ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ»
(Н И У « Б е л Г У »)

ИНСТИТУТ ИНЖЕНЕРНЫХ ТЕХНОЛОГИЙ
И ЕСТЕСТВЕННЫХ НАУК

КАФЕДРА МАТЕМАТИЧЕСКОГО И ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ
ИНФОРМАЦИОННЫХ СИСТЕМ

ТОЧНОСТЬ ВЫЧИСЛЕНИЙ СИМПЛЕКС МЕТОДА ДЛЯ РЕШЕНИЯ ЗАДАЧИ ПЛАНИРОВАНИЯ ПРОИЗВОДСТВА

Выпускная квалификационная работа
обучающегося по направлению подготовки 02.04.01 Математика и компьютерные науки, группы 07001631
Иваницкого Алексей Валерьевича

Научный руководитель
д.т.н., профессор
Корсунов Н.И.

Рецензент
Кандидат технических наук
Инженер-исследователь центра высоких технологий
БГТУ им. В.Г. Шухова
И.Ш. Рахимбаев

БЕЛГОРОД 2018

Оглавление

Введение

Методы линейного программирования широко используются для оптимизации многих процессов деятельности. В качестве критериев эффективности выступают такие параметры как: оптимальное распределение ресурсов предприятия, минимизация расходов на производство того или иного вида продукции, максимизация доходов производства и т.д.

На сегодняшний день актуальным является применение методов линейного программирования в экономической сфере деятельности, так как использование математических моделей в задачах линейного программирования представляет собой важное направление по совершенствованию планирования и анализа деятельности любого предприятия. Решение оптимизационных задач линейного программирования позволяет выбирать оптимальные решения в условиях ограниченности ресурсов из совокупности альтернатив.

Таким образом, темой исследования является «Точность вычислений симплекс метода для решения задачи планирования производства».

Цель исследования: разработать математическую модель задачи планирования производством, реализовать ее решение симплекс-методом и определить точность вычислений. Создать программную реализацию для планирования производства.

Для реализации поставленной цели необходимо выполнить следующие задачи:

1. Изучить теоретические основы линейного программирования;

2. Изучить теоретические основы математического моделирования задач линейного программирования;

3. Рассмотреть методы решения задач линейного программирования;

4. Построить математическую модель задачи планирования производства;

5. Реализовать решение задачи планирования производства с использованием симплекс метода;

6. Проектирование и реализация программного продукта планирования производства;

7. Апробация системы.

Объект исследования - математическая модель задачи планирования производства.

Предмет исследования возможность реализации математической модели задачи планирования производства симплекс методом с заданной точностью вычислений.

В первой главе будут проанализированы и изучены история развития экономико-математического планирования производства, и необходимость решения задач планирования. Будет произведен обзор основных алгоритмов решения задач линейного программирования применительно к задачам планирования производства.

Во второй главе будет составлена задача планирования производства симплекс методом. Будет описано математическая модель и решение задачи. Также будет представлен алгоритм для реализации системы.

В третьей главе будет разработана система для планирования производства на основе дополненного симплекс метода. Также будет спроектирована архитектура системы, которая будет применена при реализации планирования производства.

В четвертой главе будет производится тестирование веб приложения.

Данная выпускная квалификационная работа содержит 64 страницы, 14 рисунка, 4 листинга кода и 1 приложение.

ГЛАВА 1. АНАЛИЗ ПРИМЕНЕНИЯ ЛИНЕЙНОГО ПРОГРАММИРОВАНИЯ ПРИ ПЛАНИРОВАНИИ ПРОИЗВОДСТВА

1.1 История развития экономико-математического планирования

В практической деятельности человека математика используется очень давно. Алгебра и геометрия служили основой для вычислений и измерений на протяжении многих веков. Несмотря на то, что развитие математики долгое время отвечало потребностям естественных наук и внутренней логике самой математики, использование математических методов в сфере экономики также имеет богатое прошлое.

В. Петти (1623-1687), который являлся родоначальником классической политической экономии, писал в предисловии своей работы «Политическая арифметика»: «...вместо того, чтобы употреблять слова только в сравнительной и превосходной степени и прибегать к умозрительным аргументам, я вступил на путь выражения своих мнений на языке чисел, весов и мер...». [1]

Французский ученый Ф. Кенэ (1694-1774) создал первую в мире модель народного хозяйства. В 1758 г. Ф. Кене публикует первый вариант своей работы «Экономическая таблица», которая получила название «Зигзаг»; второй вариант, под названием «Арифметическая формула» был опубликован в 1766 году. «Эта попытка, - писал К. Маркс о таблице Ф. Кенэ, - сделанная во второй трети XVIII века, в период детства политической экономии, была в высшей степени гениальной идеей, бесспорно, самой гениальной из всех, какие только выдвинула до сего времени политическая экономия».[2] «Экономическая таблица» Ф. Кенэ представляет схему (графико-числовую модель) процесса общественного воспроизводства, из которой он делает вывод, что нормальный ход общественного воспроизводства может

осуществляться только при соблюдении определенных оптимальных материально-вещественных пропорций.

На развитие методологии экономико-математических исследований большое влияние оказали труды К. Маркса. Работа К. Маркса «Капитал» содержит множество примеров, показывающих использование математических методов:

– обстоятельный параметрический анализ формулы средней прибыли;

– уравнения, которые связывают абсолютную, дифференциальную и суммарную ренту;

– соотношение стоимости и производительности труда в математической постановке;

– законы массы прибавочной стоимости и денежного обращения;

– условия формирования цены производства;

– и т.д.

Буржуазная экономическая наука XIX - XX исков выделяет три основных этапа развития экономико-математических исследований:

– математика в политэкономии (математическая школа);

– статистика;

– эконометрика.

Представителями математической школы было обосновано мнение, о том, что положения экономической теории можно обосновать только с помощью математических выводов, а все обоснования, которые получены другими способами, можно принять в лучшем случае как научные гипотезы. Родоначальник математической школы - французский ученый О. Курно (1801-1877), являющийся выдающимся математиком, философом, историком и экономистом, опубликовавший в 1838 г. книгу «Исследование математических принципов теории богатства».

Видным представителями математической школы были: Г. Госсен (1810-1858), Л. Вальрас (1834-1910), У. Джевонс (1835- 1882), Ф. Эджворт

(1845-1926), В. Парето (1848-1923), В. Дмитриев (1868-1913). Математическая школа представляет субъективистское направление буржуазной политэкономии, чьи идеологические и методологические принципы неоднократно критиковались учеными-марксистами. Вместе с тем, представители математической школы показали большие возможности применения методов математического моделирования. Представителями математической школы были выдвинуты и развиты важные теоретические подходы и принципы:

– введено понятие экономический оптимум;

– осуществлена возможность применения показателей затрат и предельных эффектов в рациональном хозяйствовании;

– показана взаимосвязь проблем ценообразования и общей пропорциональности народного хозяйства.

Современная экономическая наука оперирует понятиями представителей математической школы:

– кривые безразличия и ядро экономической системы (Ф. Эджворт);

– многоцелевой оптимум (В. Парето);

– модель общего экономического равновесия (Л. Вальраса);

– исчисление полных затрат труда и других ресурсов (В. Дмитриев).

Статистическое направление, которое возникло на пороге XX века, являлось прямой противоположностью математической школы с точки зрения методологии исследования. Использование эмпирического материала, конкретных экономических фактов являлось, несомненно, прогрессивным явлением. Идеологи статистической экономики, провозгласили тезис: «Наука есть измерение», впадали в другую крайность, при этом пренебрегали теоретическим анализом. Представители статистического направления разработали большое количество «математико-статистических моделей» экономических процессов и явлений, которые использовались в основном при краткосрочном прогнозировании. В качестве примера можно привести

«Гарвардский барометр» - является моделью прогнозирования хозяйственной конъюнктуры (предсказание «экономической погоды»), которая была разработана учеными Гарвардского университета (США) под руководством Т. Парсона (1902-1979). Гарвардская и другие модели подобного типа, были построены во многих капиталистических странах, и носили экстраполяционный характер. Такие модели не вскрывали глубинных факторов экономики. В соответствии с чем, на протяжении ряда лет они хоть и хорошо предсказывали «экономическую погоду», но «не заметили» приближения крупнейшего в истории капитализма экономического кризиса 1929-1932 гг. Крах Нью-Йоркской биржи в 1929 г. Одновременно означал и закат в экономико-математических исследованиях статистического направления.

В качестве заслуг статистического направления можно обозначить:

– разработку методических вопросов обработки экономических данных;

– статистические обобщения и статистический анализ (выравнивание рядов динамики и их экстраполяция, сезонные и циклические колебания, факторный анализ, корреляционный и регрессионный анализ, проверка статистических гипотез и т.д.).

Статистическое направление сменила эконометрика, пытавшаяся соединить достоинства математической школы и статистической экономики. Для обозначения нового направления в экономической науке был введен термин эконометрика (или эконометрия) норвежским ученым Р. Фришем (1895-1973), который провозгласил, что экономика является синтезом экономической теории, статистики и математики.

Эконометрика являлась наиболее быстро развивающейся областью экономической науки. В настоящее время применяются практически все теоретические и практические математические методы, и модели. В экономической науке Запада математическое моделирование являлось наиболее престижным направлением. Не случаен тот факт, что с момента

учреждения Нобелевских премий по экономике (1969 г.) их присуждение, как правило, назначается за исследования в области экономико-математических исследованиях. Среди Нобелевских лауреатов виднейшие эконометрики: Р. Фриш, Я. Тинберген, П. Самуэльсон, Д. Хис, В. Леонтьев, Т. Купманс, К. Эрроу.

Значительный вклад в развитие экономико-математических исследований был внесен русскими учеными. В журнале «Отечественные записки» в 1867 году была опубликована заметка об эффективности применения математических методов к изучению экономических явлений. Российские издания проводили критический анализ работ Курно, Вальраса, Парето и других западных экономистов-математиков.

С конца XIX века русские ученые В.К. Дмитриев, В.И. Борткевич, В.С. Войтинског, М. Оржнецког, В.В. Самсонов, Н.А. Столяров, Н.Н. Шапошников публикуют свои экономико-математические исследования.

Интересными являются работы по применению методов математической статистики, в частности по корреляционному анализу экономических явлений, А.А. Чупрова (1874-1926). Выдающимся математиком-экономистом дореволюционной России являлся В.К. Дмитриев (1868-1913). Первой известной работой В.К. Дмитриева становится работа «Теория ценности Д.Рикардо. Опыт органического синтеза трудовой ценности и теории предельной полезности» была опубликована в 1898 г. Основным трудом В.К. Дмитриева является работа «Экономические очерки», вышедшая в 1904 год. Она состояла в разработке модели полных затрат труда и сбалансированных цен в виде системы линейных уравнений с технологическими коэффициентами. «Формула В.К.Дмитриева» спустя несколько десятков лет нашла широкое применение в моделировании межотраслевых связей в СССР и за рубежом.

Широкую известность получили работы по теории вероятности и математической статистике Е.Е. Слуцкого (1880-1948). В 1915 г. Е.Е. Слуцкий в итальянском журнале «Giornale degli economisti e rivista di statistica», № 1

опубликовал статью «К теории сбалансированности бюджета потреби геля», которая оказала большое влияние на экономико-математическую теорию. Спустя 20 лет, данная статья получила мировое признание. Лауреат Нобелевской премии Д. Хикс в книге «Стоимость и капитал» (1939) писал, что Е.Е. Слуцкий первый экономист, сделавший значительный шаг вперед по сравнению с классиками математической школы. Д. Хикс свою книгу оценивал, как первое систематическое исследование теории, открытой Е.Е. Слуцким.

