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 Разработка алгоритма распределения ресурсов на предприятии АвтоВАЗ

Обучающийся В. В. Шленкин _____
(Инициалы Фамилия) (личная подпись)

Руководитель канд. пед.наук, доцент, О. М. Гущина
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Консультант канд. пед.наук, доцент, С. А. Гудкова
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Тольятти 2024

Аннотация

Работа была выполнена студентом Тольяттинского Государственного Университета, группы ПМИб-2002а, Шленкиным Владимиром Васильевичем.

Выпускная квалификационная работа по теме «Разработка алгоритма распределения ресурсов на предприятии АвтоВАЗ» посвящена реализации задачи оптимизации целевой функции с заданными ограничениями.

Цель работы - разработка программного обеспечения для распределения ресурсов на предприятии АвтоВАЗ с целью получения максимальной прибыли

Объект исследования - процесс оптимального использования ресурсов на предприятии.

Предмет исследования – алгоритм распределения ресурсов на предприятии с целью оптимизации производственных процессов, управления запасами и ресурсами, а также повышения эффективности использования ресурсов и снижения затрат.

Задачи работы:

- выбрать оптимальный алгоритм оптимизации целевой функции;
- выбрать модель и средства для разработки программного обеспечения;
- реализовать выбранный алгоритм в программном обеспечении;
- протестировать полученный продукт и исправить ошибки, если они будут.

Выпускная квалификационная работа содержит пояснительную записку объемом 40 страниц, состоит из введения, трех разделов, заключения, списка литературы, состоящего из 25 литературных источников и 24 рисунков.

Abstract

The work was performed by Vladimir Vasilyevich Shlenkin, a student of Tolyatti State University, PMIb-2002a group.

The title of the graduation work is «Development of an algorithm for allocating resources at the AvtoVAZ enterprise» is devoted to the implementation of the task of optimizing the objective function with specified constraints.

The purpose of the work is to develop software for the allocation of resources at the AvtoVAZ enterprise in order to maximize profits.

The object of the study is the process of optimal use of resources in the enterprise.

The subject of the study is an algorithm for allocating resources in an enterprise in order to optimize production processes, manage stocks and resources, as well as increase resource efficiency and reduce costs.

To achieve the purpose, the following tasks have to be performed:

- choose the optimal algorithm for optimizing the objective function;
- choose a model and tools for software development;
- implement the selected algorithm in the software;
- test the resulting product and correct errors if there are any.

The graduation work contains an explanatory note of 40 pages, consists of an introduction, three sections, a conclusion, a list of references consisting of 25 literary sources and 24 illustrations.

Содержание

Введение

В текущий момент многие промышленные предприятия в Российской Федерации сталкиваются с санкциями со стороны иностранных государств. Это может привести к различным проблемам, которые могут негативно сказаться на бизнесе и финансовой производительности компаний. Одной из основных проблем, связанных с санкциями, является прерывание поставок ключевых материалов или услуг, необходимых для бизнеса. Иностранные санкции могут ограничить доступ предприятий к определенным рынкам или поставщикам, что может привести к дефициту необходимых ресурсов. Это может привести к задержкам в производственном процессе или предоставлении услуг, а также к увеличению затрат на поиск источников альтернативных ресурсов. Кроме того, санкции могут повлиять на финансовую производительность компании. Ограничения в доступе к международным рынкам и партнерам могут снизить объемы продаж и прибыль предприятий. Это, в свою очередь, может затруднить конкуренцию на рынке и снизить финансовую устойчивость компании. В таких условиях остро стоит вопрос о наилучшем использовании ресурсов с целью получения максимальной прибыли и обеспечения устойчивого развития промышленных предприятий в Российской Федерации. Разработка эффективного алгоритма распределения ресурсов на примере предприятия "АвтоВАЗ" становится крайне актуальной задачей, поскольку позволит минимизировать негативное воздействие санкций на производственные процессы и финансовые показатели компании, а также обеспечить ее конкурентоспособность на рынке.

В ходе выполнения данной работы объектом исследования будет являться процесс оптимального использования ресурсов на предприятии. В данном объекте исследования выделим предмет, который будем изучать в ходе написания данной ВКР – алгоритм распределения ресурсов на предприятии с целью оптимизации производственных процессов, управления

запасами и ресурсами, а также повышения эффективности использования ресурсов и снижения затрат.

Цель данной выпускной квалификационной работы состоит в разработке программного обеспечения для распределения ресурсов на предприятии АвтоВАЗ для получения максимальной прибыли посредством реализации специальных алгоритмов и методов оптимизации.

В процессе выполнения работы необходимо будет выполнить следующие задачи:

- выбрать оптимальный алгоритм оптимизации целевой функции;
- выбрать модель и средства для разработки программного обеспечения;
- реализовать выбранный алгоритм в программном обеспечении;
- протестировать полученный продукт и исправить ошибки, если они будут.

В первом разделе мы рассмотрим задачу более подробно и проанализируем существующие решения и их недостатки.

Во втором разделе мы проведем анализ методов оптимизации, выберем наиболее подходящей для нашей задачи и подробно рассмотрим его алгоритм, после чего реализуем его в программном обеспечении.

Третий раздел будет содержать в себе тестирование полученного продукта и его доработку. После чего будет сделан вывод о проделанной работе и полученных результатах.

1 Теоретическое обоснование задачи распределения ресурсов

1.1 Описание исследуемой задачи распределения ресурсов на предприятии

Задача об оптимальном распределении ресурсов на предприятии для получения максимальной выгоды состоит в создании алгоритма или модели, которые позволят эффективно использовать имеющиеся ресурсы (такие как сырье, трудовые ресурсы, оборудование и капитал) с целью максимизации прибыли или других целевых показателей.

Для решения этой задачи следует пройти ряд этапов:

- анализ потребностей и возможностей предприятия. Провести анализ текущих производственных процессов, потребностей рынка, а также оценить доступные ресурсы и их возможности использования;
- определение целей и критериев эффективности. Определить цели, которые должны быть достигнуты при оптимизации распределения ресурсов, такие как максимизация прибыли, минимизация затрат, оптимизация производственных показателей и т. д. Также необходимо выбрать критерии, которые будут использоваться для оценки эффективности алгоритма;
- разработка математической модели или алгоритма. На основе полученной информации разработать математическую модель или алгоритм, который будет оптимально распределять ресурсы с учетом поставленных целей и критериев эффективности;
- проверка и адаптация. Провести проверку разработанного алгоритма на реальных данных или в условиях симуляции. В случае необходимости внести корректировки и адаптировать модель или алгоритм для более точного соответствия реальным условиям предприятия;
- внедрение и мониторинг. Внедрить разработанный алгоритм на

предприятии и регулярно мониторить его работу, а также проводить анализ результатов с целью выявления возможных улучшений или корректировок.

Данная задача может быть применена не только на предприятии ООО «АвтоВАЗ», но и на любом другом производстве, где стоит остро стоит вопрос о наилучшем распределении ресурсов, что делает эту задачу широко применимой.

1.2 Анализ существующих решений распределения ресурсов

В данный момент большой популярностью пользуется табличный процессор MS Excel. В нем также есть функции для оптимизации. Например, модуль Solver, который позволяет оптимизировать значение целевой функции, изменяя значения некоторых переменных [24]. Это очень полезно для решения задач линейного и нелинейного программирования. Пользователь может указать целевую функцию, ограничения и переменные, а затем Solver попытается найти оптимальное решение, максимизируя или минимизируя значение целевой функции [25]. Также стоит отметить инструменты анализа данных в Excel, такие как таблица сценариев, могут быть использованы для анализа различных вариантов и сценариев на основе изменения значений переменных. Это позволяет исследовать, как изменения в одной части таблицы могут повлиять на другие части и, следовательно, на целевую функцию.

Однако, хоть и Excel является мощным инструментом для работы с данными и выполнения различных аналитических задач, включая оптимизацию, у него также есть несколько минусов при использовании функций оптимизации. Давайте рассмотрим их более детально:

Во-первых, сложные задачи оптимизации, например решение больших систем линейных уравнений или нелинейных задач, могут быть трудноразрешимы в MS Excel и занимать большое количество времени.

Во-вторых, при использовании модуля Solver могут возникнуть проблемы недостаточной точности при решении сложных оптимизационных задач. Это связано с ограниченными возможностями численных методов, используемых в Excel.

В-третьих, в Excel есть ограничения на количество переменных и ограничений, которые можно учесть при решении задач оптимизации. Это может затруднить использование программы при работе с большими моделями.

В-четвертых, для использования функций оптимизации в Excel от пользователя требуется глубокого понимания инструментов и методов анализа данных, что может составить трудности начинающим работникам.

Исходя из всего вышесказанного, можно сделать вывод, что табличный процессор Microsoft Excel хоть и является очень распространённым средством, однако имеет ряд недостатков, которые могут стать критичными.

