решения в этом случае.

Случай 3. Если хотя бы один из коэффициентов p_j^k, где $j = m + 1, \dots, n$ является отрицательным (например, $p_l^k < 0$), и при этом среди коэффициентов a_{il}^k в равенстве (13) присутствует хотя бы один положительный, это указывает на существование угловой точки x^{k+1} в множестве S, в которой значение целевой функции $f(x^{k+1})$ будет меньше, чем $f(x^k)$.

В случаях 1 и 2 процесс решения задачи оптимизации заканчивается. В случае 3 продолжается поиск оптимального решения, и необходимо

переходить к следующему шагу алгоритма, указанному как шаг 4.

Шаг 4: рассмотрим, что в уравнении (12) коэффициент p_l^k отрицателен для некоторого $l \neq 0$, и в уравнении (13) присутствуют положительные коэффициенты a_{il}^k. Для определения индекса строки t, используем правило минимального частного (16):

$$\frac{b_t^k}{a_{tl}^k} = min\left\{\frac{b_i^k}{a_{al}^k}\right\} \forall a_{il}^k > 0, \tag{16}$$

где минимум определяется среди всех индексов $i = 1, 2, \dots, m$, для которых $a_{il}^k > 0$. Выбранный элемент a_{tl} будет считаться разрешающим или опорным элементом.

Далее решаем систему ограничений (17):

$$\sum_{j=1}^{n} a_{ij} x_j = b_i; i = 1, 2, \dots, m\,. \tag{17}$$

В этой системе переменные $x_{m+1}, \dots, x_{l-1}, x_t, x_{l+1}, \dots, x_n$ считаются свободными, после чего происходит замена свободной переменной x_l на базисную переменную x_t. Таким образом, система уравнений преобразуется в следующий вид (18):

$$\begin{gathered} x_i + \sum_{j=m+1}^{n} \left(a_{ij}^k - a_{il}^k \frac{a_{tj}^k}{a_{tl}^k}\right) x_j - \frac{a_{il}^k}{a_{tl}^k} x_t = b_i^k - a_{il}^k \frac{b_t^k}{a_{tl}^k}, \\ j \neq l, i = 1, 2, \dots, m, i \neq t, \\ x_l + \sum_{j=m+1}^{n} \frac{a_{tj}^k}{a_{tl}^k} x_j + \frac{1}{a_{tl}^k} x_t = \frac{b_t^k}{a_{tl}^k}. \end{gathered} \tag{18}$$

Зависимость целевой функции от новых свободных переменных принимает вид (19):

$$f(x) = \sum_{j=m+1}^{n} \left(p_j^k - p_l^k \frac{a_{tj}^k}{a_{tl}^k} \right) x_j - \frac{p_l^k}{a_{tl}^k} x_t + = p_0^k + p_l^k + p_l^k \frac{b_t^k}{a_{tl}^k}. \qquad (19)$$

Новое базисное решение x^{k+1} можно определить, приравняв к нулю свободные переменные x_j для $j = m+1, \dots, n, j \neq l,$ и x_t, и вычислив при этом условии значения базисных переменных из (16). Это базисное решение x^{k+1} является допустимым, то есть угловой точкой множества ОДР S, и при этом гарантируется выполнение неравенства $f(x^{k+1}) < f(x^k)$.

Шаг 5: задать $k = k + 1$ и вернуться к шагу 3 для продолжения процесса оптимизации.

Рассмотрим алгоритм метода Гомори.

Шаг 1: используя симплекс-метод, находим оптимальное решение x^* задачи оптимизации без учета условия целочисленности переменных. Если все переменные в x^* удовлетворяют целочисленности и условиям задачи, то задача считается решенной. Если же среди элементов b_i^k последнего столбца симплекс таблицы присутствуют нецелые значения, процесс продолжается.

Шаг 2: из всех нецелых значений b_i^k выбирается одно, например b_z^k, с максимальной дробной частью. Формируется новое ограничение по z-й строке симплекс таблицы (20):

$$-\sum_j \{a_{zj}^k\} x_j \leq -\{b_z^k\}, \qquad (20)$$

где j обозначает индексы всех свободных переменных.