Английский экономист-математик Р. Аллен, который являлся автором книги «Математическая экономия», писал в журнале «Эконометрика», о работах Слуцкого, что они оказали «великое и прочное влияние на развитие эконометрики». Е.Е. Слуцкий явился одним из родоначальников праксеологии (науки о принципах рациональной деятельности людей) и первым, кто ввел праксеологию в экономическую науку.

Одним из первых советских специалистов в области экономико-математических исследований являлся А.А. Конюс, опубликовавший в 1924 году по данной теме статью «Проблема истинного индекса стоимости жизни».

Значительной вехой в истории экономико-математических исследований явилась разработка Г.А. Фельдманом (1884-1958) математических моделей экономического роста.

В 1938-1939 гг. ленинградский математик и экономист Л.В. Канторович в результате анализа ряда проблем организации и планирования производства сформулировал новый класс условно-экстремальных задач с ограничениями в виде неравенств и предложил методы их решения. Эта новая область прикладной математики позже получила название «линейное программирование». Л.В. Канторович (1912-1986) является одним из создателей теории оптимального планирования и управления народным хозяйством, теории оптимального использования сырьевых ресурсов. В 1975 году Л.В. Канторовичу совместно с американским ученым Т. Купмансом была присуждена Нобелевская премия за исследования по оптимальному

использованию ресурсов.

1.2 Необходимость решения задач линейного программирования

Необходимость решения задач линейного программирования обусловлена возможностью существенного улучшения качества планирования. Экономико-математические методы позволяют просчитать результат любого производственного процесса. Это связано с тем, что сфера применения этих методов в планировании очень велика и постоянно расширяется.

Однако на практике существует ряд трудностей, ограничивающих широкое применение экономико-математических методов в планировании. К таким трудностям можно отнести следующие:

– сложность, возникающая при определении критерия оптимальности в некоторых экономических задачах;

– сложность, возникающая при попытке внедрить математические модели в систему планирования и управления, которая существует на данный момент, что влечет необходимость создания новой технологии планирования и управления, которая бы базировалась на системном использовании экономико-математических методов и ЭВМ;

– стохастический и динамический характер планируемых процессов, для которого необходимо усложнить используемый математический аппарат и программное обеспечения ЭВМ, увеличивающий вычислительные объемы;

– сложность, возникающая при измерении многих экономических явлений и получение массовой достоверной информации для наполнения разработанных моделей;

– сложность, возникающая при проверке правильности (верификации) экономико-математических моделей, которые ориентированы не столько на

то, чтобы подтвердить действительность, сколько на то чтобы решить новые социально-экономические задачи (в первую очередь это модели планирования и прогнозирования), и т. д.

Основная сложность использования экономико-математических методов заключается в том, что достаточно трудно моделировать некоторые экономические процессы и явления. Сложность моделирования, прежде всего, связана с тем, что большинство экономических объектов можно охарактеризовать как «сложная система» и при изучении таких систем иногда практически невозможно использовать методы расщепления на элементы с тем, чтобы в последующем изучить их по отдельности. [3]

Усложняется возможность моделирования экономических систем еще тем, что сфера охвата экономической науки не ограничивается только рассмотрением производственных процессов, а рассматривает еще и производственные отношения [4]. В соответствии с чем, моделирование совокупности экономических процессов, с учетом производственных отношений, практически невозможно, не учитывая поведение людей, их интересы и индивидуально принятые решения.

Принятие плановых и управленческих решений в реалиях производственно-хозяйственных или социально-экономических ситуациях осуществляется по более сложным моделям, чем те, которые может предоставить экономико-математическое моделирование, однако планирование некоторых их процессов по отдельности осуществимо методами линейного программирования и широко применяется на практике.

Линейное программирование – один из наиболее широко распространённых аппаратов математической теории для оптимального принятия решений, включая и финансовую математику. В настоящее время для решения задач линейного программирования имеется большое количество прикладных программ, которые позволяют эффективно решать практические задачи экономических процессов и явлений. [6] Такое программное обеспечение снабжено системами обработки исходных данных,

возможностью проведения анализа данных и представления полученных результатов.

Владение аппаратом линейного программирования является необходимым для специалиста, решающего как управленческие задачи, так и задачи экономических процессов и явлений. [10]

Линейное программирование представляет собой наиболее часто используемый метод оптимизации. К числу задач, решаемых с использованием аппарата линейного программирования, относятся следующие:

– задача рационального использования сырья и материалов;
– задача оптимального раскроя;
– оптимизационная задача производственной программы предприятий;
– оптимизационная задача о размещении и концентрации производства;
– задача составления оптимального плана перевозок;
– оптимизационная задача управления запасами производства;
– и др. из сферы оптимального планирования.

Работу Л.В. Канторовича «Математические методы организации и планирования производства», опубликованную в 1939 г. Можно считать первым исследованием в области задач линейного программирования.

В данной работе Л.В. Канторович выполнил постановку задачи линейного программирования, разработал и дал теоретическое обоснование методу разрешающих множителей для решения задач линейного программирования.

Математическая формулировка проблемы поучения плана из возможных альтернатив, который бы являлся наиболее выгодным, с учетом ограниченности ресурсов называется прямой задачей линейного программирования.

Математическое программирование является прикладной отраслью математики, представляющей теоретическую основу для решения задач оптимального планирования.

Математическое программирование содержит следующие разделы:

– линейное;
– параметрическое;
– нелинейное;
– динамическое.

Раздел линейного программирования является наиболее разработанным и широко применяемым для нахождения оптимального значения (максимального или минимального) линейной функции, с учетом ограничений, представляющих собой систему линейных неравенств или уравнений.

В данной теории термин «программирование» понимается в смысле «планирования». Данный термин предложил в середине 1940-х годов Джордж Данциг, являющийся одним из основателей теории линейного программирования.

Проанализируем сущность задач оптимизации.

В рамках данной выпускной квалификационной работы термин оптимизация будем использовать в его экономическом смысле. Методы оптимизации используется в различных областях науки, в инженерных дисциплинах, математических, экономических и т.д. В соответствии с этим существует большое количество определений данного термина. Оптимизация, по сути, является процессом, приводящим ту или иную систему (экономическую, математическую) в оптимальное состояние. Задача оптимизации заключается в нахождении такого вектора решений целевой функции, при котором целевая функция принимает максимальное или минимальное значение, при этом существует ряд ограничений на использование ресурсов. В качестве целевой функции могут выступать издержки компании (задача нахождения минимума функции издержек),

прибыль компании (задача нахождения максимума функции прибыли) и т.д. В качестве ограничений могут использоваться финансовые, сырьевые, трудовые ресурсы компании, ограниченные по каким-либо причинам.

Для применения метода оптимизации к планированию, задачу оптимизации можно рассматривать как определение таких объемов производства, выполненных предприятием, при которых совокупный результат, выраженный в приросте целевых показателей, будет максимальным.

Таким образом, в процесс оптимизации планирования производственных объемов будут входить следующие составляющие.

Переменные - это величины, которые задают объемы производства по выпуску той или иной продукции.

Ограничения - это пределы, в которых варьируются переменные.

Целевая функция - описание зависимости результатов деятельности от объемов производства продукции.

В соответствии с методом целей и задач построение целевой функции чаще всего осуществляется на основании экспертных оценок, полученных для конкретно поставленной задачи. [12] При использовании аппарата математического моделирования для метода целей и задач, целевая функция представляется в виде модели, отображающей зависимость между переменными и результатом.

Вообще говоря, для того чтобы определить оптимальный план производства, необходимо построить целевую функцию, что в сфере экономических процессов представляет довольно сложный и трудоемкий процесс. Сложность построения целевой функции обусловлена трудностями в оценке эффективности мероприятий в их математической формализации. Исходя из цели, определяются направления производственной деятельности, с помощью которых данная цель может быть достигнута. Затем по каждому направлению производственной деятельности осуществляется поиск зависимости между выпуском объемом продукции и достижением результата.

Используя математическое моделирование, представляется возможным обобщить модель поведения рынка, учитывая в ней факторы воздействия на целевой показатель.

В общем виде процесс моделирования включает в себя четыре основные стадии:

1) Исследование и понимание законов функционирования конкретного рынка;

2) Выявление и оценка всех факторов, влияющих на уровень продаж (или долю рынка);

3) Разработка модели, включающая определение зависимостей между факторами и целевым значением функции, с учетом фактических данных;

4) Проверка модели и ее использование.

Очевидным является тот факт, что модели, разрабатываемые для определения объемов производства продукции, будут сильно разниться в зависимости от рынка, на котором работает компания и метода бюджетирования, который используется в компании.

Система ограничений, определяющая множество планов, диктуется условиями производства. Задачей линейного программирования (ЗЛП) является выбор из множества допустимых планов наиболее выгодного (оптимального). В общей постановке задача линейного программирования выглядит следующим образом:

Необходимо найти такие значения действительных переменных x_1, x_2,…, x_n ($x_k \geq 0$), при которых целевая функция (показатель эффективности), заданная формулой (1.1):

$$F(X) = c_1x_1 + c_2x_2 + \ldots + c_jx_j + \ldots + c_nx_n + c \qquad (1.1)$$

будет принимать экстремальное (максимальное или минимальное) значение при следующей системе ограничений (1.2):

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1j}x_j + \ldots + a_{1n}x_n \le b_1, \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2j}x_j + \ldots + a_{2n}x_n \le b_2, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{ij}x_j + \ldots + a_{in}x_n \le b_i, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mj}x_j + \ldots + a_{mn}x_n \le b_m, \end{cases} \quad (1.2)$$

где a_{ij}, b_i, c_j, c - заданные постоянные величины, $i = 1, 2, \ldots, m, j = 1, 2, \ldots, n,\ k \in \{1, 2, \ldots, n\}$. [2]

1.3. Обзор основных алгоритмов решения задач ЛП

1.3.1 Метод отсечений Гомори

Постановка целочисленной задачи линейного программирования.

Найти вектор $X = (x_1 \ldots, x_n)$, что минимизирует целевую функцию

$$L(X) = c_1x_1 + c_2x_2 + \ldots + c_jx_j + \ldots + c_nx_n \quad (1.3)$$

и удовлетворяет системе ограничений (1.4):

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1j}x_j + \ldots + a_{1n}x_n = a_{10}, \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2j}x_j + \ldots + a_{2n}x_n = a_{20}, \\ \ldots\ldots\ldots\ldots\ldots\ldots \\ a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{ij}x_j + \ldots + a_{in}x_n = a_{i0}, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mj}x_j + \ldots + a_{mn}x_n = a_{m0}, \end{cases} \quad (1.4)$$

где $x_j \ge 0, j = 1 \ldots, n$

x_j - целые, $j = 1 \ldots, n$. (1.5)

Метод Гомори

Метод Гомори является одним из методов отсечения, идея данного метода заключается в следующем.

Решается вспомогательная *ЗЛП* (1.3) - (1.4), которую получают из исходной *ЦЗЛП* (3)−(5) отбросив условие целочисленности переменных (1.5).