Исходя из всего вышесказанного, основной задачей данной работы будет являться разработка программного обеспечения для работников ООО «АвтоВАЗ», которое будет оптимизировать целевую функцию с ограничениями. Главные требования к разрабатываемому продукту: реализация вычислений на стороне сервера, точность вычислений при большом количестве переменных, удобство и простота использования.

Вывод и результаты по первому разделу:

В данном разделе была рассмотрена общая постановка задачи об оптимальном использовании ресурсов, этапы ее решения. Также было рассмотрено существующее решение и выявлены его конкретные недостатки для больших предприятий.

2 Математическая формулировка модели распределения ресурсов на предприятии

2.1 Анализ и выбор вычислительного метода оптимизации ресурсов

Рассмотрим несколько численных методов оптимизации ресурсов для задач линейного программирования:

- симплекс-метод. Это классический метод решения задач ЛП, который оперирует на вершинах многогранника области допустимых решений. Он последовательно перемещается от одной вершины к другой, улучшая значение целевой функции до достижения оптимального решения [10][16];
- метод внутренней точки. Этот метод используется для нахождения решений внутри многогранника допустимых значений, вместо перемещения по его вершинам, как это делает симплекс-метод. Метод внутренней точки обычно сходится быстрее на практике, особенно для больших задач ЛП [1][20];
- двойственный симплекс-метод. Этот метод основан на симплекс-методе, но решает сразу исходную задачу и соответствующую ей двойственную, что может быть полезно, когда требуется оптимизировать структуру стоимости [2];
- метод градиентного спуска. Хотя он обычно используется для задач нелинейной оптимизации, его можно также адаптировать под задачи линейного программирования [12][18].

Все из вышеперечисленных методов оптимизации имеет свои преимущества и недостатки. Выбор какого-то конкретного метода зависит от таких факторов как: размерность пространства переменных, свойства целевой функции, наличие ограничений и требования к точности решения.

В данной работе мы рассмотрим алгоритм градиентного спуска, исходя

из следующих его преимуществ:

- простота реализации. «Метод градиентного спуска – один из самых простых методов оптимизации. Его концепция легко понятна и реализуется сравнительно легко» [4][23];
- масштабируемость. Метод градиентного спуска может быть эффективно применен к задачам с большим числом переменных. При правильном выборе скорости обучения и других параметров он может справляться с большими объемами данных, что не может не быть учтено при применении на предприятии таких масштабов как АвтоВАЗ;
- простота настройки начальных параметров. Метод градиентного спуска имеет относительно небольшое количество начальных параметров, таких как скорость обучения (learning rate) и критерий остановки [3]. Это делает его относительно простым в настройке и позволяет быстро начать работу над задачей оптимизации, в то время как другие методы могут иметь больше параметров, требующих тщательной настройки для достижения хороших результатов [20].

Таким образом, можно сделать вывод о том, что учитываю специфику нашей задачи, из описанных выше методов целесообразно будет выбрать метод градиентного спуска.

2.2 Общая структура алгоритма метода градиентного спуска

Рассмотрим математическую модель метода градиентного спуска для максимизации целевой функции. Обычно данный метод используется для минимизации функции, но его можно адаптировать и для максимизации. Рассмотрим метод по шагам.

Шаг 1. Инициализация.

Предположим, у нас есть целевая функция f(x), которую мы хотим максимизировать. Выбирается начальное значение x_0, после чего

устанавливается значение коэффициента скорости обучения a.

Шаг 2. Итерационный процесс.

Мы также предполагаем, что у нас есть доступ к градиенту этой функции ∇f(x), который показывает направление наибольшего возрастания функции в точке x. Метод градиентного спуска обновляет текущую оценку x в направлении градиента, чтобы двигаться к точке максимума. Формула обновления может быть записана как (1).

$$x_{n+1} = x_n + a\nabla f(x_n), \quad (1)$$

где

x_n – текущая оценка параметра,

x_{n+1} – новая оценка параметра x,

a – коэффициент скорости обучения (learning rate), который определяет размер шага.

Шаг 3: Условие остановки.

Этот процесс продолжается до тех пор, пока не будет достигнуто условие остановки:

- градиент целевой функции приближается к нулю $\|\nabla f(x)\| \approx 0$;
- достигается заданное максимальное количество итераций.

Шаг 4: Оптимизация с ограничениями.

Для задач с ограничениями метод градиентного спуска может быть адаптирован с использованием различных техник, таких как метод проекции, метод множителей Лагранжа и другие. Например:

- метод проекции: после каждого шага обновления, новое значение параметра проецируется на допустимое множество значений;
- метод множителей Лагранжа: в задачи вводятся дополнительные переменные (множители Лагранжа) для учета ограничений и составляется расширенная целевая функция, которая затем оптимизируется.

Далее рассмотрим структуру алгоритма распределения ресурсов на предприятии АвтоВАЗ в рамках нашей задачи оптимизации с ограничениями. Схема алгоритма представлена на рисунке 1.

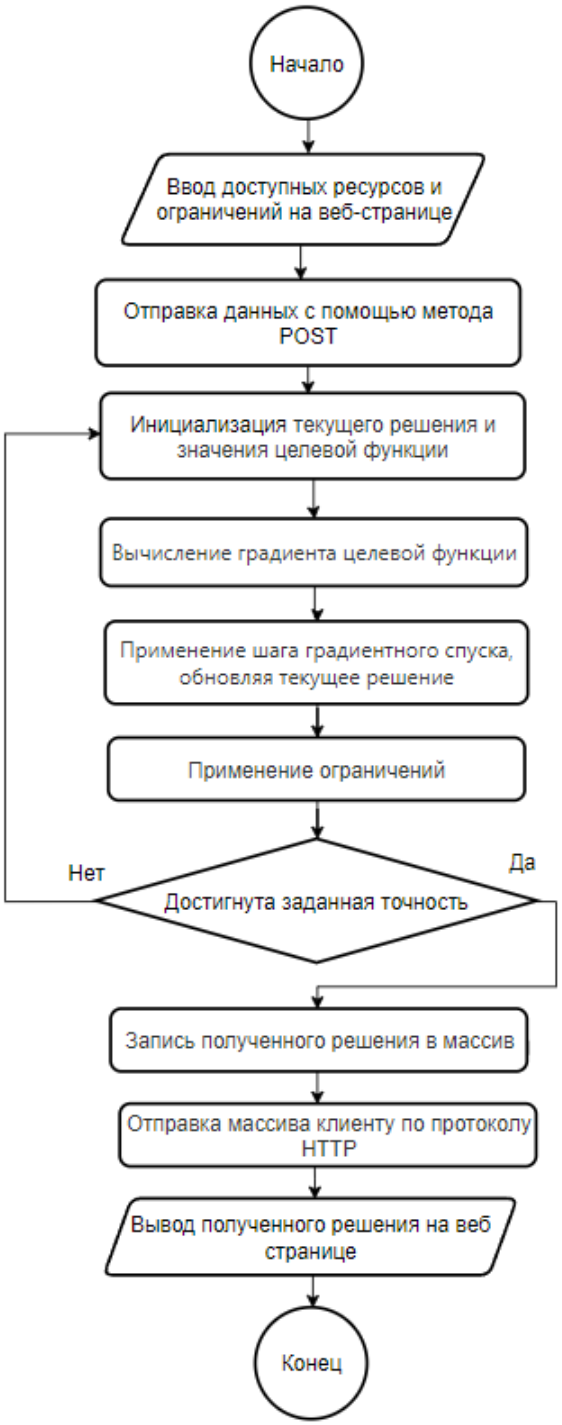


Рисунок 1 – Алгоритм метода градиентного спуска

Также стоит отметить, что метод градиентного спуска имеет широкую применимость и может быть использован для различных задач, включая линейную регрессию, логистическую регрессию, нейронные сети и другие модели машинного обучения [13].

2.3 Общая структура алгоритма решения задачи оптимизации ресурсов

Так как процесс оптимизации целевой функции для большого количество ресурсов может потребовать сильных вычислительных мощностей, то оснащение большого количества работников компьютерами с высокой производительностью является дорогостоящим. Следовательно, оптимальным решением является применение метода, в котором основные вычисления будут производиться на сервере, а сторона клиента будет использоваться лишь для ввода данных и получения результата. В качестве архитектурного стиля для нашего приложения выберем REST API.

«REST (Representational State Transfer) – это архитектурный стиль разработки веб-сервисов, который обеспечивает стандартизированный способ взаимодействия между клиентом и сервером посредством передачи представлений ресурсов, позволяющий эффективно организовать взаимодействие между клиентскими приложениями и сервером, минимизируя затраты на обработку данных на стороне клиента» [15]. REST API основан на нескольких принципах, включающих описание ресурсов и методов.

Ресурсы: Все данные представляются как ресурсы, к которым можно получить доступ по определенному URI (Uniform Resource Identifier). В нашем случае, ресурсом может быть набор данных, который пользователь вводит на странице HTML.