С использованием вспомогательной переменной $x_{n+1} \geq 0$ это ограничение преобразуется в равенство (21):

$$x_{n+1} - \sum_j \{a_{zj}^k\} x_j = -\{b_z^k\} \quad (21)$$

и добавляется в симплекс таблицу новой строкой с номером m +1 (тем самым количество ограничений задачи увеличивается до m +1, а количество переменных до n +1), причем (22):

$$a_{m+1,j}^k = -\{a_{zj}^k\}, b_{m+1}^k = -\{b_z^k\}. \quad (22)$$

Так как $b_{m+1}^k < 0$, то после дополнения строкой (19) симплекс таблица перестает соответствовать допустимому базисному решению задачи оптимизации, которую она описывает.

Шаг 3: для перехода к допустимому базисному решению производятся следующие операции:

Случай 1. Строка с отрицательным свободным членом b_{m+1}^k считается опорной.

Случай 2. Если все коэффициенты $a_{m+1,j}^k \geq 0$, то задача не имеет решения, в противном случае номер l разрешающего столбца находится из условия (23):

$$\frac{b_{m+1}^k}{a_{m+1,l}^k} = min\left\{\frac{b_{m+1}^k}{\left|a_{m+1,j}^k\right|}\right\} \forall a_{m+1,j}^k < 0. \quad (23)$$

Случай 3. Совершается преобразование симплекс таблицы с опорным элементом $a_{m+1,l}^k$. Если в новой симплекс таблице по-прежнему есть хотя бы один отрицательный свободный член, то описанная процедура повторяется необходимое число раз. Если в новой симплекс таблице все элементы $b_i^{k+1} \geq 0$, то допустимое базисное решение найдено.

Отметим, что выбор опорного элемента $a_{m+1,l}^{k}$ гарантирует неотрицательность коэффициентов b_i^{k+1} новой симплекс таблицы.

Поэтому найденное допустимое решение является и оптимальным.

Шаг 4: если найденное на шаге 3 решение задачи оптимизации не удовлетворяет условию целочисленности, то возвращаемся на шаг 2.

Шаг 5: проверяется, удовлетворяет ли найденное решение критериям оптимальности для задачи оптимизации.

Если да, решение задачи завершено.

В противном случае, процесс возвращается на шаг 1.

Выводы по главе 2

Метод Гомори позволяет последовательно и точно находить полностью целочисленное решение задачи оптимизации или определять, что решения не существует, в рамках конечного числа итераций.

Глава 3 Программная реализация и тестирование алгоритма для решения задачи оптимизации выпуска готовой продукции

3.1 Обоснование выбора инструментов и среды обработки

Выбор Python как инструмента для разработки и реализации алгоритмов оптимизации выпуска готовой продукции является хорошим решением по нескольким причинам. Python – это мощный, гибкий и легко доступный язык программирования, который обладает рядом преимуществ для обработки и анализа данных, моделирования, разработки алгоритмов и автоматизации процессов.

Python поддерживается обширным набором библиотек и модулей, которые значительно упрощают процесс разработки и внедрения алгоритмов оптимизации [15].

Python известен своим чистым и легко читаемым синтаксисом, что делает его особенно подходящим для научных и инженерных приложений, где важно, чтобы код был понятен и доступен для анализа и модификаций [12]. Это упрощает разработку, отладку и сопровождение программного обеспечения.

Python обладает одним из самых активных и поддерживаемых сообществ разработчиков. Это обеспечивает богатую экосистему, в которой легко найти ответы на возникающие вопросы, а также обширные ресурсы для обучения и развития.

Python может выполняться практически на любой операционной системе, включая Windows, macOS и Linux. Это делает его идеальным выбором для проектов, где предполагается развертывание систем на различных платформах.

Python хорошо интегрируется с другими языками программирования и технологиями, такими как C/C++ или Java, что позволяет эффективно использовать уже существующий код [2]. Кроме того, Python может легко

взаимодействовать с различными системами баз данных и веб-сервисами.

Благодаря высокому уровню абстракции и широкой поддержке библиотек, Python позволяет быстро прототипировать решения, что особенно ценно в условиях, когда требуется быстро тестировать и модифицировать алгоритмы в ответ на изменяющиеся требования производства.