Если найденное оптимальное решение вспомогательной *ЗЛП* является целочисленным, то оно и будет и решением исходной *ЦЗЛП*. В противном случае, если полученное решение вспомогательной задачи не целочисленное, то от решенной *ЗЛП* необходимо перейти к новой вспомогательной *ЗЛП* присоединив условие линейного ограничения, удовлетворяющее целочисленности исходной *ЦЗЛП* и не удовлетворяющее полученному нецелочисленному решению вспомогательной *ЗЛП*. Данное дополнительное условие целочисленности определяет некоторую отрезающую плоскость и называется правильным ограничением. Присоединяем новые правильные ограничения к начальной вспомогательной *ЗЛП* до тех пор, пока на некотором шаге не будет получено целочисленное решение вспомогательной задачи, являющееся оптимальным решением исходной *ЦЗЛП*. Построение правильного ограничения в методе Гомори осуществляется следующим образом. [23]

Пусть на последней итерации симплекс-метода при решении вспомогательной *ЗЛП* непрямые ограничения этой задачи приобрели вид:

$$x_i + \mathrm{a}'_{i,m+1}\, x_{m+1} + \ldots + \mathrm{a}'_{in}\, x_n = \mathrm{a}'_{i0}, \qquad i = 1\ldots, m$$

и, таким образом, решением вспомогательной *ЗЛП* будет вектор

$$\boldsymbol{x} = (\mathrm{a}'_{10}\ldots, \mathrm{a}'_{m0}, 0, \ldots, 0).$$

Пусть существует номер r такой, что a'_{r0} - дробь, и, $\{\mathrm{z}\}$ - дробная часть z. Тогда правильное ограничение по методу Гомори задается неравенством:

$$\{\mathrm{a}'_{r,m+1}\}\, x_{m+1} + \ldots + \{\mathrm{a}'_{r,n}\}\, x_n \geq \{\mathrm{a}'_{r0}\}.$$

Алгоритм метода Гомори:

1. Решаем вспомогательную *ЗЛП* (1.3) - (1.4).

Пусть $x(0)$ является оптимальным решением вспомогательной задачи.

Если оптимальное решение не существует, то исходная *ЦЗЛП* также не имеет оптимального решения.

2. Пусть на $s-$ й итерации решена вспомогательная *ЗЛП*, которая имеет M ограничений и N переменных, вектор $x(s)$ является ее оптимальным решением.

Будем считать, что $x(s)$ определяется каноничными ограничениями последней итерации, то есть:

$$x_i + b_{i,M+1}\, x_{M+1} + \ldots + b_{iN}\, x_N = b_{i0}, \qquad i = 1\ldots, M$$

Откуда

$$\boldsymbol{x}\,(s) = (b_{10}\ldots, b_{M0}, 0, \ldots, 0\,).$$

3. Если b_{i0} $(i=1\ldots,M)$ – является целочисленным, то - конец, и x(s) является оптимальным решением исходной *ЦЗЛП*.

Если существует хотя бы одно нецелочисленное значение и такое, что b_{i0} - дробь, то переход к пункту 4.

4. $r = min\{i\}$ и строим дополнительное ограничение:

$$x_{N+1} - \{b_{r,M+1}\}\, x_{M+1} - \ldots - \{b_{rN}\}\, x_N = -\{b_{r0}\}$$

где $x_{N+1} \geq 0$ - дополнительная переменная.

5. Расширяем симплекс-таблицу за счет $(M+1)$ -ой строки (дополнительное ограничение) и $(N+1)$ -го столбца, что отвечает дополнительной переменной x_{N+1}.

6. Решаем расширенную, таким образом *ЗЛП* двойственным симплекс-методом и затем переходим к пункту 2 с заменой s на $s+1$. Если при этом на какой-либо итерации двойственного симплекс-метода одна из дополнительной переменной задачи повторно становится базисной, то она исключается из последующего рассмотрения, соответствующие ей строка и столбец, тоже исключаются.

1.3.2 Метод ветвей и границ

Метод ветвей и границ впервые предложили в своей работе американские математики Лэнд и Дойг в 1960г. Данный метод был применен к задаче линейного целочисленного программирования. На развитие целочисленного программирования по ряду определенных причин, заметного влияния эта работа не оказала. Второе рождение метод ветвей и границ получил в работе других американских математиков Литтла, Мурти и др. (Little, Murty, Sweeney и Karel), которая была посвящена задаче о коммивояжере (1963г.). В ней же впервые прозвучало название «Метод ветвей и границ», которое теперь является общепринятым.

Метод ветвей и границ используется в различных модификациях, объединенных общей идеей, заключающейся в замене полного перебора сокращенным, направленным перебором допустимых решений (комбинаций). Решение достигается за счет массового отсеивания так называемых «бесперспективных» альтернатив.

Различные модификации метода ветвей и границ строятся на общих принципах, которые подразумевают следующее:

1. Вычислить верхнюю или нижнюю границу (оценку) целевой функции на допустимом множестве или некотором его подмножестве;

2. Последовательно разбить допустимое множество (ветви), шаг за шагом сокращающая допустимое множество;

3. Пересчитать оценки (границы);

4. Установить признак оптимальности;

4' Произвести оценку точности приближенного решения.

Проанализируем каждый пункт более подробно.

1. Вычисление границы

Для определенности рассмотрим задачу комбинаторного типа на минимум целевой функции:

$$F(X) \to \min_{X \in G},$$

где G – множество допустимых решений (комбинаций);

X –элемент множества G.

Найдем нижнюю границу целевой функции F на множестве G или на некотором подмножестве $G' \subset G$.

Нижняя граница является числом $\eta(G)$ (или $\eta(G')$), таким что $\forall X \in G$ имеет место неравенство: $F(X) \geq \eta(G)$. Для вычисления нижней границы используются различные способы, которые зависят от содержательной постановки задачи.

2. Ветвление

Осуществляется постепенное разбиение допустимого множества G на некоторые подмножества меньшей мощности и представление этих подмножеств в виде дерева (рис. 1.1). Способ разбиения на подмножества зависит от постановки конкретной задачи.

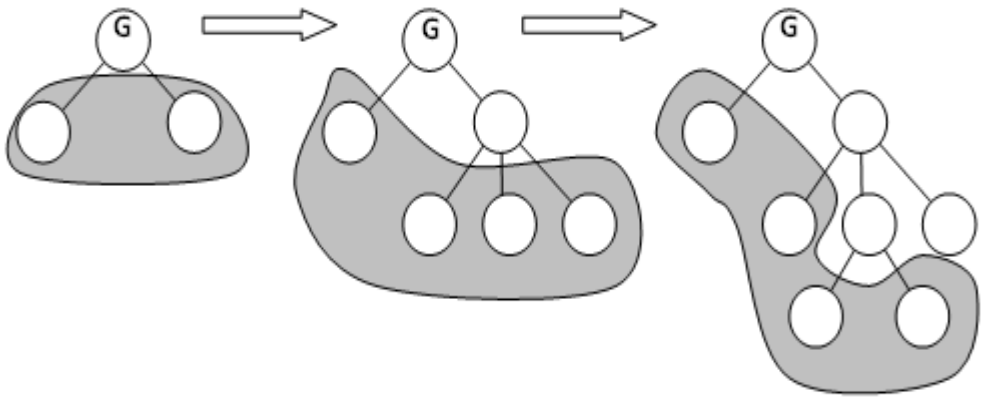

Рис. 1.1. Разбиение множества на подмножества и представление подмножеств в виде дерева

3. Пересчет оценок

Если $G1 \subset G2$, то имеет место следующее неравенство: $\min_{X \in G_1} \mathbf{F(X)} \geq \min_{X \in G_2} \mathbf{F(X)}$.

С учетом данного обстоятельства, предполагается, что, при разбиении некоторого множества $G' \subset G$ на подмножества $G'_1, G'_2, \ldots, G'_s$, причем G'= $\bigcup_{i=\overline{1,s}} \mathbf{G'_i}$, то оценка любого подмножества должна быть не меньше оценки подмножества G':

$$\boldsymbol{\eta(G'_i) \geq \eta(G')}, \quad i = \overline{1,s}.$$

После разбиения подмножества вычисляются оценки всех подмножеств, образовавшихся в результате разбиения, то есть уточняются оценки.

4. Признак оптимальности

Пусть G= $\bigcup_{i=\overline{1,s}} \mathbf{G_i}$ и известно решение $\overline{\mathbf{X}} \in \mathbf{G}_\nu$ ($\nu \in \{1,2,\ldots,s\}$). И имеет место соотношение: $\mathbf{F(\overline{X}) = \eta(G_\nu) \leq \min_{i=\overline{1,s}} \{\eta(G_i)\}}$

Тогда $\overline{\mathbf{X}}$ - является оптимальным решением задачи.

4'. Оценка точности приближенного решения

Пусть G= $\bigcup_{i=\overline{1,s}} \mathbf{G_i}$, и $\boldsymbol{\eta = \min_{i=\overline{1,s}} \{\eta(G_i)\}}$.

И пусть известно некоторое допустимое решение $\overline{\mathbf{X}} \in \mathbf{G_i}$. Тогда в соответствии с определением границы имеет место соотношение: $\boldsymbol{\eta \leq \min_{X \in G_i} F(X) \leq F(\overline{X})}$

Если значение $\Delta = \mathbf{F(\overline{X}) - \eta}$ невелико, то есть не превышает некоторого допустимого порога, выбранного для данной задачи, то $\overline{\mathbf{X}}$ принимается в качестве приближенного решения задачи, а Δ используется для оценки точности решения.

Проанализируем алгоритм решения задачи методом ветвей и границ.

Дана задача: $F(X) \to \min_{X \in G}$,

Введем понятие «рекорд», которое имеет принципиально важное значение для метода ветвей и границ.

Рекордом $R = F(X')$ называется наименьшее значение из всех известных, которое принимает целевая функция на допустимом решении $X' \in G$.

Шаг 1. Занесем множество G в список задач S: $S = \{G\}$. Полагая при этом $R = M$, где -является достаточно большим положительным числом.

Шаг 2. Для каждого подмножества списка вычислим нижние границы (оценки). При этом, оценки необходимо вычислить только для тех подмножеств, для которых они не были вычислены ранее.

Шаг 3. Последовательно проводится анализ подмножеств списка и анализ вычисленных для этих подмножеств оценок в порядке неубывания оценок.

При рассмотрении очередного подмножества списка $G*$, если $G* = \varnothing$, то оно удаляется из списка.

При $\eta(G*) \geq R$, то $G*$ также удаляем его из списка (как являющееся бесперспективным).

Если $G*$ является одноэлементным подмножеством, или устанавливается, что на некотором решении $X* \in G*$ имеет место $\eta(G*) = F(X*) < R$, то фиксируем новый рекорд: $R = F(X*)$.

В результате все бесперспективные подмножества удаляются из списка.

Шаг 4. Если список пустой ($S = \varnothing$), то конец и задача не имеет решения.

Шаг 5. Из списка выбираем подмножество с минимальной оценкой. После выполнения процедуры на шаге 3 оно является первым в списке подмножеством. Обозначение его установится как G^{min}.