HTTP методы: REST API использует HTTP методы для указания действий, которые нужно выполнить с ресурсами. Наиболее часто используемые методы:

– GET: получить данные с сервера;
– POST: отправить данные на сервер для создания нового ресурса;
– PUT: отправить данные на сервер для обновления существующего ресурса;
– DELETE: удалить ресурс с сервера.

Ресурсы могут быть представлены в различных форматах, таких как JSON, XML и другие. Наш сервер будет возвращать результат вычислений в формате JSON. Каждый запрос от клиента к серверу должен содержать всю необходимую информацию для обработки запроса. Сервер не должен хранить информацию о состоянии клиента между запросами [14][17]. Клиенты могут кэшировать ответы сервера для улучшения производительности.

Архитектура REST API представлена на рисунке 2.

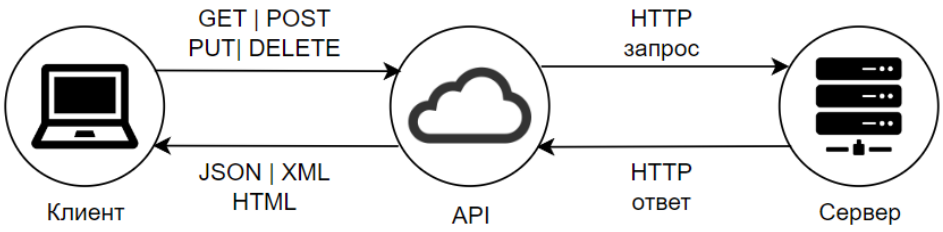


Рисунок 2 – Схема REST API

Реализовывать данную архитектуру мы будем с помощью фреймворков Node.js и express.

2.4 Реализация программных модулей алгоритма решения задачи оптимизации ресурсов

«Node.js – это среда выполнения JavaScript, построенная на движке V8 от Google Chrome. Она позволяет запускать код JavaScript на сервере, что делает его идеальным инструментом для создания масштабируемых и быстрых сетевых приложений» [8]. В Node.js использует неблокирующий

ввод/вывод, что позволяет обрабатывать множество запросов без блокировки потоков. Это достигается с помощью механизма обратного вызова (callback), обещаний (Promises) и асинхронных функций. Благодаря асинхронной природе и использованию событий, Node.js обеспечивает высокую производительность при обработке большого количества одновременных запросов [21].

Так же стоит отметить, что данная среда выполнения включает в себя встроенную систему модулей (CommonJS), которая позволяет разбивать приложение на отдельные компоненты с возможностью повторного использования кода. Совместно с Node.js будем использовать веб-фреймворк Express, который облегчает создание веб-приложений и API. Он построен поверх HTTP модуля Node.js и предоставляет богатый набор функций для обработки запросов, маршрутизации, работы с шаблонами и многое другое [8][19].

Основная особенность Express является простота и минимализм. Данный фреймворк предлагает минимальный набор функций, что делает его легким в изучении и использовании [22]. Так же Express.js использует концепцию промежуточного ПО, которое позволяет выполнять функции обработки запроса до того, как они достигнут маршрутов [19]. Промежуточное ПО может выполняться последовательно и выполнять различные функции, такие как аутентификация, логгирование и обработка ошибок.

Следует отметить, что Express позволяет определять обработчики запросов для различных URL-адресов и методов HTTP. Это делается с помощью методов, таких как «app.get()», «app.post()», «app.put()» и «app.delete()» [21]. Диаграмма архитектуры разрабатываемого приложения показана на рисунке 3.

Первым шагом создадим корневую папку приложения, и назовем ее «server». Для создания проекта Node.js напишем команду «npm init -y», после чего в нашей папке появится конфигурационный файл «package.json», который будет хранить в себе все зависимости в проекте.

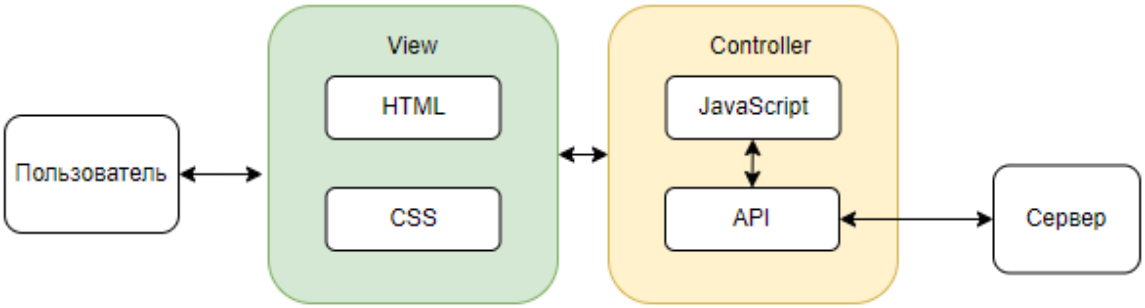


Рисунок 3 – Диаграмма архитектуры приложения

Далее создадим главный файл нашего приложения «index.js». В начале данного файла подключим необходимые библиотеки для приложения, такие как express, cors, path и body-parser. Затем создадим константу «app», и запишем в нее объект класса express. Далее, вызывая методы этого объекта, мы можем настраивать наше приложение.

Для разработки веб приложения, запускать сервер будем на локальном хосте. Создадим отдельный файл «.env», который будет хранить переменные среды. Внутри файла объявим переменную «PORT» и присвоим ей значения 5000. В файле index.js будем считывать номер порта из файла «.env», после чего будем вызывать метод listen у объекта app, передавая туда номер порта, на котором будет запускаться сервер. В случае успешного запуска выведем в консоль соответствующее сообщение. Исходный код файла index.js представлен на рисунке 4.

Далее создадим папку «views», в которой будем хранить файлы формата html и ejs для отображения страниц на стороне клиента. В данной папке создадим стартовую страницу index.html, которая будет отображаться при заходе на сервер. В начале страницы добавим тег <script>, чтобы внутри указать функцию javaScript.

Рисунок 4 – Файл index.js

В начале функции создадим переменную «nameOfMatrix». Значение этой переменной будет записано как «matrix + строка, переданная в функцию», чтобы элементы носили разное название. Далее создадим две переменные n и m, которые будут хранить данные о количестве строк и столбцов создаваемой матрицы соответственно. После чего добавим три структуры типа if, и в зависимости от типа создаваемой матрицы будем присваивать переменным n и m советующие значения. Так, например если нужно создать матрицу для коэффициентов целевой функции, то n будет равно 1, в m равно переменной numOfVar, которая обозначает число переменных в функции и значение который, мы получаем от пользователя.

Для того чтобы получить значение numOfVar используем метод getElementById("numOfVar ") у объекта document. Также применим к полученным данным метод ParseInt, чтобы мы получали именно число, а не строку.

Далее создадим два вложенных цикла для создания полей ввода матрицы. Внутри цикла на каждой итерации создаем элемент «input», тип элемента указываем «text», название же будет состоять из названия матрицы и номеров текущей итерации. Далее добавляем созданный элемент в

результирующий контейнер, который будет отображаться на странице.

Схема данных условий представлена на рисунке 5.