Выбор Python для решения задач оптимизации выпуска готовой продукции предоставляет значительные преимущества в гибкости, скорости разработки и мощности выполнения. Эти качества делают его идеальным инструментом для исследователей, инженеров и разработчиков, стремящихся к созданию эффективных и инновационных решений в сфере производственной оптимизации.

3.2 Тестирование алгоритма

Грамотное тестирование – это залог успешной работы как крупных программных систем, так и небольших коробочных программных продуктов [13].

Тестирование программного продукта можно структурировать и проводить как эксперимент, где целью является не только проверка функциональности и производительности программы, но и изучение её поведения в различных условиях.

Организуем и опишем процесс тестирования в виде эксперимента.

Цель эксперимента: проверить работоспособность и устойчивость программного продукта в условиях, максимально приближенных к реальной эксплуатации, а также изучить его поведение при крайних значениях входных данных и экстремальных условиях работы.

Выборка: подготовка различных наборов данных, которые включают типичные, пограничные и экстремальные значения.

Эти данные будут использоваться для имитации реальных сценариев использования программы, а также для тестирования реакции программы на

нестандартные и неожиданные ситуации.

Процедура тестирования будет проводиться в два этапа:

– функциональное тестирование. Целью функционального тестирования является удостовериться, что каждая функция программного обеспечения работает корректно. Для этого разрабатываются и реализуются различные тестовые случаи, которые покрывают все возможные сценарии использования программы [17]. Каждый тест направлен на определение корректности выполнения функций, таких как ввод данных, выполнение расчетов, обработка ошибок и взаимодействие с пользовательским интерфейсом. Результаты тестирования должны подтвердить, что программа выполняет все заложенные функции без сбоев и ошибок;

– нагрузочное тестирование. Задача этого этапа – оценить поведение программы в условиях интенсивного использования при обработке значительных объемов данных. В ходе нагрузочного тестирования также проверяется, как программа справляется с пиковыми нагрузками и как быстро она может восстановиться после возможных сбоев или перегрузок [7]. Эти испытания помогают определить максимальные рабочие характеристики системы, выявить узкие места в архитектуре и подсистемах;

Контрольные точки: фиксация результатов на каждом этапе тестирования для последующего анализа и сравнения с ожидаемыми результатами.

Результаты каждого этапа тестирования будут собираться и анализироваться для определения соответствия программы техническим требованиям и её способности справляться с экстремальными условиями.

Особое внимание будет уделено случаям, где результаты не соответствуют ожиданиям, для выявления потенциальных проблем в коде и дизайне программы.

В случае обнаружения ошибок или недостатков будет проведен дополнительный анализ причин их возникновения, что поможет в доработке и улучшении программного продукта [9].

Подведение итогов эксперимента, включая оценку степени готовности программы к эксплуатации.

Такой подход к тестированию в форме эксперимента позволяет не только убедиться в качестве продукта, но и глубже понять его поведение и потенциальные проблемы, что важно для разработки надёжных и устойчивых программных решений [25].

Итоговые значения результатов тестирований представлены в таблицах 2 и 3.

Таблица 2 – Результаты функционального тестирования

Наименование модуля	Описание тестового случая	Ожидаемый результат	Тестовый случай пройден?
Ввод данных	Заполнение данных в отведенные места	Ввод данных	да
Вывод результата	Нажатие кнопки «симплекс-метод»	Вывод оптимального количества продуктов и максимальной прибыли	да
Вывод результата	Нажатие кнопки «графический метод»	Вывод графика производства оптимального количества продукции и максимальной прибылью	да

Таблица 3 – Результаты нагрузочного тестирования

Цена продажи А	Цена продажи В	Мин. количество А	Мин. количество В	Себестоимость А	Себестоимость В	Доступный бюджет	Ожидаемый результат	Результат выполнения программы	Сообщение об ошибке
124	121	105	110	107	101	99999	193645	193645	Нет
2437	3579	1070	1101	1587	1695	18965200	20101808	20101808	Нет
4587	6874	689	787	872	647	Ш	ошибка	ошибка	да

Результаты тестирования кода показали следующее:

– тестирование обработки ошибок ввода данных: программа успешно обрабатывает некорректные входные данные, такие как отрицательные числа или строки вместо чисел. Выводит информативные сообщения об ошибке и предотвращает дальнейшее выполнение программы до ввода корректных данных;