Если G^{min} является одноэлементным подмножеством, или устанавлено, что на некотором решении $X^{min} \in G^{min}$ имеет место $\eta(G^{min}) = F(X^{min})$, то получаем оптимальное решение. Конец

Шаг 6. По признаку, который устанавливается заранее, подмножество G^{min} разбиваем на ряд новых подмножеств. Эти подмножества заносим в список S, после чего выполняем шаг 2.

Проанализируем метод Лэнд и Дойг для решения целочисленных и частично-целочисленных задач, относящийся к методу ветвей и границ.

Пусть дана задача линейного частично-целочисленного программирования:

$$\sum_{j=1}^{n} c_j x_j \to \max \tag{1.6}$$

$$\sum_{j=1}^{n} a_{ij} x_j = a_i, \; i = \overline{1,m} \tag{1.7}$$

$$x_j \geq 0, \; j = \overline{1,n} \tag{1.8}$$

$$x_j - \textit{целые}, \; j = \overline{1,n_1}, \; n_1 \leq n \tag{1.9}$$

Задача (1.6) - (1.8) является задачей линейного программирования, соответственно верхнюю границу удобнее всего вычислить, решив именно эту задачу. Действительно, имеет место неравенство: $Z^{opt}_{1,2,3} \geq Z^{opt}_{1,2,3,4}$

Вопрос вычисления границы решен.

В качестве рекорда возьмем допустимое решение задачи линейного целочисленного программирования, при котором целевая функция имеет значение, не меньше, чем любое известное на данный момент допустимое решение.

Шаг 1. Необходимо решать задачу линейного программирования (1.6) - (1.8). Если задача является неразрешимой, то и задача линейного целочисленного программирования (1.6) - (1.9) тоже неразрешима.

Полагаем $N = 1$ и в очищенный список задач S вставляется задача линейного программирования (1.6)-(1.8), обозначив ее 3_1. Сюда же заносится оптимальное решение задачи, принимаемое в качестве верхней границы $\boldsymbol{\eta}_1$.

Шаг 2. Имеется список задач S, где задачи упорядочены по не возрастанию оценок: $\mathbf{S} = \{3_1(\boldsymbol{\eta}_1), 3_2(\boldsymbol{\eta}_2), \ldots, 3_N(\boldsymbol{\eta}_N)\}$.

Если список пуст ($S = \varnothing$), то задача линейного целочисленного программирования не имеет решения. Конец.

В противном случае из списка выбирается задача $З_1$ - задача, которая имеет максимальную оценку.

Шаг 3. Если оптимальный план выбранной задачи $\mathbf{X_1}$ удовлетворяет требованию целочисленности (1.9), то $\mathbf{X_1}$ является оптимальным решением исходной задачи, $\boldsymbol{\eta}_1$ - оптимальное значение целевой функции. Конец.

Шаг 4. Выбираем нецелую координату решения $\mathbf{X_1}$. Пусть такой координатой является координата $x_{\mathbf{v}}$, которая имеет в решении $\mathbf{X_1}$ значение $\bar{x}_{\mathbf{v}}$. Далее сформированы и последовательно решается две задачи - $З_1^1$ и $З_1^2$:

$З_1^1$

$З_1$
$x_{\mathbf{v}} \geq [\bar{x}_{\mathbf{v}}]+1$

$З_1^2$

$З_1$
$x_{\mathbf{v}} \leq [\bar{x}_{\mathbf{v}}]$

Решая задачи $З_1^1$ и $З_1^2$ произведем следующие действия:

Если задача имеет решение, то ее вместе с оптимальным решением заносим в список задач S – в позицию, которая соответствует полученному значению целевой функции. Таким образом, все задачи в новом списке S’ упорядочиваются по не возрастанию оценок.

Шаг 5. Проводится сквозная перенумерация задач в новом списке. В соответствии с чем список приобретет вид:

$$\mathbf{S'} = \{З_1(\boldsymbol{\eta}_1), З_2(\boldsymbol{\eta}_2), \ldots, З_l(\boldsymbol{\eta}_l), \ldots, З_{N'}(\boldsymbol{\eta}_{N'})\}$$
$$\boldsymbol{\eta}_1 \geq \boldsymbol{\eta}_2 \geq \ldots \geq \boldsymbol{\eta}_l \geq \ldots \geq \boldsymbol{\eta}_{N'}$$

Из списка удаляем все бесперспективные задачи. При этом задача $З_r$ является бесперспективной, если в списке есть задача $З_l$, которая имеет целочисленное решение и $r > l$, то есть, $\boldsymbol{\eta}_l \geq \boldsymbol{\eta}_r$. Таким образом, сработает механизм рекордов: задача $З_l$, которая имеет целочисленное решение (если таковая есть), в новом списке займет последнюю позицию. Этой задаче будет

соответствовать рекорд.

В новом, списке, очищенном от бесперспективных задач будет N задач ($N \leq N'$). Принимается $S = S'$ и переход к шагу 2.

1.3.3 Симплекс метод

Симплексный метод — метод решения задач линейного программирования. Этот метод, изобретенный Джорджем Данцигом в 1947 году, последовательно проверяет соседние вершины допустимого множества (который является многогранником), так что в каждой новой вершине целевая функция улучшается или не изменяется. Одним из наиболее важных алгоритмов математической оптимизации является симплекс метод. Он был представлен Джорджем Данцигом. Это наиболее часто используемый метод решения задач линейного программирования, хотя его наихудшая сложность не является полиномиальной и до сих пор. Симплекс-метод очень эффективен на практике, как правило, занимает не более 2m до 3m итераций (где m - количество ограничений равенства) и сходится в ожидаемое многочленное время для определенных распределений случайных входов. Однако его наихудшая сложность является экспоненциальной, что можно продемонстрировать с помощью тщательно сконструированных примеров. [42]

Другим типом методов для задач линейного программирования являются методы внутренней точки, сложность которых полиномиальная как для среднего, так и для худшего случая. Эти методы строят последовательность строго выполнимых точек (т. Е. Лежащих внутри многогранника, но не на ее границе), которая сходится к решению. Исследования по методам внутренней точки были подкреплены статьей Кармаркара (1984). На практике одним из лучших методов внутренней точки является метод прогнозирования-корректора Mehrotra (1992), который является конкурентным с симплексным методом, особенно для

крупномасштабных задач. [1]

Симплексный метод Данцига не следует путать с методом простого спунга (Spendley 1962, Nelder and Mead 1965, Press et al., 1992). Последний метод решает проблему безусловной минимизации в n измерениях, сохраняя на каждой итерации n + 1 точки, которые определяют симплекс. На каждой итерации этот симплекс обновляется путем применения определенных преобразований к нему, чтобы он «катился вниз» до тех пор, пока не найдет минимум.

Линейная программа описывает некоторый многомерный многогранник, и мы ищем точку, которая максимизирует линейную функцию, которая геометрически означает, что она является самой дальней точкой в каком-то определенном направлении. На рисунке 1.2 представлено это в 2D формате.

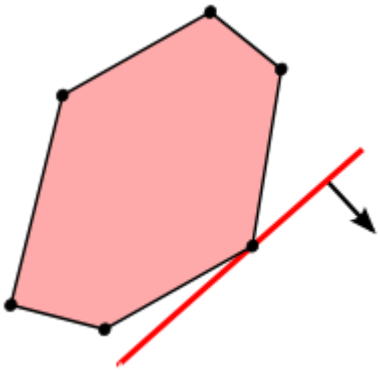

Рис. 1.2. Многомерный треугольник с линейной функцией.

Несмотря на эту простоту, линейные программы очень универсальны и не могут быть охарактеризованы огромным количеством проблем, хотя такие модели, вероятно, будут иметь тысячи переменных.

Прежде чем мы начнем решать линейную программу, нам нужно сначала взглянуть на свойства оптимального решения. Для любой ограниченной проблемы оптимизации есть две возможности, в которых оптимальным может быть:

- либо он находится где-то внутри области ограничений

- либо находится на его границе

Это понимание не удивительно. Но, учитывая линейную цель, мы не можем быстро найти оптимальное решение. Если даже ограничения линейны (что делает допустимую область полиэдром), мы получаем хорошее свойство, что всегда будет оптимальное решение на одном из углов многогранника. Могут быть и другие оптимальные точки (одна из сторон многогранника может иметь тот же угол, что и объект). [2]

Следующее понимание — это то, что мы находимся на углу многогранника, который не является оптимальным. Это связано с тем, что наш возможный регион выпуклый представленный на рисунке 1.3.

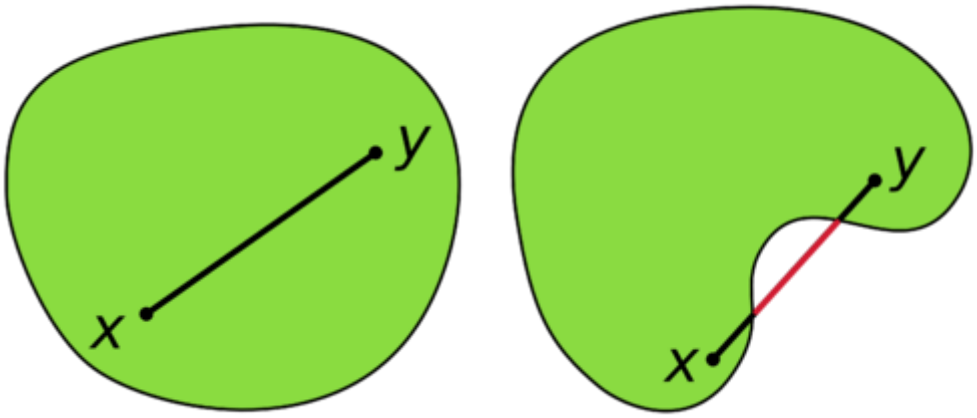

Рис. 1.3. Выпуклое и невыпуклое множество.

Теперь идея начать в каком-то углу многогранника. Затем идем по какой-то улучшающейся стороне, пока мы не сможем больше идти, - и там у нас есть оптимальный существует оптимальное решение.

Как это работает подробно? Сначала нам нужен способ описания решения. Поскольку это связано с некоторыми ограничениями, эти ограничения должны быть удовлетворены равенством.

Это утверждение полностью эквивалентно, оно просто переформулирует неравенства путем введения слабых переменных. Самое приятное, что имеем только равенства с большим количеством переменных, чем ограничения. Это должна быть система равенства. И такое решение

возможно, если никакая переменная не является отрицательной. Учитывая, что у нас есть m ограничений, мы можем указать угловую точку многогранника и выбрать его из соответствующей подматрицы, которую мы называем B (часто обозначаемой как базис).

Пусть поставлена задача: найти минимальное значение целевой функции:

$$F(X) = c_1x_1 + c_2x_2 + \ldots + c_jx_j + \ldots + c_nx_n + c \tag{1.10}$$

будет принимать экстремальное (максимальное или минимальное) значение при следующей системе ограничений (1.11):

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1j}x_j + \ldots + a_{1n}x_n \le b_1, \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2j}x_j + \ldots + a_{2n}x_n \le b_2, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{ij}x_j + \ldots + a_{in}x_n \le b_i, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mj}x_j + \ldots + a_{mn}x_n \le b_m, \end{cases} \tag{1.11}$$

где a_{ij}, b_i, c_j, c - заданные постоянные величины, $i = 1, 2, \ldots, m, j = 1, 2, \ldots, n,\ k \in \{1, 2, \ldots, n\}$. [2]

Проанализируем алгоритм симплекс-метода.