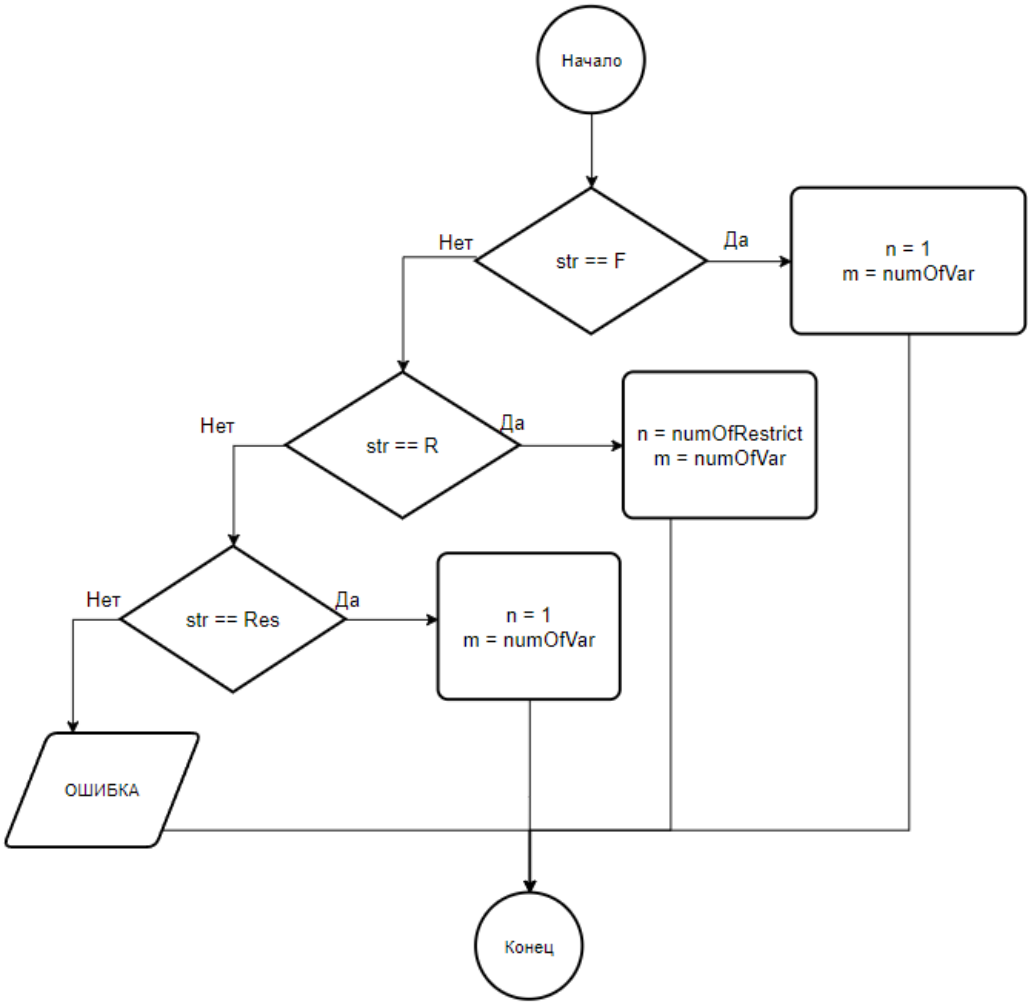


Рисунок 5 – Алгоритм объявления переменных n и m

Реализации функции «createMatrix» представлена на рисунке 6.

В элементе «body» нашей HTML страницы создадим форму для отправки введённых данных на сервер. Метод для отправки укажем POST, а в качестве пути, куда будет отправлены данные, укажем «/api/opt», где будет находиться контроллер.

В начале тела страницы создадим поле для ввода количества переменных, используемых в функции, типа «number». Так же создадим элемент «label» для поля ввода, чтобы вывести пользователю, какие данные

нужно вводить в данном поле.

```
<script>
    function createMatrix(str) {
      var matrixContainer = document.getElementById("matrixContainer"+str);
      matrixContainer.innerHTML = ""; // Очистка контейнера перед созданием новой матрицы

      let nameOfMatrix="matrix"+str;;
      var n;
      var m;
      if (str=='F'){
        n=1
        m=parseInt(document.getElementById("numOfVar").value);

      }
      if (str=="R"){
        n=parseInt(document.getElementById("numOfRestrict").value);
        m=parseInt(document.getElementById("numOfVar").value);
      }
      if(str=="Res"){
        n=1;
        m=parseInt(document.getElementById("numOfVar").value);
      }


      for (var i = 0; i < n; i++) {
        for (var j = 0; j < m; j++) {

          var input = document.createElement("input");
          input.type = "text";
          input.name = nameOfMatrix+"[" + i + "][" + j + "]";
          matrixContainer.appendChild(input);
        }

        var lineBreak = document.createElement("br");
        matrixContainer.appendChild(lineBreak);
      }
    }
  </script>
```

Рисунок 6 – Функция «createMatrix»

Ниже создадим кнопку «Создать матрицу», по нажатию которой будет два раза вызываться функция «createMatrix». В первый раз в функцию передаем строку «F», для создания матрицы коэффициентов целевой функции, затем передадим строку «Res» для создания матрицы доступных ресурсов. После чего добавим созданные элементы драг под другом.

Аналогичным образом добавим поле ввода количества уравнений ограничений с кнопкой создания матрицы для их ввода. В самом конце формы добавим кнопку типа «submit» для отправки данных на сервер.

Полученная HTML-страница и ее исходный код приставлены на рисунках 7 и 8.

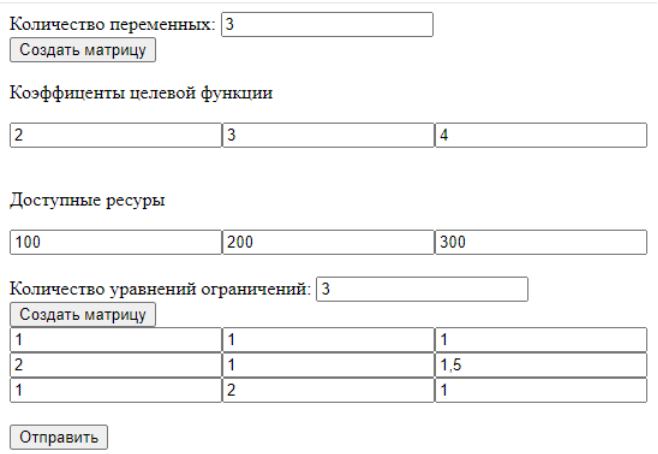


Рисунок 7 – Страница «index.html»

```
<body>

    <form method="post" action="/api/opt">

      <label for="numOfVar">Количество переменных:</label>
      <input type="number" id="numOfVar" name="numOfVar">

      <br>

      <button type="button" onclick="createMatrix('F'), createMatrix('Res')">Создать матрицу</button>

      <br>
      <p>Коэффициенты целевой функции</p>
      <div id="matrixContainerF"></div>
      <br>
      <p>Доступные ресуры</p>
      <div id="matrixContainerRes"></div>

      <br>

      <label for="numOfRestrict">Количество уравнений ограничений:</label>
      <input type="number" id="numOfRestrict" name="numOfRestrict">

      <br>

      <button type="button" onclick="createMatrix('R')">Создать матрицу</button>

      <br>

      <div id="matrixContainerR"></div>

      <br>

      <input type="submit" value="Отправить">
  </form>

</body>
</html>
```

Рисунок 8 – Исходный код «index.html»

Следующим шагом создадим папку routes, в которой будут находиться файлы, связанные с URL-адресами нашего веб-приложения. В данной папке создадим файл router.js. Первым делом экспортируем класс «Route» из библиотеки express, после чего создадим объект данного класса. Далее экспортируем класс «optController», реализация которого будет управлять переданными данными. Затем через метод «post» объекта router укажем путь «/opt», и функцию «optimize» из импортированного ранее контроллера. Таким образом при отправке POST запроса по адресу «/opt» будет вызываться функция optimize. В конце файла экспортируем созданный объект, чтобы использовать его в файле index.js. В файле index.js вызываем метод use, у объекта app, и передаем в него эскортированный класс router. Теперь наше приложение будет использовать пути из этого класса. Полученный код представлен на рисунке 9.

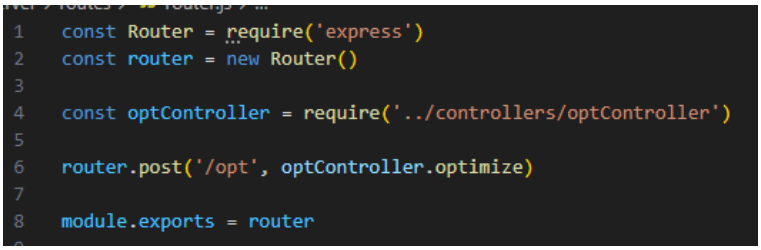


Рисунок 9 – Исходный код файла «router.js»

Затем создадим папку controllers, в которой будут храниться наши контроллеры. Создадим в ней файл optController.js, в котором в свою очередь создадим функцию optimize, указанную в файле router.js для пути «api/opt». Данная функция принимает два параметра: req – запрос и res – ответ. В начале функции создадим три переменных: matrixCols – хранящая количестов столбцов в матрице, numOfRestrict – хранящая количество неравенств ограничений и matrixData – матрицы ограничений, коэффициентов целевой функции и доступных ресурсов.

Чтобы получить данные для переменной matrixCols, обратимся к

объекту req и его свойству body. Далее укажем название нашего элемента на реализованной ранее HTML-странице – numOfVar. Таким образом, мы считываем переменные, переданные из запроса в данный контроллер. Так же обернем данную процедуру в метод parseInt, чтобы значения считывались в нужном нам типе. Аналогичную процедуру проведем для переменной numOfRestrict. А вот для Данных матрицы используем другой подход. Так как матрицы у нас три, то переменной matrixData присвоим значение req.body, без указания конкретной переменной. Таким образом, мы получим все переменные с запроса в формате JSON, после чего останется лишь разбить их на соответствующие массивы.

Затем создадим все необходимые массивы: arrFunc – массив коэффициентов целевой функции, arrRes – массив доступных ресурсов, arrRestrict – массив коэффициентов ограничений. Первые два массива будут иметь тип float и размер равным количеству переменным, а вот для ограничений необходимо создать двумерный массив, поэтому тип данного массива будет array, а размер соответствовать количеству неравенств ограничений. Далее в цикле проходимся по этому массиву и создаем еще одни массив тип float. Таким образом мы создали двумерный массив.

Далее заполним три созданных массива данным из JSON объекта, полученного из запроса. Для этого создадим три цикла, внутри которых будем обращаться к переменной matrixData. В качестве ключа будем указывать название необходимого массива и текущую итерацию цикла. Реализация начала файла представлена на рисунке 10.