– тестирование обработки исключений: при имитации ситуаций, приводящих к возникновению исключений, программа корректно обрабатывает исключения и выводит соответствующие сообщения об ошибке. Она не завершается аварийно и сохраняет стабильность работы;

– тестирование производительности: при использовании тестовых данных большого объема, программа демонстрирует стабильную производительность без значительного увеличения времени выполнения [11]. Она эффективно обрабатывает данные и сохраняет производительность при увеличении объема данных;

– тестирование на стабильность: программа успешно проходит тест на стабильность, не проявляя признаков утечек памяти или других проблем с производительностью в течение длительного времени работы или многократного выполнения.

Код программы демонстрирует высокую степень устойчивости к различным видам ошибок и непредвиденным ситуациям.

Он успешно справляется с обработкой исключений, корректно

обрабатывает ошибочные входные данные и эффективно работает с различными объемами данных.

Программа стабильно функционирует в течение длительного времени и сохраняет работоспособность.

3.3 Работа программного обеспечения в конкретной задаче

Код программы представлен в листинге 1.

Листинг 1 – Код программы

```
import tkinter as tk
from tkinter import ttk
import numpy as np
import matplotlib.pyplot as plt
from threading import Thread
import itertools
def plot_graph(optimal_quantity1, optimal_quantity2, max_profit, dostbudj, seba, sebb, mina, minb):
    x = np.linspace(0, 450, 400)  # Число 450 - примерное значение для визуализации
    y = (dostbudj - seba * x) / sebb  # Уравнение бюджетного ограничения
    plt.figure(figsize=(8, 6))
    plt.plot(x, y, label='Бюджетное ограничение')
    plt.fill_between(x, np.maximum(minb, np.zeros_like(y)), y, where=(x >= mina) & (y >= minb), color='gray', alpha=0.1, label='Допустимая область')
    plt.axvline(x=mina, color='red', linestyle='--', label='Мин. кол-во часов')
    plt.axhline(y=minb, color='blue', linestyle='--', label='Мин. кол-во сумок')
    plt.scatter(optimal_quantity1, optimal_quantity2, color='green', label='Оптимальная точка: ({}, {})'.format(optimal_quantity1,
```

```
optimal_quantity2))
    plt.text(optimal_quantity1, optimal_quantity2, f'  Прибыль: {max_profit:.2f}',
verticalalignment='bottom')
    plt.xlim(0, 450)
    plt.ylim(0, 300)
    plt.xlabel('Количество часов (x)')
    plt.ylabel('Количество сумок (y)')
    plt.title('Графическое решение задачи оптимизации производства')
    plt.legend()
    plt.grid(True)
    plt.show()
def maximize_profit(graf):
    try:
        selling_price1 = float(selling_price1_entry.get())
        selling_price2 = float(selling_price2_entry.get())
        min_quantity1 = int(min_quantity1_entry.get())
        min_quantity2 = int(min_quantity2_entry.get())
        cost1 = float(cost1_entry.get())
        cost2 = float(cost2_entry.get())
        budget = float(budget_entry.get())
       # Расчеты для определения оптимальных количеств
        max_quantity1 = int(budget / cost1)
        max_quantity2 = int(budget / cost2)
        max_profit = 0
        optimal_quantities = (0, 0)
        for quantity1, quantity2 in itertools.product(range(min_quantity1,
max_quantity1 + 1), range(min_quantity2, max_quantity2 + 1)):
            total_cost = cost1 * quantity1 + cost2 * quantity2
            if total_cost <= budget:
                current_profit = (selling_price1 - cost1) * quantity1 + (selling_price2 -
```

```
cost2) * quantity2
                if current_profit > max_profit:
                    max_profit = current_profit
                    optimal_quantities = (quantity1, quantity2)
        optimal_quantity1, optimal_quantity2 = optimal_quantities
        result_label.config(text=f"Оптимальное количество продукта 1: {optimal_quantity1}\nОптимальное количество продукта 2: {optimal_quantity2}\nМаксимальная прибыль: {max_profit:.2f}")
        if graf == 1:
            # Показываем график в отдельном потоке
            thread = Thread(target=plot_graph, args=(optimal_quantity1, optimal_quantity2, max_profit, budget, cost1, cost2, min_quantity1, min_quantity2))
            thread.start()
    except ValueError:
        result_label.config(text="Пожалуйста, введите корректные числовые значения.")
# Инициализация главного окна
root = tk.Tk()
root.title("Оптимизация прибыли от продажи продуктов")
# Создание интерфейса для ввода данных
labels = ["Цена продажи продукта 1:", "Цена продажи продукта 2:", "Минимальное количество продукта 1:",
          "Минимальное количество продукта 2:", "Себестоимость продукта 1:", "Себестоимость продукта 2:",
          "Доступный бюджет:"]
entries = []
for i, label in enumerate(labels):
    ttk.Label(root, text=label).grid(column=0, row=i, sticky=tk.W, padx=10, pady=10)
```

```
    entry = ttk.Entry(root)
    entry.grid(column=1, row=i, sticky=tk.E, padx=10, pady=10)
    entries.append(entry)
selling_price1_entry, selling_price2_entry, min_quantity1_entry, min_quantity2_entry, cost1_entry, cost2_entry, budget_entry = entries
```