Шаг 1. Формулируем задачу линейного программирования в канонической форме основываясь на методе искусственного базиса, так чтобы матрица ограничений имела единичную базисную матрицу.

С этой целью дополним матрицу ограничений единичными столбцами, которые в совокупности с исходными столбцами матрицы ограничений обеспечивали бы существование единичной базисной матрицы. При этом естественным образом введем соответствующие искусственные переменные, которые включаются в целевую функцию с большими положительными

весовыми коэффициентами для задачи на минимум.

В результате исходная матрица A ограничений запишется в симплекс-таблицу (1.1), а коэффициенты целевой функции C запишутся в строку этой же симплекс-таблицы. В симплекс-таблицу 1.1 также включаются компоненты исходного базисного решения, определяемого вектором x_{B_0}.

Таблица 1.1

Исходная симплекс-таблица

№	Базисные столбцы	B_s	Базисное решение X_s	C_1	C_2	…	C_m	C_{m+1}	…	C_k	…	C_n
				A_1	A_2	…	A_m	A_{m+1}	…	A_k	…	A_n
1	A_1	c_1^s	x_1^s	1	0	…	0	$X_{1,m+1}^s$	…	$X_{1,k}^s$	…	$X_{1,n}^s$
2	A_2	c_2^s	x_2^s	0	1	…	0	$X_{2,m+1}^s$	…	$X_{2,k}^s$	…	$X_{2,n}^s$
…	…	…	…	…	…	…	…	…	…	…	…	…
l	A_l	c_l^s	x_l^s	0	0	…	0	$X_{l,m+1}^s$	…	$X_{l,k}^s$	…	$X_{l,n}^s$
…	…	…	…	…	…	…	…	…	…	…	…	…
m	A_m	c_m^s	x_m^s	0	0	…	1	$X_{m,m+1}^s$	…	$X_{m,k}^s$	…	$X_{m,n}^s$
			Оценки	∇_1	∇_2	…	∇_m	∇_{m+1}	…	∇_k	…	∇_n

Шаг 2. Вычисляются характеристические разности (оценок) по формулам и запишем оценки в $(m+1)$-ю строку симплекс-таблицы.

Шаг 3. Вычисление оценки v_k, которое удовлетворяет условию: если все $v_j > 0$, то в соответствии с выполнением критерия оптимальности вектор x^s является оптимальным решением, и далее переход к шагу 9, иначе - к шагу 4.

Шаг 4. Вычисляем новое базисное решение x_k^{s+1} из условия:

$$x_k^{s+1} = \min_{x_{ik}^s} \left[x_i^s / x_{ik}^s \right] = x_l^s / x_{lk}^s$$

Шаг 5. Вычисляем компоненты нового базисного решения x^{s+1} по формулам:

$$x_i^{s+1} = x_i^s - x_{ik}^s x_k^{s+1}, i \in B_{s+1}, i \neq k; x_i^{s+1} = 0,\ j.l \notin B_{s+1}$$

Шаг 6. Вычисление элементов новой симплекс-таблицы для $(s+1)$ -ой итерации метода по формулам:

$$x_{ij}^{s+1} = x_{ij}^s - x_{ik}^s x_{lj}^s / x_{lk}^s, i \neq l; x_{lj}^{s+1} = x_{lj}^s / x_{lk}^s$$

$$u_{ij}^{s+1} = u_{ij}^s - x_{ik}^s \frac{u_{lj}^s}{x_{lk}^s}, i \neq l; u_{lj}^{s+1} = \frac{u_{lj}^s}{x_{lk}^s}$$

Шаг 7. Корректировка симплекс-таблицы с учетом изменений коэффициентов целевой функции, соответствующих новому базисному решению. Формируется таблица 1.2.

Таблица 1.2

Симплекс-таблица

№	Базисные столбцы	B_{s+1}	Базисное решение X_{s+1}	C_1	C_2	…	C_m	C_{m+1}	…	C_k	…	C_n
				A_1	A_2	…	A_m	A_{m+1}	…	A_k	…	A_n
1	A_1	c_1^{s+1}	x_1^{s+1}	1	0	…	0	$X_{1,m+1}^s$	…	$X_{1,k}^s$	…	$X_{1,n}^s$
2	A_2	c_2^{s+1}	x_2^{s+1}	0	1	…	0	$X_{2,m+1}^s$	…	$X_{2,k}^s$	…	$X_{2,n}^s$
…	…	…	…	…	…	…	…	…	…	…	…	…
l	A_l	c_l^{s+1}	x_l^{s+1}	0	0	…	0	$X_{l,m+1}^s$	…	$X_{l,k}^s$	…	$X_{l,n}^s$
…	…	…	…	…	…	…	…	…	…	…	…	…
m	A_m	c_m^{s+1}	x_m^{s+1}	0	0	…	1	$X_{m,m+1}^s$	…	$X_{m,k}^s$	…	$X_{m,n}^s$
			Оценки	∇_1	∇_2	…	∇_m	∇_{m+1}	…	∇_k	…	∇_n

Шаг 8. Переходим к шагу 2.

Шаг 9. Остановка, получено оптимальное решение.

ГЛАВА 2. ПОСТАНОВКА ЗАДАЧИ ПЛАНИРОВАНИЯ ПРОИЗВОДСТВА СИМПЛЕКС МЕТОДОМ

2.1 Постановка задачи планирования производства в общем случае

Пусть некоторое предприятие осуществляет производство n видов продукции, при этом затрачивается m видов ресурсов.

Известны следующие параметры:

a_{ij} – количество i-го ресурса, которое необходимо для производства единицы j-й продукции;

$a_{ij} > 0$ $(i = 1, \dots, m;\ j = 1, \dots, n)$;

b_i - запас -го ресурса на предприятии, $b_i > 0$;

c_j-цена единицы -й продукции, $c_j > 0$.

Предполагается, что затраты ресурсов будут расти пропорционально росту объема производства.

Пусть x_j – является планируемым объемом производства -ой продукции. Тогда набор производимой продукции $x = (x_1,\ x_2, \dots, x_n)$ является допустимым, если при таком наборе суммарные затраты каждого вида -го ресурса не превосходят его запаса (2.1):

$$\sum_{j=1}^{n} a_{ij} \cdot x_j \le b_i; \quad i = 1,\dots, m. \tag{2.1}$$

Так же имеется следующее ограничение:

$$x_j \ge 0;\ j = 1, \dots, n. \tag{2.2}$$

Стоимость набора продукции x выражается величиной (2.3):

$$\sum_{j=1}^{n} c_j \cdot x_j . \quad (2.3)$$

Тогда постановка задачи планирования производства осуществляется следующим образом: среди всех векторов x, которые удовлетворяют ограничениям (2.1), (2.2), найти такой, при котором величина (2.3), будет принимать наибольшее значение.

2.2 Математическое описание поставленной задачи планирования симплекс методом

Одним из основных методов решения задач линейного программирования является симплекс-метод, с помощью которого ищется оптимальное базисное решение.

Пусть решается задача минимизации функции (2.4):

$$f = \sum_{j=1}^{n} c_j x_j \quad (2.4)$$

при ограничениях (2.5):

$$\begin{gathered} \sum_{i=1}^{m} a_{ij} x_j = b \\ i = \overline{1,r} \\ x_j \geq 0, i = \overline{1,n} \end{gathered} \quad (2.5)$$

,

Выбрав некоторый базис, например, для определенности (x_1, x_2,...,x_r), можно преобразовать задачу (2.4), (2.5) к виду:

$$\begin{cases} x_1 + q_{1,r+1}x_{r+1} + \cdots + q_{1,s}x_s + \cdots + q_{1,n}x_n = P_1 \\ x_2 + q_{2,r+1}x_{r+1} + \cdots + q_{2,s}x_s + \cdots + q_{2,n}x_n = P_2 \\ \cdots \\ x_r + q_{r,r+1}x_{r+1} + \cdots + q_{r,s}x_s + \cdots + q_{r,n}x_n = P_r \end{cases} \quad (2.6)$$

$$f + q_{0,r+1}x_{r+1} + \cdots + q_{0,s}x_s + \cdots + q_{0,n}x_n = P_0 \quad (2.7)$$

2.3 Решение поставленной задачи планирования производства

Схема решения задачи планирования производства симплекс-методом

Из коэффициентов уравнений системы (2.4) и выражения (2.5), которое также записано в виде линейного уравнения с правой частью, составляется таблица 2.1, которая называется симплекс-таблицей:

Таблица 2.1

Симплекс-таблица

Базис	Переменные										P
	x_1	x_2	....	x_j	....	x_r	x_{r+1}	....	x_s	....	x_n	
x_1	1	0	....	0	....	0	$q_{1(r+1)}$	....	q_{1S}	....	q_{1n}	P_1
x_2	0	1	....	0	....	0	$q_{2(r+1)}$	....	q_{2S}	....	q_{2n}	P_2
....	....	....	....	....	....	....	....	....	....	....	....	....
x_j	0	0	....	1	....	0	$q_{j(r+1)}$	....	q_{jS}	....	q_{jn}	P_j
....	....	....	....	....	....	....	....	....	....	....	....	....
x_r	0	0	....	0	....	1	$q_{r(r+1)}$	....	q_{rS}	....	q_{rn}	P_r
f	0	0	....	0	....	0	$q_{0(r+1)}$	....	q_{0S}	....	q_{0n}	P_0

Каждому базису соответствует своя симплекс-таблица. Фиксируется, что без учета последней строки столбцы коэффициентов при базисных неизвестных образуют единичную матрицу, а последний столбец дает нам

значения базисных переменных. Элемент же P_0 представляет собой значение целевой функции в базисном решении. Симплекс-таблицу удобно использовать для анализа и оценки соответствующего ей базиса.

Если в последнем столбце симплекс-таблицы нет отрицательных элементов, кроме, может быть, последнего (2.8):

$$P_i \geq 0, i = \overline{1,r} \tag{2.8}$$

то соответствующий этой симплекс-таблице базис допустим. Критерий оптимальности базиса для задачи минимизации формулируется следующим образом. Если в последней строке симплекс-таблицы нет положительных элементов, кроме, может быть, последнего (2.9):

$$q_{0s} \leq 0, s = \overline{r+1,n} \tag{2.9}$$

то соответствующий базис оптимален.

Действительно, последней строке симплекс-таблицы соответствует выражение (2.7), из которого можно получить соотношение (2.10):

$$f = P_0 - \left[q_{0,r+1}x_{r+1} + \cdots + q_{0,n}x_n\right] \tag{2.10}$$

В любом решении значения свободных переменных не могут быть отрицательными. Если для некоторого решения хотя бы одно q_{0s} положительно, то имеется возможность уменьшить значение f за счет роста соответствующего x_s. В случае, если выполняется соотношение (2.9), возможности уменьшить функцию S ниже зафиксированного значения P_0 нет. Сформулируем теперь критерий отсутствия оптимального решения.