Следующим шагом объявим переменные, необходимы для процесса градиентного спуска. Начальное приближение укажем как массив float, размер которого равен количеству переменных целевой функции. Затем в цикле заполним каждый его элемент значением 0,5. Скорость обучения укажем 0,1, допустимую погрешность – $1*e^{-6}$. Далее вызовем метод градиентного спуска, в которой передадим все вышеперечисленные параметры. Результаты запишем в переменную solution. Также рассчитаем значение целевой функции

для найденного решения.

```
async optimize(req, res){

    const matrixCols = parseInt(req.body.numOfVar);
    const numOfRestrict = parseInt(req.body.numOfRestrict);
    const matrixData = req.body;


    //Создаем массивы
    let arrFunc = new Float32Array(matrixCols);
    let arrRes= new Float32Array(matrixCols);
    let arrRestrict = new Array(numOfRestrict);

    //Создаем вложенные массивы для матрицы массива ограничений
    for (var i = 0; i < numOfRestrict; i++) {
        arrRestrict[i] = new Float32Array(matrixCols);
    }

    //Заполняем массивы данными из JSON-объекта
    for (var i = 0; i < numOfRestrict; i++) {
        for (var j = 0; j < matrixCols; j++) {
            arrRestrict[i][j]= matrixData["matrixR["+i+"]["+j+"]"];
        }
    }

    for (var i = 0; i < matrixCols; i++) {
        arrRes[i] =  matrixData["matrixRes[0]["+i+"]"];
    }

    for (var i = 0; i < matrixCols; i++) {
        arrFunc[i] =  matrixData["matrixF[0]["+i+"]"];
    }
```

Рисунок 10 – Начало файла «optController.js»

Следующим шагом необходимо передать полученные результаты на сторону клиента. Для этого сперва преобразуем результаты в формат JSON методом JSON.stringify. После этого вызовем метод render, у объекта res, в которой передадим название страницы, которая будет отображаться у клиента, а также полученные JSON-объекты. Также в самом конце данного файла экспортируем созданный класс, чтобы его можно было использовать в файле route.js. полученный результат представлен на рисунке 11.

Рисунок 11 – Конец файла «optController.js»

На следующем этапе создадим модуль, отвечающий за реализацию градиентного спуска. В той же папке controllers создадим файл gradient.js. Для расчета значения целевой функции в каком-либо решение создадим функцию «calculateObjectiveValue», которая принимает на вход коэффициенты целевой функции и массив значений переменных. К последнему массиву применим метод reduce, для обратного вызова к каждому элементу массива. Для каждого элемента массива переменных и соответствующего коэффициента целевой функции функция вычисляет их произведение и добавляет его к накопленной сумме. После обхода всех элементов функция возвращает итоговое значение, которое представляет собой значение целевой функции для данного решения.

Следующим шагом создадим в этом файле функцию «maximizeObjectiveWithGradientDescent». Данная функция принимает семь параметров:

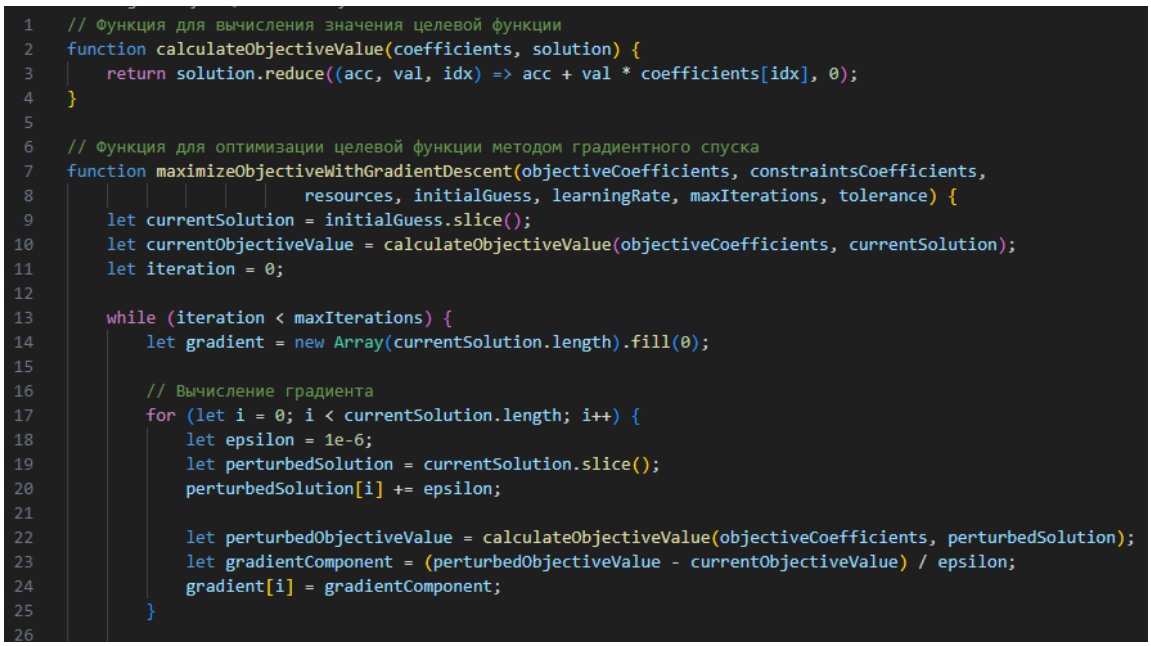
– массив коэффициентов целевой функции;
– массив коэффициентов ограничений;
– массив доступных ресурсов;
– массив начального приближения;

– величина шага обучения;
– максимальное количество итераций;
– точность вычислений.

Затем инициализируем переменную «currentSolution», которая обозначает текущее решение, инициализируется начальным приближением. Для этого у массива начального приближения вызовем метод slice(), который создает копию массива. Мы должны использовать копию для того, чтобы изменения в одном массиве не сказывались на изменения в другом. Далее создадим переменную «currentObjectiveValue», хранящая значения целевой функции при текущем решении. Для ее инициализации используем функцию, описанную выше. Также создадим переменную для отслеживания количества итераций.

Далее объявим цикл с предусловием, который будет выполняться, пока количество итераций не достигнет максимального значения. Внутри цикла первым шагом инициализируем массив gradient, представляющий градиент целевой функции и заполняем его нулями. После чего создадим цикл для вычисления градиента, который будет проходить по всем переменным решения и каждая переменная будет рассматриваться отдельно. Внутри цикла будем создавать копию массива текущего решения и прибавлять к i-той переменной очень малое число, тем самым получать новое решение. После чего вычислим значение целевой функции в полученном решении.

Затем мы будем высчитывать значение компоненты для i-той переменной. Она вычисляется как разница между значениями целевой функции для текущего и нового решений, деленная на какое-либо очень малое число. Это приближенное значение производной целевой функции по i-той переменной в точке текущего решения. В конце цикла запишем полученный результат в массив градиента. Результат представлен на рисунке 12.



```
// Функция для вычисления значения целевой функции
function calculateObjectiveValue(coefficients, solution) {
    return solution.reduce((acc, val, idx) => acc + val * coefficients[idx], 0);
}

// Функция для оптимизации целевой функции методом градиентного спуска
function maximizeObjectiveWithGradientDescent(objectiveCoefficients, constraintsCoefficients,
                                              resources, initialGuess, learningRate, maxIterations, tolerance) {
    let currentSolution = initialGuess.slice();
    let currentObjectiveValue = calculateObjectiveValue(objectiveCoefficients, currentSolution);
    let iteration = 0;

    while (iteration < maxIterations) {
        let gradient = new Array(currentSolution.length).fill(0);

        // Вычисление градиента
        for (let i = 0; i < currentSolution.length; i++) {
            let epsilon = 1e-6;
            let perturbedSolution = currentSolution.slice();
            perturbedSolution[i] += epsilon;

            let perturbedObjectiveValue = calculateObjectiveValue(objectiveCoefficients, perturbedSolution);
            let gradientComponent = (perturbedObjectiveValue - currentObjectiveValue) / epsilon;
            gradient[i] = gradientComponent;
        }

```

Рисунок 12 – Начало файла «gradient.js»

Далее вычислим новое приближенное значение. У массива с текущем решением вызовем метод map, для изменения всех элементов на сумму текущего значения и произведения шага обучения на соответствующий элемент в массиве градиента.

Затем необходимо учесть имеющиеся ограничения. Создадим цикл, который будет проходиться по массиву нового решения, чтобы отдельно проверять каждый вид ресурса. Первым шагом проверим, не превышает ли текущее значение все имеющиеся ресурсы. Для этого используем метод «min» из встроенной библиотеки Math. Затем для каждого ограничения, представленного в массиве, вычисляется сумма произведений коэффициентов ограничения на соответствующие значения переменных в новом решении. Выполним это с помощью метода «reduce». Если значение ограничения превышает доступные ресурсы, то решение корректируется путем масштабирования каждой переменной так, чтобы ограничение было выполнено. Коэффициент масштабирования будет вычисляться как отношение доступных ресурсов к значению ограничения. Это определяет, во сколько раз нужно уменьшить значения переменных, чтобы ограничение было

выполнено. Далее с использованием метода «map» для массива с новыми решениями производится масштабирование каждой переменной. Функцию обратного вызова «reduce» реализуем таким образом, чтобы она изменяла только одно значение в полученном решении, а все остальные оставляла без изменения.