Рассмотрим работу программы.

Рассмотрим следующий сценарий.

Малое предприятие, специализирующееся на производстве и продаже крафтовых продуктов, сталкивается с задачей оптимизации своего производственного плана.

Предприятие производит два типа продуктов: ручные часы и кожаные сумки.

Из-за ограниченного бюджета и необходимости максимизации прибыли, компания хочет определить оптимальное количество каждого продукта для производства, учитывая различные затраты, цены продажи и минимально необходимые объемы производства.

Ниже представлены исходные данные.

Стоимость продажи часов – 1000 рублей.

Стоимость продажи сумки – 3000 рублей.

Себестоимость часов – 800 рублей.

Себестоимость сумки – 1500 рублей.

Минимальный объем производства часов – 52 штуки.

Минимальный объем производства сумок – 100 штук.

Доступный бюджет – 340097 рублей.

Вывод результата программы:

Оптимальное количество часов – 53 штуки.

Оптимальное количество сумок – 198 штук.

Максимальная прибыль – 307600 рублей.

Решение задачи используя графический метод, выполненной в

программе представлено на рисунке 8.

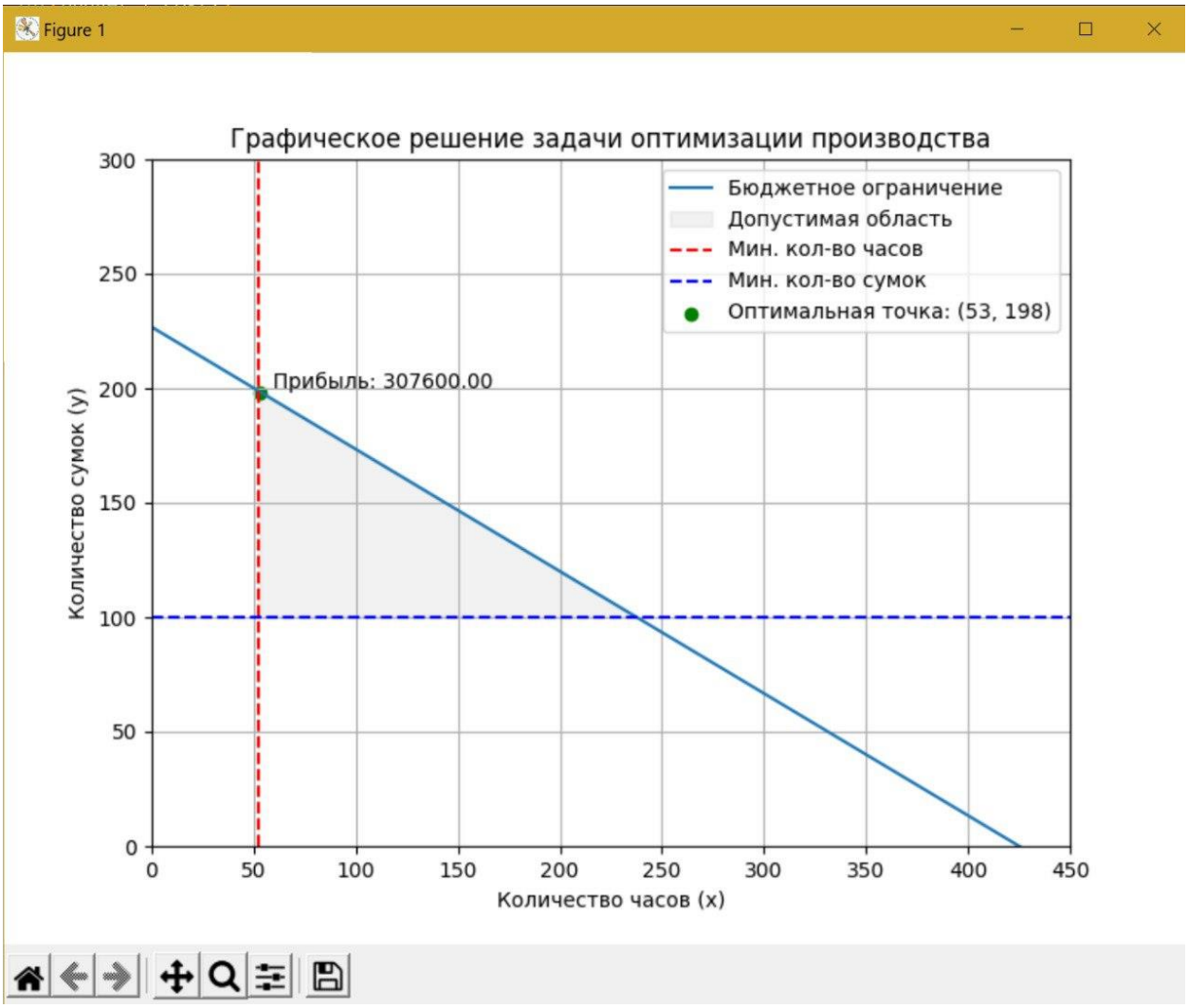

Рисунок 8 – Работа программы графическим методом

Решение задачи используя симплекс-метод, выполненной в программе представлено на рисунке 9.