Если в симплекс-таблице имеется столбец, последний элемент которого положителен, а все остальные элементы неположительные, то соответствующая задача линейного программирования не имеет

оптимального решения. Основным фактом является также то, что такой столбец может быть только столбцом коэффициентов при свободных переменных. Формально критерий записывается следующим образом: если имеется такое значение S $(r + l \leq S \leq n)$, что

$$q_{0s} > 0, q_{is}s \leq 0, i = \overline{1,r} \tag{2.11}$$

то

$$f_{min} = -\infty \tag{2.12}$$

Эти условия означают, что область допустимых решений системы (2.6) не ограничена. Действительно, проанализируем решение, в котором все свободные переменные, кроме x_s, равны нулю. В таком решении значения базисных переменных в силу (2.6) определяются по формулам (2.13):

$$x_i = P_0 - q_{0,s}x_s \ , i = \overline{1,r} \tag{2.13}$$

а значение целевой функции (25):

$$f = P_0 - q_{0,s}x_s \tag{2.14}$$

В силу (2.11) значение целевой может при росте x_s уменьшаться беспредельно, в то же время значения остальных переменных остаются в допустимой области.

Проанализируем теперь процесс перехода от некоторого фиксированного базиса (x_1, x_2,..., x_r) к новому базису. Такой переход необходим в случае, если не выполнены критерии оптимальности базисного решения. В этом случае в симплекс-таблице имеется столбец, соответствующий некоторой свободной переменной, в которой последний элемент положителен. Пусть этот столбец соответствует переменному x_s

$$q_{0s} > 0. \tag{2.15}$$

Пусть, кроме того, в s -ом столбце имеются и другие положительные элементы. Среди них выбираем тот, который стоит в строке j соответствующей следующему соотношению (2.16):

$$\frac{P_i}{q_{is}} = \underset{q_{is} > 0}{min} \left\{\frac{P_\lambda}{q_{\lambda s}}\right\} \tag{2.16}$$

Такой элемент q_{is} при высказанных предложениях всегда существует. Его называют разрешающим элементом. Столбец и отроку симплекс-таблицы, на пересечении которых находится разрешающий элемент, называют, соответственно, разрешающим столбцом и разрешающей строкой.

Сформулируем правило преобразования симплекс-таблицы с помощью разрешающего элемента:

1. Все элементы разрешающего столбца, кроме разрешающего элемента, заменяются нулями (2.17):

$$q'_{\lambda s} = 0,\ \lambda = \overline{0, r}, \lambda \neq i \tag{2.17}$$

Здесь и далее штрихом отмечены новые значения элементов, получающиеся после преобразования.

2. Все элементы разрешающей строки получаются делением на разрешающий элемент (2.18):

$$q'_{ik} = \frac{q_{ik}}{q_{is}},\ k = \overline{1, n} \tag{2.18}$$

3. Все остальные элементы симплекс-таблицы преобразуются по так называемому правилу прямоугольника (2.19):

$$q'_{\lambda k} = q_{\lambda k} - \frac{q_{\lambda k} q_{ik}}{q_{is}},\ \lambda = \overline{0,r}, \lambda \neq i \qquad (2.19)$$

Выделяется, что элементы столбцов, соответствующих переменным, входящим в старый и новый базис, при этом остаются неизменными и их просто надо переписать в прежнем виде.

4. Элементы последнего столбца преобразуются по формулам (2.20):

$$P'_i = \frac{P_i}{P_{is}}$$

$$P'_\lambda = P_\lambda - \frac{q_{\lambda s}}{q_{is}} \quad \lambda = \overline{0,r}, \lambda \neq i \qquad (2.20)$$

$$P'_0 = P_0 - \frac{q_{0s}}{q_{is}}$$

Также выделяется, что допустимость старого базиса и условие (2.16) обеспечивают допустимость нового базиса. Действительно, очевидно, что $P'_0 \geq 0$. Кроме того, рассматривая (2.20) при двух возможных предположениях, получаем:

а) если $q_{is} \leq 0$, тогда

$$P'_\lambda = P_\lambda - \frac{|q_{\lambda s}|}{q_{is}}, P_i \geq P_\lambda \geq 0$$

б) $q_{is} < 0$,тогда с учетом (31).

$$P'_\lambda = q_{\lambda s}\left(\frac{P_\lambda}{q_{\lambda s}} - \frac{P_i}{q_{is}}\right) \geq 0$$

Условие (2.19) и допустимость старого базиса обеспечивают невозрастание базисного значения целевой функции.

Действительно

$$\frac{q_{0s}}{q_{is}} P_i \geq 0$$

следовательно, из (2.12) следует, что $P_0' \leq P_0$. Необходимо отметить, что разрешающий элемент в данной симплекс-таблице может быть не единственный. В этом случае для преобразования симплекс-таблицы можно выбрать любой из них.

С учетом изложенного выше можно описать следующий метод решения основной задачи линейного программирования, называемый симплекс-методом.

Шаг 1. Отыскиваем какое-нибудь допустимое базисное решение. Обычно здесь используется специфика решаемой задачи. Формируем первую симплекс-таблицу, соответствующую выбранному базису.

Шаг 2. Анализируем полученный базис на оптимальность. Если среди элементов последней строки нет положительных элементов, кроме, может быть, последнего, то решение задачи найдено, все данные о нем содержатся в последнем столбце симплекс-таблицы. Если среди элементов последней строки есть положительные, то рассматриваются два исхода: а) если хотя бы над одним из положительных элементов последней отроки в столбце нет других положительных элементов, то задача не имеет решения;

б) в противном случае, переходим к третьему шагу.

Шаг 3. Выбираем разрешающий элемент и преобразуем симплекс-таблицу по приведенному выше правилу преобразования. Затем переходим к выполнению шага 2.

Симплекс-метод за конечное число шагов приводит к решению задачи, либо позволяет убедиться в его отсутствии.

ГЛАВА 3. РЕАЛИЗАЦИЯ ПРОГРАММЫ ДЛЯ РЕШЕНИЯ ЗАДАЧИ ПЛАНИРОВАНИЯ ПРОИЗВОДСТВА СИМПЛЕКС МЕТОДОМ

3.1 Выбор средств реализации

Java

Java - это высокоуровневый язык программирования и вычислительная платформа, разработанная Sun Microsystems в 1995 году. С тех пор язык регулярно обновляется с версией Java SE 10.0, являющейся последней версией, выпущенной в марте 2018 года.

Основываясь на преимуществах Java, он приобрел широкую популярность, и множество конфигураций были построены в соответствии с различными типами платформ, включая Java SE для Macintosh, Windows и UNIX, Java ME для мобильных приложений и Java EE для корпоративных приложений.

С ростом популярности веб-приложений и мобильных приложений Java сегодня является основой для большинства сетевых приложений и считается полезной для создания скриптов, веб-контента, корпоративного программного обеспечения, игр и мобильных приложений.

Приложения Java

Каждое предприятие использует Java так или иначе. Согласно Oracle, более 3 миллиардов устройств запускают приложения, разработанные на платформе разработки. Java используется для проектирования следующих приложений:

- Настольные графические интерфейсы;
- Встроенные системы;

- Веб-приложения, в том числе приложения электронной коммерции, электронные системы торговли на переднем и заднем офисах, системы расчетов и подтверждения, проекты обработки данных и многое другое;
- Веб-серверы и серверы приложений;
- Мобильные приложения, в том числе приложения для Android;
- Корпоративные приложения;
- Научные приложения;
- Продукты Middleware.

Преимущества Java

Java предлагает более высокую кросс-функциональность и переносимость, поскольку программы, написанные на одной платформе, могут работать на различных компьютерах, мобильных телефонах и встроенных системах.

Java является бесплатной, простой, объектно-ориентированной, распределенной, поддерживает многопоточность.

Java - это зрелый язык, поэтому он более стабилен и предсказуем. Библиотека классов Java обеспечивает кроссплатформенную разработку.

Будучи очень популярным на корпоративном, встроенном и сетевом уровне, Java имеет большое активное сообщество пользователей и поддержку.

В отличие от C и C ++, Java-программы скомпилированы независимо от платформы в байт-код, что позволяет одной и той же программе работать на любом компьютере, на котором установленном JVM.

Java имеет мощные средства разработки, такие как Eclipse SDK, NetBeans, Intellij IDEA которые имеют возможности отладки и предлагают интегрированную среду разработки.

Возрастающее языковое разнообразие, о чем свидетельствует совместимость Java с такими языками как Scala, Groovy, Kotlin, JRuby и Clojure.

Относительно бесшовная передовая совместимость от одной версии к следующей.

Выбор остановился на реализации программного продукта с помощью языка Java так как у него множество преимуществ, большая значимость среди всех языков программирования высокого уровня и он постоянно развивается.

Spring

В 2018 году большинство написанных систем на языке программирования Java не обходятся без такого программного продукта как Spring Framework - это платформа с открытым исходным кодом в Java для разработки корпоративных приложений. Модульность данного продукта, одна из важных особенностей которая позволяет решать множество задач не подключая множество сторонних продуктов.

Преимущества Spring Framework:

- Решение трудностей разработки корпоративных приложений

Spring решает задачи разработки сложных приложений, предлагая Spring Core, Spring IoC и Spring AOP для интеграции компонентов бизнес-приложений;

- Поддержка разработки корпоративных приложений через POJO (Plain Old Java Object)

Spring поддерживает разработку приложений Enterprise, используя классы POJO, которые устраняют необходимость импорта тяжелого контейнера Enterprise во время разработки. Это значительно упрощает тестирование приложений;

- Легкая интеграция других фреймворков

Spring предназначен для использования с другими платформами Java, такие как ORM, Struts, Hibernate и т.д. Spring framework не налагает никаких ограничений на структуры, которые будут использоваться вместе;

- Тестирование приложений

Spring Container может использоваться для разработки и запуска тестовых примеров, что значительно упрощает тестирование;

- Модульность

Spring framework - это модульная структура, и в нее входят множество модулей, таких как Spring MVC, Spring ORM, Spring JDBC, Spring Transactions и т.д., Которые могут использоваться в соответствии с требованиями приложения модульным способом;

- Управление срочными транзакциями

Интерфейс Spring Transaction Management очень гибкий, он может настроить локальные транзакции в небольшом приложении, которые можно масштабировать до JTA для глобальных транзакций.

Преимущества создания программного продукта на spring невозможно описать. Выше были описанные его преимущества и это только малая часть преимуществ Spring. Кроме Spring существуют фреймворки которые реализуют инверсию контроля над объектами такие как HiveMind, Avalon, PicoConteiner и т.д. Каждый из этих фреймворков подходит под реализацию определенной задачи. Наилучшем фреймворком для решения задачи планирования производства является Spring фреймворк.

Angular

Каковы основные преимущества, которые Angular и TypeScript могут предложить разработчикам? Это отличный вопрос, который нужно задать, прежде чем переходить на новые технологии или рамки. Услышав этот общий вопрос было решено, что пришло время собрать 5 лучших причин использования Angular и TypeScript:

- Производительность;
- Обслуживаемость;
- Модульность;
- Перехват ошибок.

Производительность

Разработчикам не нужно беспокоиться, если они делают что-то «правильно». Компоненты и службы выглядят одинаково, код многоразового использования помещается в классы обслуживания, модули ES6/ES2015

организуют связанные функции и позволяют коду быть автономным, данные передаются в компоненты с использованием свойств ввода и могут быть переданы для использования выходных свойств и т. д.

С последовательностью заранее определенных одинаковых действий вы получаете дополнительное преимущество производительности. Когда вы узнаете, как писать один компонент, вы можете написать другой, следуя тем же общим рекомендациям и структуре кода. Когда вы узнаете, как создать класс обслуживания, легко создать еще один.