Следующим шагом произведем проверку сходимости алгоритма градиентного спуска. Вычислим значение целевой функции при новом решении. После чего сравниваем модуль разности нового и старого значения. Если он меньше, чем точность, переданная в функцию, то цикл прерывается. В конце цикла переписываем новое решение и новое значение целевой функции в значение текущих данных, а также увеличиваем значений счетчика итераций на единицу. В конце функции возвращаем последний полученный массив решений.

Также в конце файла необходимо экспортировать две полученные функции, чтобы мы могли их использовать в контроллере. В свое время, в контроле импортируем две функции из написанного файла. Полученный результат представлен на рисунках 13 и 14.

```
27        // Применение градиентного шага
28        let newSolution = currentSolution.map((value, index) => value + learningRate * gradient[index]);
29
30        // Применение ограничений
31        for (let i = 0; i < newSolution.length; i++) {
32            // Учитываем доступные ресурсы как ограничения
33            newSolution[i] = Math.min(newSolution[i], resources[i]);
34
35            // Учитываем ограничения
36            for (let j = 0; j < constraintsCoefficients.length; j++) {
37                let constraintValue = constraintsCoefficients[j].reduce((acc, val, idx) =>
38                                                                    acc + val * newSolution[idx], 0);
39                if (constraintValue > resources[j]) {
40                    let scaleFactor = resources[j] / constraintValue;
41                    newSolution = newSolution.map((value, index) => index === i ? value * scaleFactor : value);
42                    // После коррекции решения, выходим из внутреннего цикла
43                    break;
44                }
45            }
46        }
```

Рисунок 13 – Продолжение файла «gradient.js»

```
        // Проверка сходимости
        let newObjectiveValue = calculateObjectiveValue(objectiveCoefficients, newSolution);
        if (Math.abs(newObjectiveValue - currentObjectiveValue) < tolerance) {
            break;
        }

        currentSolution = newSolution;
        currentObjectiveValue = newObjectiveValue;
        iteration++;
    }
    console.log('iter', iteration)
    return currentSolution;
}

module.exports = { maximizeObjectiveWithGradientDescent,  calculateObjectiveValue};
```

Рисунок 14 – Конец файла «gradient.js»