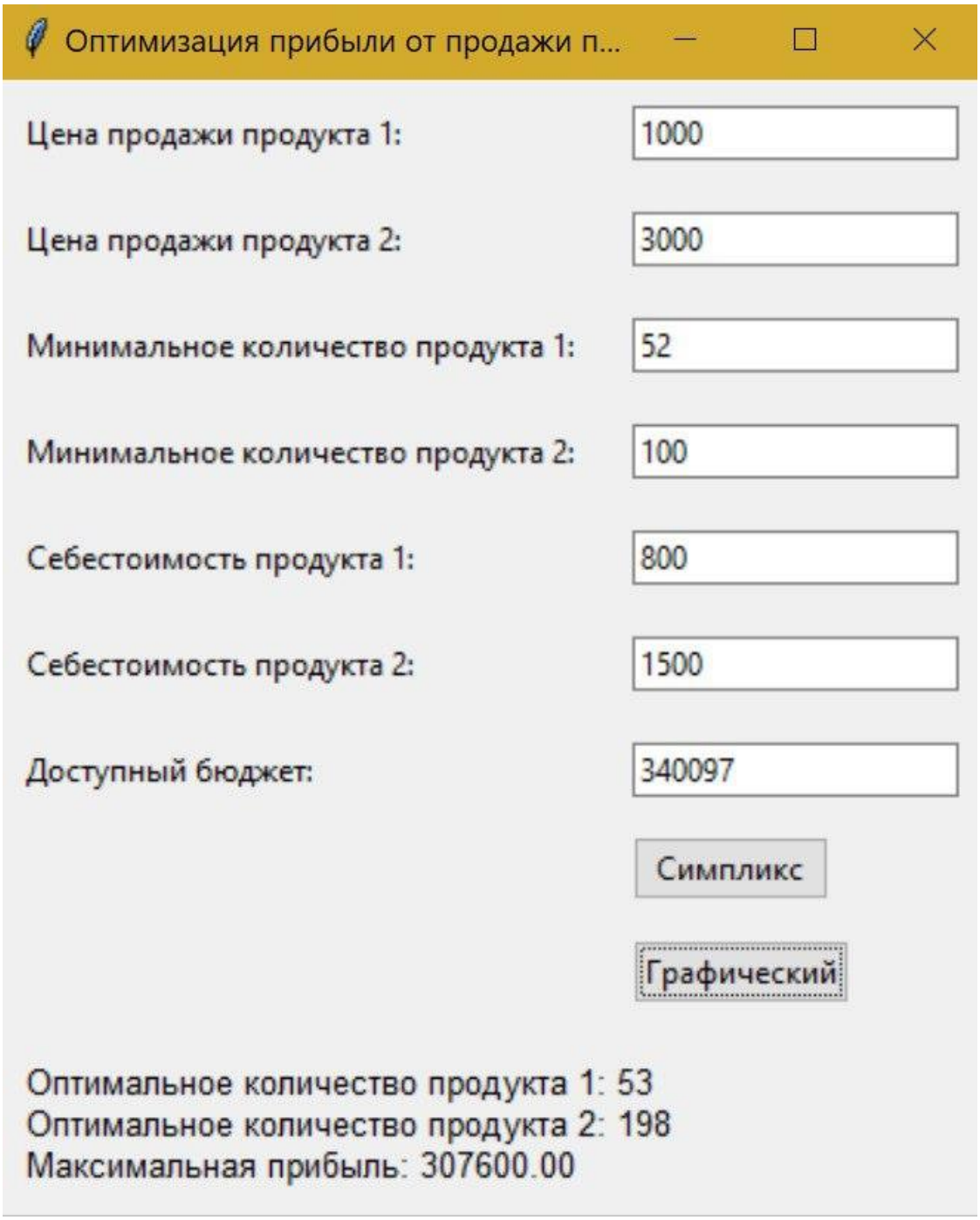

Рисунок 9 – Работа программы симплекс-методом

Возможности для дальнейшего развития и улучшения системы:

– дополнительные ограничения и параметры: включение переменных затрат - учёт переменных расходов в зависимости от объёма производства, учёт времени производства – добавление временных ограничений;

– разработка продвинутого интерфейса: интерактивные графики – визуализация результатов через графики и диаграммы;
– расширенный аналитический функционал: сценарный анализ – создание моделей "что если" для оценки изменений на рынке, учет рисков – анализ возможных рисков, связанных с изменениями цен и спроса.

Улучшение масштабируемости достигается параллельной обработкой.

Ускорение обработки данных на многопроцессорных системах, масштабируемость – адаптация системы к требованиям крупных производственных предприятий.

Выводы по главе 3

В третьей главе ВКР было проведено тестирование различных алгоритмов для решения задачи оптимизации выпуска готовой продукции на примере конкретной задачи.

Сравнительный анализ показывает, что алгоритм для решения задачи оптимизации выпуска готовой продукции должен основываться на ее конкретных требованиях. Алгоритм графического метода позволяет решать не сложные задачи с двумя переменными, в результате выдает простой и понятный пользователю график оптимизации производства, в то же время симплекс-метод дает краткий и понятный ответ по оптимизации производства и не имеет такого ограничения по переменным, и, следовательно, имеет большую практическую применяемость.

Заключение