Если используется TypeScript для создания Angular приложений, также получается несколько преимуществ производительности. В редакторах, таких как VS Code и Intellij IDEA, есть доступ к справочной системе (intellisense) показанной на рисунке 3.1 по мере ввода, что упрощает поиск типов и функций, которые предлагаются. Если используются интерфейсы TypeScript, то можно получить помощь от данных JSON, которые возвращаются из вызовов на внутреннюю службу. Это чрезвычайно полезно, когда различные объекты данных/модели используются и обрабатываются разработчиками. TypeScript, конечно, не только для Angular (можно использовать его с React, AngularJS, Vue.js, Node.js и любыми другими библиотеками/фреймворками JavaScript), но он очень хорошо интегрируется с Angular.

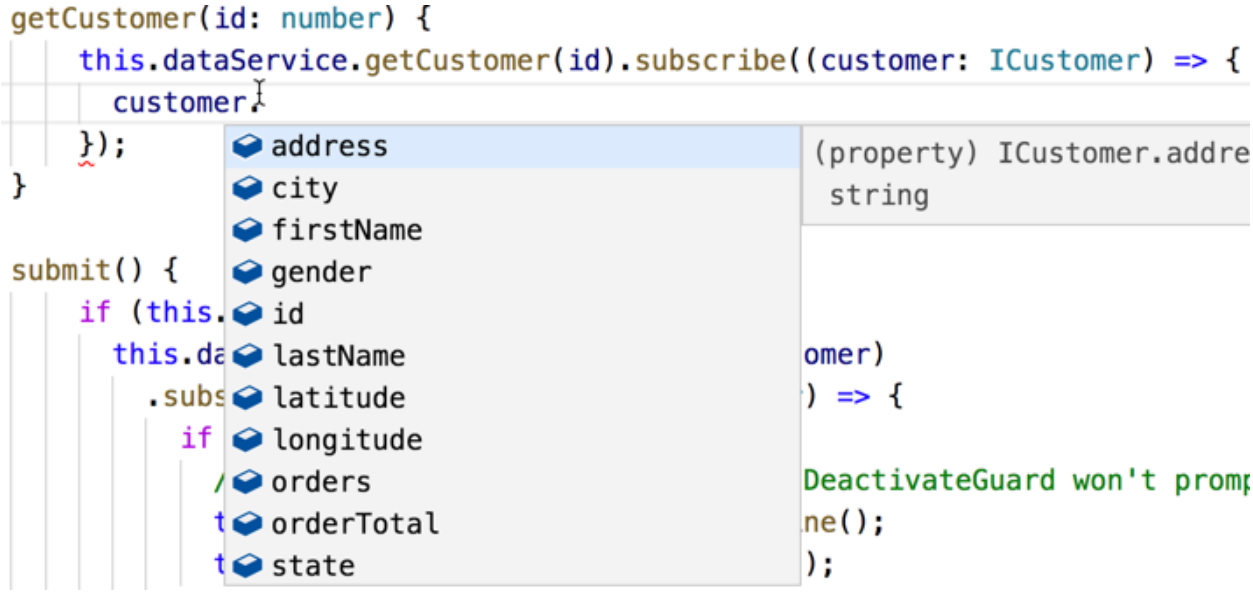

Рис. 3.1. Справочная система intellisense.

Обслуживаемость

Angular обеспечивает очень конкретный путь для написания кода, что в конечном итоге приводит к упрощенному обслуживанию системы. Благодаря руководству по стилю, знаниям, команда разработчиков может создать любую структуру и создать последовательный способ разработки приложений.

Код Angular может быть построен с использованием TypeScript который предоставляет множество преимуществ, особенно на предприятии.

Angular создает инфраструктуру, позволяющую легко поддерживать старый код, находить ошибки и исправлять их, и без особых затрат дописывать новый код.

Модульность

Приложения для предприятий могут быть достаточно большими, а способность разделить труд между несколькими членами команды, сохраняя организованный код, достижимо с помощью Angular. Все, что вы создается в Angular состоит из компонент, служб или директив объединятся в модули. Это позволяет избавится от «спагетти-кода» в приложении. Модули могут использоваться для стороннего использованиями организаций в собственных приложениях, подобно пакетам и пространствам имен в других языках/фреймворках, таких как Java или .NET.

Перехват ошибок

Преимещства предложений на TypeScript и Angular выражается в тестировании и обработке кода. Angular CLI создает процесс модульного тестирования и сквозного тестирования оснасткой (он полагается на Karma и Jasmine по умолчанию для модульных тестов, но можно использовать любые инструменты тестирования). Это, безусловно, еще одно из преимуществ, которое поможет отловить ошибки на ранней стадии разработки приложения.

В итоге после разбора различных фреймворков для создания приложения и рассмотрения их преимуществ был выбран в качестве реализации фреймворк Angular.

Liquibase

Liquibase - это инструмент управления изменениями базы данных с открытым исходным кодом, построенный на Java. Вместо того, чтобы писать SQL непосредственно в базе данных для создания, обновления или удаления объектов базы данных, разработчики определяют свои желаемые изменения в XML-файлах. XML-файл, называемый списком изменений, содержит список наборов изменений, которые определяют изменение базы данных в абстракции базы данных. Список изменений предназначен для того, чтобы хранить изменяющийся список изменений, которые команда хотела бы применить к базе данных. Пример кода, написанного в liquibase показан в листинге 3.1.

Листинг 3.1. Пример написание кода в liquibase.

```
<DatabaseChangeLog>
  <changelog id = "FOO-196-01" author = "Mike McGarr">
    <createTable tableName = "users">
      <column name = "id" type = "int">
        <ограничения primaryKey = "true" nullable = "false" />
      </ Столбец>
      <column name = "name" type = "varchar (100)">
        <constraints nullable = "false" />
      </ Столбец>
    </ CreateTable>
  </ Изменения>
</ DatabaseChangeLog>
```

Liquibase может выполняться либо через командную строку, либо как часть сборки с использованием Ant, Maven и т.д. Liquibase будет применять набор изменений непосредственно к базе данных и может обрабатывать откаты и теги состояния базы данных.

Сценарии Liquibase могут быть написаны в нескольких форматах, таких как JSON, XML, YAML и т.д., Которые не зависят от фактической базы данных. Это упрощает перенос приложения из одной реляционной базы данных в другую, когда Liquibase делает тяжелую работу для перевода сценариев Liquibase в SQL-запросы, специфичные для базы данных.

Liquibase состоит из наборов изменений, которые в основном представляют собой небольшой набор изменений, применяемых к базе данных. Liquibase отслеживает выполнение наборов изменений и записывает их в таблицу журналов изменений. Наборы изменений могут быть созданы из нескольких выпусков и упорядочены в последовательном порядке в файле журнала изменений. Это позволяет управлять версиями базы данных и поддерживать все сценарии в центральном расположении. Всякий раз, когда изменения необходимо применять к базе данных, мы можем создать новый набор изменений и добавить его в журнал изменений. Это позволяет иметь неизменный журнал всех изменений схемы базы данных и сводит к минимуму риск непреднамеренного применения неправильного патча во время установки приложения или миграции базы данных.

В итоге по реализации был выбран программный продукт Liquibase для создания и управления базой данных.

Git

Git - это система контроля версий для отслеживания изменений в компьютерных файлах и координации работы над этими файлами среди нескольких людей. Он в основном используется для управления исходным кодом в разработке программного обеспечения, но его можно использовать для отслеживания изменений в любом наборе файлов. В качестве

распределенной системы контроля версий она нацелена на скорость, целостность данных и поддержку распределенных нелинейных рабочих процессов.

Одним из самых больших преимуществ Git является его способность к ветвлению. В отличие от централизованных систем контроля версий, ветви Git легко объединяются. Это облегчает работу отраслевого рабочего процесса пользователей Git.

Использование ветвей функций не только более надежно, чем прямое редактирование производственного кода, но также обеспечивает организационные преимущества. Они позволяют представлять просмотреть детализацию по разработке. Например, можно реализовать политику, в которой каждое задание в Jira адресуется в свою собственную ветвь реализации.

Для разработчиков он устраняет все, начиная от времени, потраченного впустую, передает коммиты через сетевое соединение с человеческими часами, необходимыми для интеграции изменений в централизованную систему контроля версий. Это даже улучшает использование младших разработчиков, предоставляя им безопасную среду для работы.

Быть гибким - это правило по которому работает git, чтобы работать как можно быстрее.

3.2 Реализация программного продукта

Реализация программного продукта был выбран веб интерфейс. Но для начала работы необходимо написать API которое будет реализовывать алгоритм планирования производства. Так весь код пишется на языке программирования Java, будет использоваться объектно-ориентированный подход показанного в листинге 3.2.

Листинг 3.2. Использование объектов в программе.

```
//create new problem
SimplexProblem problem = new SimplexProblem();
problem.parse(contents);
//solve problem
SimplexSolver solver = new SimplexSolver();
Double result = solver.solve(p);
```

При реализации алгоритма был создан класс для работы с рациональными числами, позволяющий не делить числа, работая только с целыми числами и тем самым достигать результат точнее. Результат реализации можно посмотреть в приложении 1. После реализации алгоритма можно приступать к реализации интерфейса. На 2018 год с API на java можно создать такие интерфейсы как веб, android, ios и консольные интерфейсы. Для данного проекта был выбран веб. Программный продукт будет использоваться на предприятиях и заводах где необходимо рассчитать сколько и какой продукции необходимо производить чтобы получать большую прибыль. Чтобы сохранить корпоративные дынные был реализован вход и регистрация пользователей, представленном на рисунке 3.1, через Spring Security представленном в листинге 3.3. Каждый вход фиксируется в базе данных, а также фиксируются каждое действие пользователя с системой. Безопасность определяет качество приложения и уверенность что данные не попадут к конкурентам. Spring Security предоставляет базовые инструменты для реализации безопасности системы. Spring Security - это среда Java/Java EE, которая обеспечивает аутентификацию, авторизацию и другие функции безопасности для корпоративных приложений. Первым публичным выпуском под новым именем была Spring Security 2.0.0 в апреле 2008 года, с коммерческой поддержкой и обучением, доступными из SpringSource.

Листинг 3.3 Настройка spring security.

```
public class SecurityConfig extends WebSecurityConfigurerAdapter {
    @Autowired
    public void configureGlobal(AuthenticationManagerBuilder auth)
    protected void configure(HttpSecurity http) throws Exception {
      http.authorizeRequests();
    }
}
```

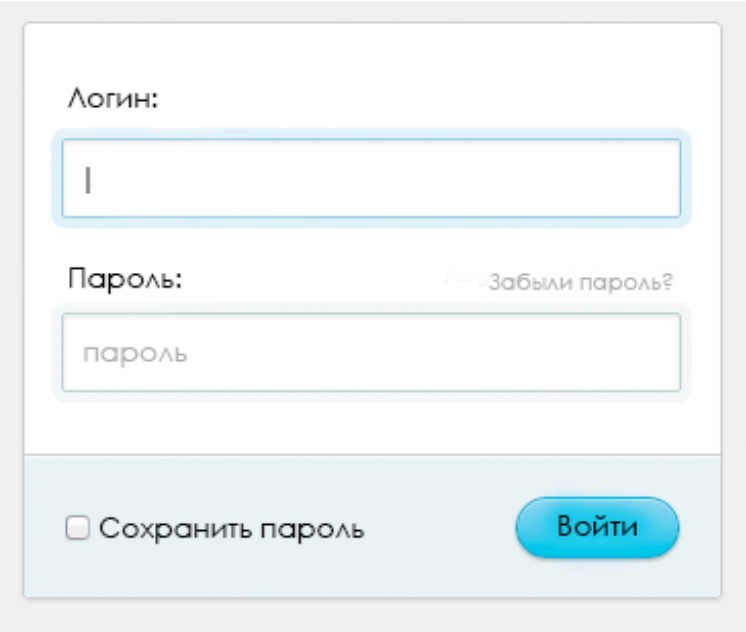

Рис. 3.1. Форма входа в приложение.