Далее в папке «views» создадим файл result, который будет отображать полученные результаты. Исходный код и полученная страница с результатами представлены на рисунках 15 и 16.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Result</title>
</head>
<body>
    <h1>Резуьтаты</h1>
    <p>Распределение ресурсов:</p>
    <ul>
        <% roundedSolution.forEach(function(item, index) { %>
            <li>Ресурс №<%= index + 1 %>: <%= item %></li>
        <% }); %>
    </ul>
    <p>Значение целевой функции: <%= jsonValue %></p>
</body>
</html>
```

Рисунок 15 – Исходный код файла «result.ejs»

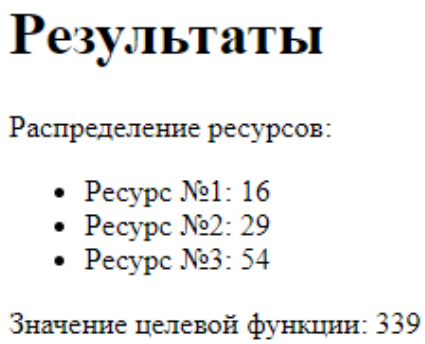


Рисунок 16 – Страница с полученными результатами

Так как при отправки из контроллера результатов мы используем шаблонизатор EJS. Поэтому файл result будет в формате не HTML, а ejs. В теле данной страницы будем перечислять полученное распределение ресурсов в виде списка, для этого используем тег «ul», внутри которого реализуем функцию forEach для массива решения. Внутри данного метода укажем функцию, которая, используя объекты index и item выводит номер ресурса и его распределение. После данного списка также выведем значение целевой функции при полученном решении.

Вывод и результаты по второму разделу:

В данном разделе был проведен анализ методов оптимизации и был выбран наилучший, исходя из специфики поставленной задачи. Далее был рассмотрен алгоритм решения задачи оптимизации методом градиентного спуска. После чего было разработано веб-приложение на языке JavsScript с применением фреймворков Node.js и Express, реализующее рассмотренный метод оптимизации.

3 Тестирование разработанного программного решения

3.1 Проведение вычислительных экспериментов

Разработанное программное обеспечебние выдает результаты, которые выглядят вполне корректно. Однако на предприятиях с большими объемами, одна ошибка может обойтись большими денежными потерями, поэтому следующим этапом протестируем полученное программное обеспечение. При решении задачи оптимизации и подстановке ресурсов в систему ограничений, ни одно из них не должно быть нарушено. Для начала возьмём целевую функцию с тремя переменными (2).

$$f(x) = 2x_1 + 3x_2 + 4x_3 \quad (2)$$

где

f(x) – целевая функция,

x_1, x_2, x_3 – количество ресурсов первого, второго и третьего типа.

Введем следующие ограничения (3).

$$\begin{cases} x_1 + x_2 + x_3 \leq 50 \\ 2x_1 + x_2 + 1.5x_3 \leq 100. \\ x_1 + 2x_2 + x_3 \leq 150 \end{cases} \quad (3)$$

Для ввода этих данных откроем браузер и в адресную строку введем «http://localhost:5000», так как приложение работает на локальном хосте и имеет порт 5000. После чего нас перенаправит на стартовую страницу нашего приложения. На ней в соответствующих полях ввода укажем количество исходных переменных, количество уравнений ограничений и заполним матрицу ограничений(рисунок 17).

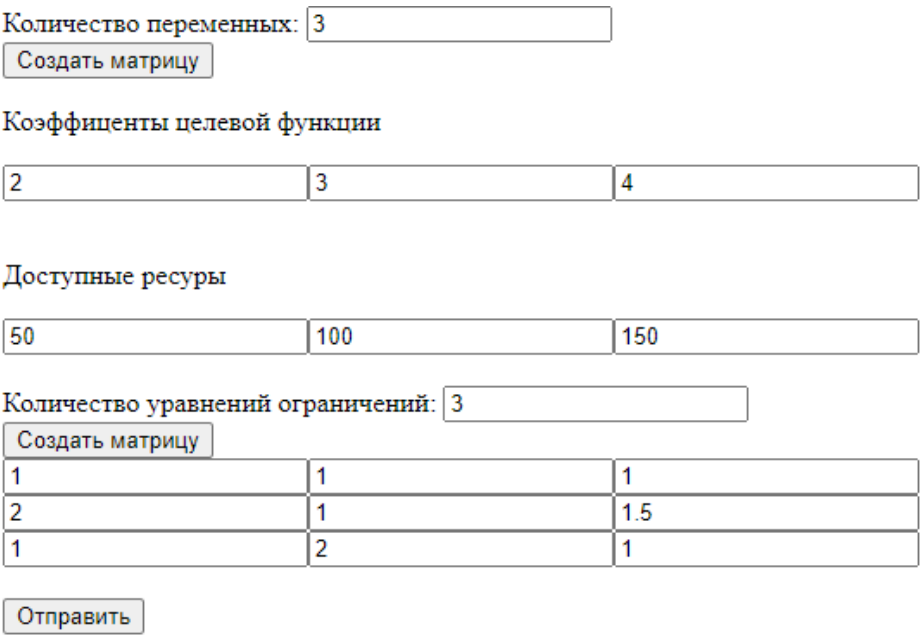


Рисунок 17 – Заполненная форма для отправки данных на сервер

После заполнения формы нажимаем кнопку «отправить». Страница с полученным результатом представлена на рисунке 18.

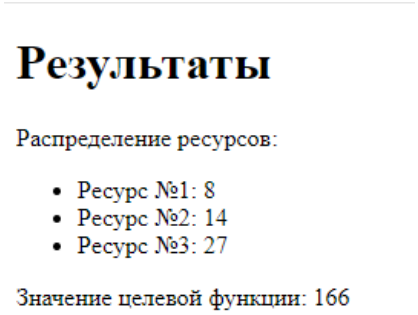


Рисунок 18 – Полученный результат

Проверим удовлетворяет ли данное решение ограничениям, записанным ранее по формулам (3).

Получаем следующие результаты: $8 + 14 + 27 = 49 \leq 50$, $2 * 8 + 14 + 1{,}5 * 27 = 70{,}5 \leq 100$, $8 + 2 * 14 + 27 = 63 \leq 150$. Как мы можем видеть, все три ненавистна выполняются, следовательно данное

решение подходит для данных ограничений.

Далее протестируем наше программное обеспечение с более большим объемом данных. Возьмем следующую целевую функцию (4):

$$f(x) = 5x_1 + 4x_2 + 6x_3 + 3x_4 + x_5 + 2x_6, \quad (4)$$

где

f(x) – целевая функция,

$x_1, x_2, x_3, x_4, x_5, x_6$ – количество ресурсов каждого типа.

И зададим новые ограничения (5).

$$\begin{cases} x_1 + 2x_2 + 3x_3 + 4x_4 + 5x_5 + 6x_6 \leq 220 \\ 2x_1 + 3x_2 + 3x_3 + 2x_4 + x_5 + 5x_6 \leq 180 \\ 3x_1 + x_2 + 5x_3 + 2x_4 + 3x_5 + 8x_6 \leq 195 \\ 2x_1 + 2x_2 + 3x_3 + 6x_4 + 2x_5 + x_6 \leq 206 \\ 6x_1 + 4x_2 + 5x_3 + 3x_4 + x_5 + 4x_6 \leq 200 \\ 4x_1 + 4x_2 + 2x_3 + 2x_4 + 3x_5 + 5x_6 \leq 187 \end{cases} \quad (5)$$

После чего введем исходные данные и посмотрим на результат (рисунки 19 и 20).

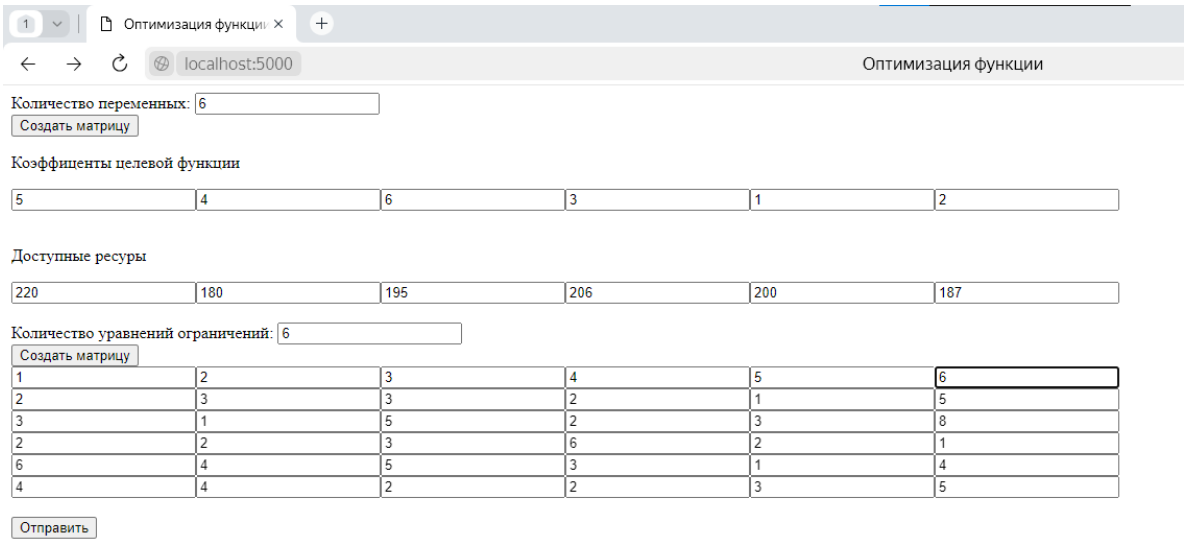


Рисунок 19 – Заполненная форма для отправки данных на сервер

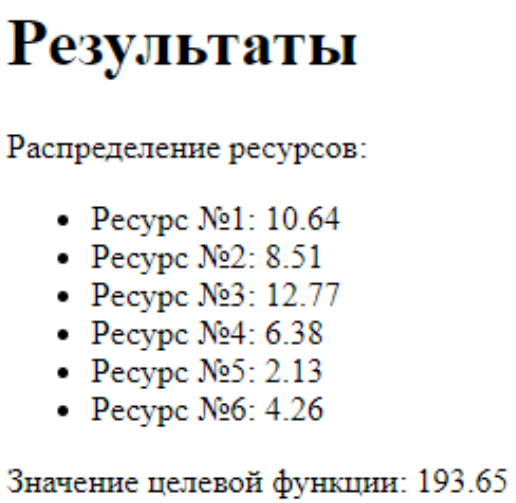


Рисунок 20 – Полученный результат

Проверим, выполняются ли ограничения:

– $10{,}64 + 2 * 8{,}51 + 3 * 12{,}77 + 4 * 6{,}38 + 5 * 2{,}13 + 6 * 4{,}26 \leq 220$,
– $2 * 10{,}64 + 3 * 8{,}51 + 3 * 12{,}77 + 2 * 6{,}38 + 2{,}13 + 5 * 4{,}26 \leq 180$,
– $3 * 10{,}64 + 8{,}51 + 5 * 12{,}77 + 2 * 6{,}38 + 3 * 2{,}13 + 8 * 4{,}26 \leq 195$,
– $2 * 10{,}64 + 2 * 8{,}51 + 3 * 12{,}77 + 6 * 6{,}38 + 2 * 2{,}13 + 4{,}26 \leq 206$,
– $6 * 10{,}64 + 4 * 8{,}51 + 5 * 12{,}77 + 3 * 6{,}38 + 2{,}13 + 4 * 4{,}26 \leq 200$,
– $4 * 10{,}64 + 4 * 8{,}51 + 2 * 12{,}77 + 2 * 6{,}38 + 3 * 2{,}13 + 5 * 4{,}26 \leq 187$.

Все условия выполняются, следовательно функция работает правильно.

3.2 Корректировка разработанного программного обеспечения

Разработанное программное обеспечение выполняет поставленные перед ним задачи корректно, однако внешний вид пользовательских страниц для ввода исходных данных и вывода результатов выглядит неоформленным и неэстетичным. Чтобы исправить ситуацию применим к ним стили CSS. CSS (Cascading Style Sheets) – язык таблиц стилей, который позволяет прикреплять стиль (например, шрифты и цвет) к структурированным документам [5][11]. Обычно CSS-стили используются для создания и изменения стиля элементов

веб-страниц и пользовательских интерфейсов, написанных на языках HTML и XHTML [6].

Для написания стилей CSS в папке views нашего проекта создадим файл «styles.css», который будет хранить в себе настройки для HTML-страниц [7].

Первым делом укажем свойства для объекта body. В качестве шрифта укажем Arial. Если указанный шрифт не будет доступен на компьютере пользователя, то будет использоваться шрифт без засечек. Далее укажем внешние и внутренние отступы для элемента равными 0, а также определим модель упаковки содержимого элемента так, что ширина и высота элемента, указанные в CSS, будут соответствовать размерам контента, а внешние размеры будут добавляться к указанной ширине и высоте [9]. Фон на странице укажем светло бежевый.