В ходе выполнения ВКР на тему «Исследование и реализация алгоритма для решения задачи оптимизации выпуска готовой продукции» проведено исследование, объектом которого являлась задача оптимизации выпуска готовой продукции, различные методы и алгоритмы оптимизации выпуска готовой продукции.

Цель бакалаврской работы – исследование и реализация алгоритмов для решения задачи оптимизации выпуска готовой продукции.

В ходе данной работы поставлены и выполнены следующие задачи:

- выполнена постановка задачи исследования и проанализированы методы оптимизации выпуска готовой продукции: симплекс-метод, графический метод, метод Гомори. Графический метод оптимизации мало применим на практике, так как он сильно ограничен в количестве переменных. Симплекс-метод достаточно универсален и имеет хорошую точность вычисления, но затрачивает достаточно много ресурсов при решении объемных задач оптимизации. Метод Гомори является сложным в реализации и вычислительно затратным, так как требует многократного решения с добавлением новых ограничений на каждом шаге;
- проанализированы алгоритмы для решения задачи оптимизации выпуска готовой продукции таких методов как: графический метод и симплекс-метод;
- выполнена программная реализация и тестирование алгоритмов для решения задачи оптимизации выпуска готовой продукции. Выполнена реализация данных алгоритмов на языке Python. Как показали результаты тестирования, итог решения задачи оптимизации графическим методом удобный в использовании, наглядный и простой, а итог решения симплекс-методом точный, достаточно быстрый при не большом объеме задачи.

Алгоритм симплекс-метода, как правило, применяется в большинстве случаев решения задач оптимизации выпуска готовой продукции, он может быть внедрен почти в любое производство и успешно помогать в оптимизации. Алгоритм графического метода может быть применим лишь в отдельных случаях, так как у него достаточно жесткие требования к количеству переменных в задаче оптимизации выпуска готовой продукции.

Не существует идеального решения, объединяющего достоинства всех подходов, поскольку оптимизация одних характеристик может приводить к ухудшению других.

Выбор компромиссного решения зависит от требований к точности под конкретную задачу, вычислительным ресурсам, гибкости, масштабируемости и отказоустойчивости.

Результаты бакалаврской работы представляют научно-практический интерес и могут быть рекомендованы для анализа и программной реализации методов и алгоритмов для решения задачи оптимизации выпуска готовой продукции.

Список используемой литературы и используемых источников

1. Аксенов Е.П. Методы оптимальных решений // Пермь 2016. 91 с.
2. Амоа К. А., Рындин Н. А., Скворцов Ю. С. Разработка программных пакетов на языке python // Воронеж 2020. 63 с.
3. Анисимова Н. П., Ванина Е. А. Линейное программирование // Санкт-Петербург 2013. 17 с.
4. Белецкая С.Ю. Методы оптимизации в автоматизированных системах // Воронеж 2017. 155 с.
5. Гареева Г.А., Григорьева Д.Р. Методы линейного программирования // Набережные Челны 2019. 59 с.
6. Жидкова Н.В. Методы оптимизации систем / Мельникова О.Ю. Саратов 2018. 149 с.
7. Захаревич А.Л., Пфейфер Д.С., Кузикевич А.С. Нагрузочное тестирование для определения общей работоспособности представителей спортивного резерва // Минск 2018. 24 с.
8. Костин В.Н., Калинин А.Н. Методы оптимизации в примерах и задачах // Оренбург 2008. 153 с.
9. Кулаков К. А., Димитров В. М. Основы тестирования программного обеспечения // Петрозаводск 2018. 57 с.
10. Леонова Н.Л. Задачи линейного программирования и методы их решения Санкт-Петербург 2017. 77 с.
11. Меженная М. М., Гордейчук Т. В., Борисик М. М., Медведев О. С., Киринович И.Ф. Тестирование, оценка программного обеспечения // Минск 2016. 68 с.
12. Озерова Г. П. Основы программирования на языке Python в примерах и задачах // Владивосток 2022. 129 с.
13. Пероцкая В. Н. Градусов Д. А. Основы тестирования программного обеспечения // Владимир 2017. 100 с.
14. Рибчинский М. Р. Графический метод решения задач линейного

программирования // Москва 2016. 24 с.

15. Сысоева М. В., Сысоев И. В. Программирование для «нормальных» с нуля на языке Python // Москва 2023. 188 с.

16. Тарасов Р.В., Тарасов Д.В. Методы оптимизации в технологических и технических задачах // Пенза 2016. 156 с.

17. Филиповский В. М. Основы программирования и алгоритмизации // Санкт-Петербург 2022. 71 с.

18. Черноруцкий И. Г. Методы оптимизации // Санкт-Петербург 2012. 152 с.

19. Черноруцкий И. Г. Методы оптимизации в теории управления // Санкт-Петербург 2011. 375 с.

20. Шевченко А.С. Линейное программирование // Рубцовск 2021. 151 с.

21. M.J.D. Powell A view of algorithms for optimization without derivatives 2007. P. 12.

22. Mykel J. Kochenderfer Tim A. Wheeler Algorithms for Optimization 2019. P. 520.

23. Robert Fourer Elementary Linear Optimization Models 2005. P. 59.

24. Robert J. Marks, William A. Dembski, Winston Ewert Introduction to evolutionary mathematics 2020. P. 278.

25. Thomas Weise Optimization Algorithms 2023. P. 65.