Затем опишем селектор для формы ввода данных. Так как фон на странице будет бежевый, то фон самой формы сделаем чисто белом. Чтобы форма не растягивалась на весь экран зададим максимальное значение 600px. Внутренние отступы вокруг содержимого элемента укажем равными 20px.Также, чтобы наша форма не была угловатой, зададим радиус скругления углов элемента равным 8px. Для более выраженного выделения добавим тень вокруг элемента с прозрачностью 0,1.

Следующим шагом опишем селектор для элементов ввода в матрице. Исходя из предыдущих страниц, поля для ввода чисел были очень широкими, поэтому зададим им значение равное 50px. Также зададим им границу шириной 1 пиксель с цветом более темным, чем цвет формы и добавим округлую форму. Первая часть файла «styles.css» изображена на рисунке 21.

Затем создадим стили для кнопок создания матрицы и отправки формы. Цвет кнопок сделаем синий, а текст на них белым. Также зададим свойству cursor значение «pointer», чтобы при наведении курсора на кнопку, указатель менял свой внешний вид. Затем для кнопок опишем селектор «hover», в котором зададим значение фона чуть темнее, чем сам цвет кнопок. Теперь при наведении на кнопки они меняют цвет. Точно также оформим страницу с

результатами вычислений.

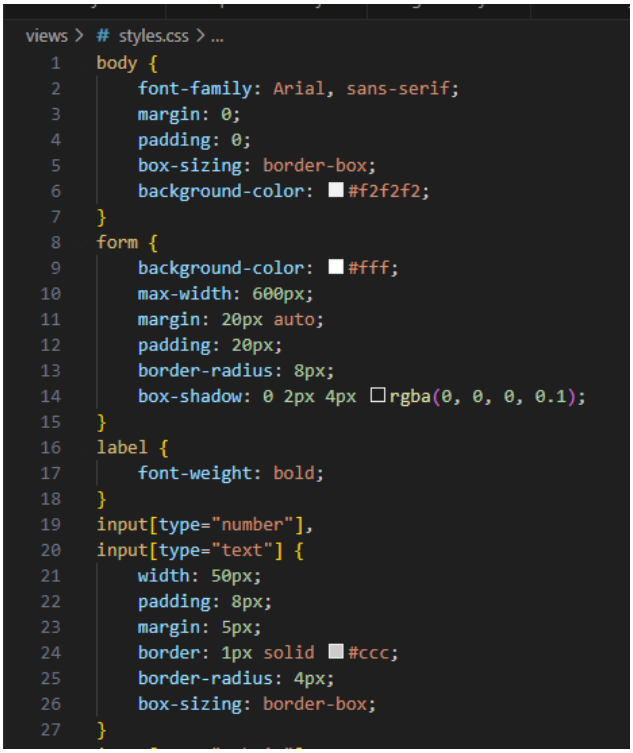


Рисунок 21 – Первая часть файла «styles.css»

Вторая часть файла «styles.css» изображена на рисунке 22.

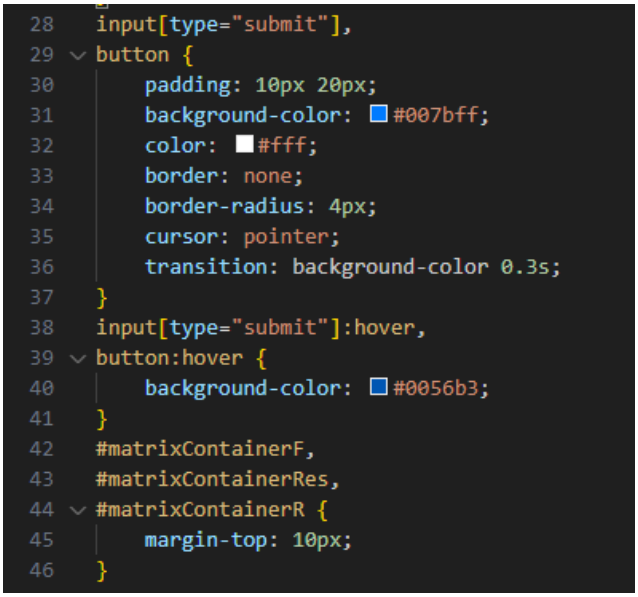


Рисунок 22 – Вторая часть файла «styles.css»

Полученные страницы представлены на рисунках 23 и 24.

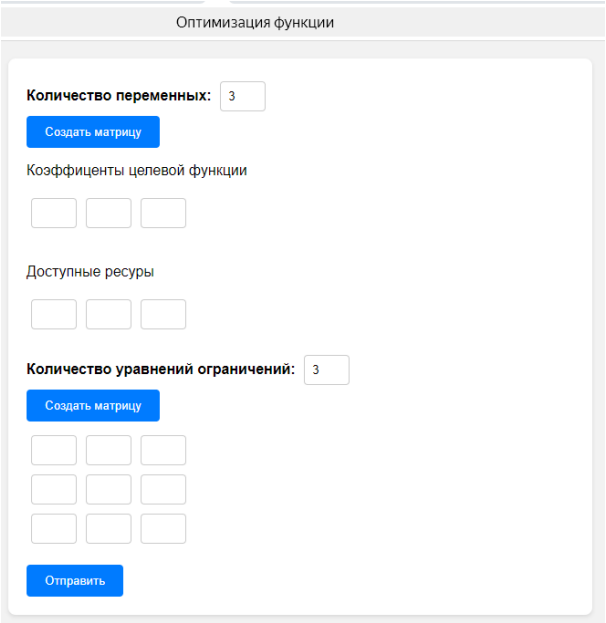


Рисунок 23 – Страница ввода данных

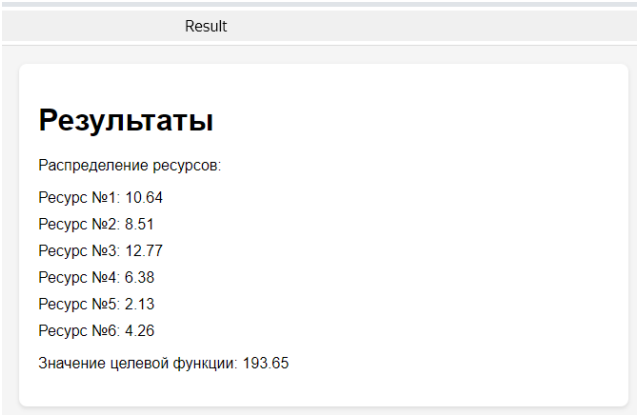


Рисунок 24 – Страница результатов

Вывод и результаты по третьему разделу:

В данном разделе было проведено тестирование разработанного программного обеспечения на разном наборе данных. Также был улучшен пользовательский графический интерфейс с помощью CSS стилей.

Заключение

В ходе данной работы было проведено исследование, которое показало необходимость создания программного обеспечения для оптимального распределения ресурсов на предприятии ООО «АвтоВАЗ» с целью получения наибольшей выгоды в условиях массового ухода поставщиков с рынка. Далее, в ходе анализа, был выбран один из четырех рассматриваемых методов оптимизации, а именно метод градиентного спуска. Также была выбрана модель REST API в качестве архитектурного стиля приложения, так как с ее помощью появляется возможность реализовать клиент-серверное приложения, что является важным фактором, так как предполагается, что использовать разработанный продукт будет с большим количеством данных, тем самым используя большие вычислительные мощности.

Следующим шагом для разработки приложения были выбраны такие фреймворки как Node.js и Express. Затем было разработано веб-приложение, серверная часть которого работает на локальном хосте и которое дает возможность вводить коэффициенты целевой функции и матрицу ограничений с вектором доступных ресурсов. Размеры данных матриц имеют динамический размер и задаются пользователем, что делает данное приложение адаптируемым под различные процессы производства. Результатом работы приложения является схема наилучшего распределение ресурсов между процессами, для получения максимальной прибыли, а также полученной значение целевой функции с данным распределением.

После чего разработанная программа была протестирована как с маленьким, так и с большим набором входных данных. Также для более удобного и приятного использования, HTML-страницы для ввода и вывода данных были улучшены при помощи CSS стилей, в результате чего они получили приятный внешний вид.

Список используемой литературы

1. Базаров, В. Ф., и Е. П. Гапоненко. "Методы оптимизации." Лаборатория базовых знаний, 2003.
2. Базаров, В. Ф., и Л. М. Мелас. "Оптимизация на множествах и в гильбертовом пространстве." Физматлит, 2006.
3. Гладких Б.А. Методы оптимизации и исследования операций. Часть 1. Введение в исследование операций. Линейное программирование. —Томск: НТЛ, 2009.
4. Гурвиц, В. В., и А. В. Фуджи. "Методы оптимизации." Наука, 1980.
5. Дакетт, Энтони Т. "HTML и CSS: разработка и дизайн веб-сайтов." ДМК Пресс, 2018.
6. Дакетт, Энтони Т., и Джеймс Шеффельд. "HTML и CSS. Путь к совершенству." ДМК Пресс, 2020.
7. Дъюкетт, Томас. "HTML5: Путеводитель по языку." Вильямс, 2013.
8. Кантелон, Хартер, Головайчук, Райлих. "Node.js в действии." Питер, 2014.
9. Лежнев, А.В. Динамическое программирование в экономических задачах [Текст]: учеб. пособие / Лежнев А.В. - Москва: Бином, 2010.
10. Львовский, С. М. "Методы оптимизации." Физматлит, 2004.
11. Макфарланд, Дэвид. "HTML5: разработка приложений для мобильных устройств." ДМК Пресс, 2013.
12. Нестеров, Ю. Е. "Методы выпуклой оптимизации." МЦНМО, 2010.
13. Окулов С.М. Динамическое программирование. – М.:Бином. Лаборатория знаний, 2017.
14. Симпсон, Кайл. "Вы не знаете JS: область видимости и замыкания." Символ-Плюс, 2016.
15. Стефанов, Стоян. "Шаблоны JavaScript." ДМК Пресс, 2012.
16. Стивен Скиена. Алгоритмы. Руководство по разработке. СПБ.: БХВ-Петербург, 2015.

17. Фланаган, Дэвид. "JavaScript. Подробное руководство." Вильямс, 2016.

18. Ширяв, В. И., "Исследование операций и численные методы оптимизации." Москва: Ленанд, 2017.

19. Bertsekas, D. P. Dynamic Programming and Optimal Control (4th ed.), Athena Scientific. 2017.

20. Cook T. & Russel R.A. Introduction to Management Science. Englewood Cliffs (New Jersey), Prentice Hall, Inc. 1989.

21. Crockford, Douglas. "JavaScript: The Good Parts." O'Reilly Media, 2008.

22. Haverbeke, Marijn. "Eloquent JavaScript: A Modern Introduction to Programming." No Starch Press, 3rd edition, 2018.

23. Heineman G.T. Algorithms in a nutshell. – O`Reilly, 2008. – 506 p.

24. Lawler E. Combinatorial optimization: Networks and Matroids. – Dover Publications Inc, 2003. – 374 p.

25. Winston W.L. Introduction to Mathematical Programming: Applications and Algorithms. Boston (Mass.): PWS-KENT Publ., 1991.