После входа в систему реализуем четыре окна которые можно изменять по размеру. В каждом из окон будет представлена своя реализация интерфейса, отображающая информацию для пользователя.

В первом окне реализация таблицы в которую вводиться информация для расчетов. Таблицу можно изменить и добавить, как продукция которые будут производится, как материал из которого производится продукция, представленная на рисунке 3.2. Таблица важнейшая часть системы из которой посылаются данные в API где они считаются. Так что реализация данной таблицы должна быть безупречной.

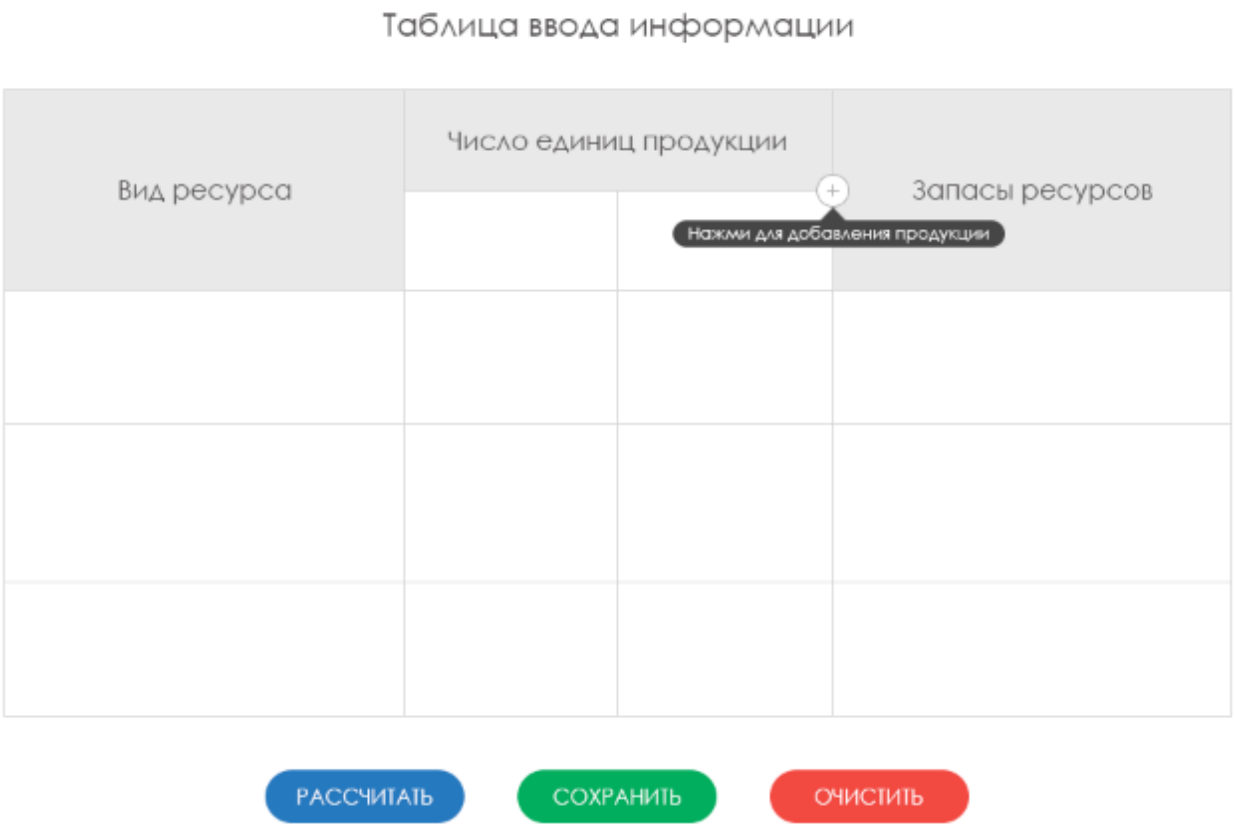

Рис. 3.2. Таблицы ввода информации для расчетов.

Особенность таблицы по изменению была сделана с помощью функций angular представленному в листинге 3.4.

Листинг 3.4. Функции для работы таблицы.

```
createRow(row: Row): Promise< Row > {
  return this.http.post(this.baseUrl + '/api /', Row)
    .toPromise().then(response => response.json() as Row)
    .catch(this.handleError);
}
updateRow (row: Row): Promise< Row > {
  return this.http.put(this.baseUrl + '/api/' + rowData.id, Row)
    .toPromise()
    .then(response => response.json() as Row)
    .catch(this.handleError);
}
```

Во втором окне реализуется визуализация результатов для лучшего понимания пользователя. Результаты преобразуются в график и отображаются пользователю. Примерный график представлен на рисунке 3.3.

Рис. 3.3. График результаты расчетов.

В третьем окне реализация таблицы в который хранятся прошлые запросы этого пользователя (см. рис. 3.4). Все данные автоматически сохраняются в эту таблицу и их можно посмотреть и проанализировать. У данный таблицы есть возможность поиска и фильтрации по дате создания.

Рис. 3.4. Таблица хранения результатов.

В четвертом окне выводится информация по решению, чтобы специалисты могли ее разобрать и проанализировать.

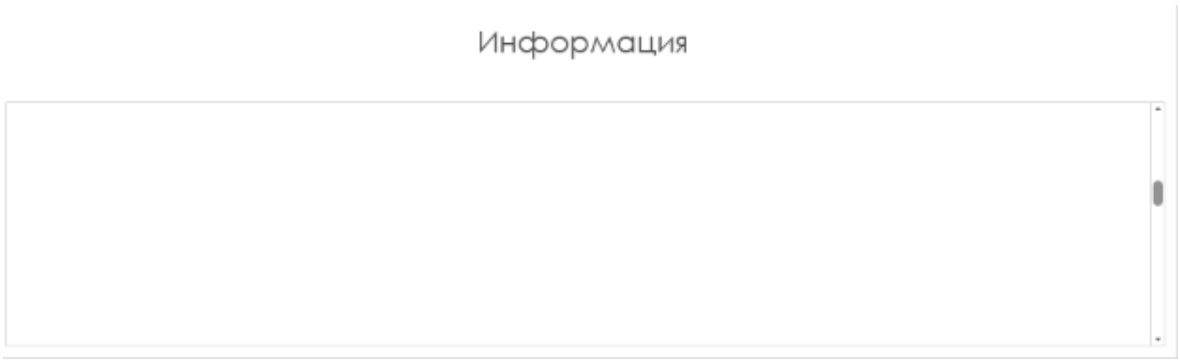

Рис. 3.5. Поле вывода математических расчетов.

Все эти четыре окна позволяют пользователям системы получать и работать с алгоритмом планирования производства. После того как работа в системе закончена, то необходимо отключится или закрыть вкладку. Кнопки для данных функций реализованы в программе.

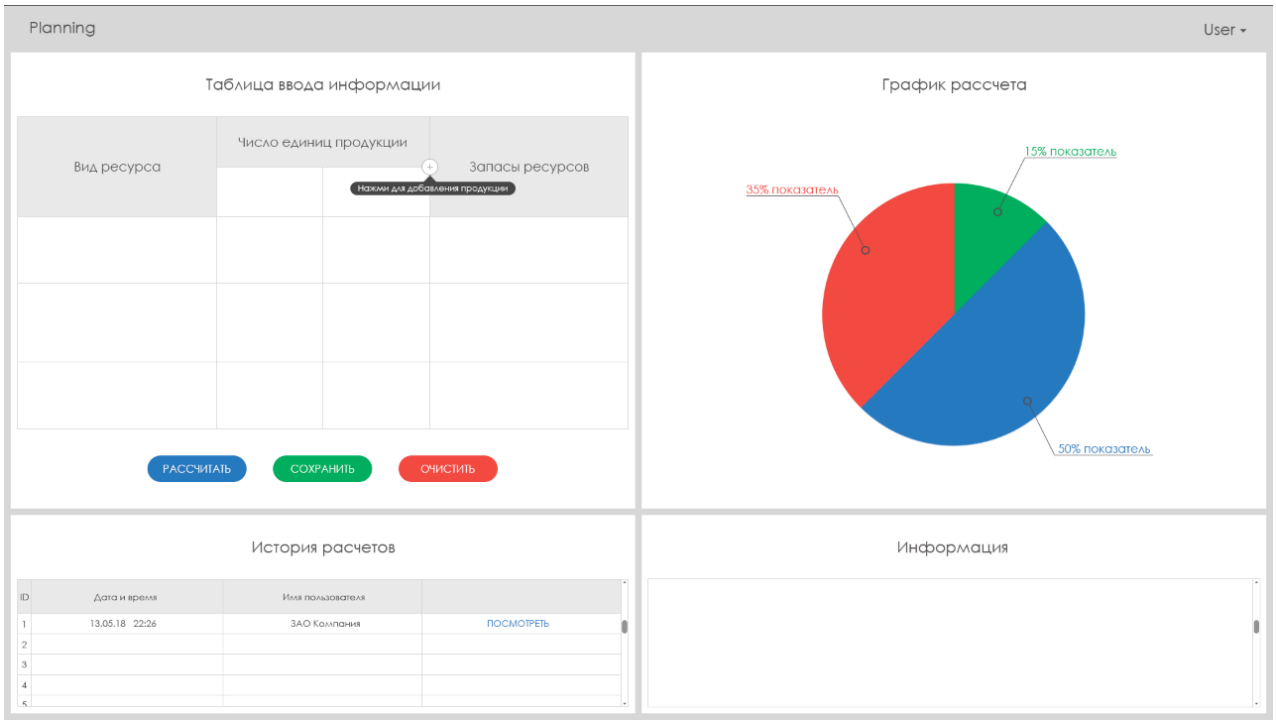

Рис. 3.5. Главная страница системы.

ГЛАВА 4. АПРОБАЦИЯ СИСТЕМЫ

Проверка системы одна из важнейших частей разработки программного продукта. Все большие проекты имеют своих тестировщиков, которые проверяют работоспособность и отказоустойчивость приложения. Это необходимо для проверки качества программного продукта и повышения репутации путем того что система отказоустойчива к различным ситуациям и конфиденциальные данные не утекут.

Для проверки, сравнения и тестирования приложения были взяты несколько систем реализации задачи планирования производства:

1. VisualData;
2. Siemens SIMATIC IT Preactor APS.

В данных системах реализовано в основном хранение и обработка информации. Планирование также в них возможно, но с учетом этого наши системы сильно отличаются и поэтому будут сравниваться успешность планирования производства.

Апробироваться работа приложения планирования производства будет с помощью тестов. После запуска система начинает с открытия окна логина, если пользователя нет в системе, то можно зарегистрироваться (см. рис 4.1).

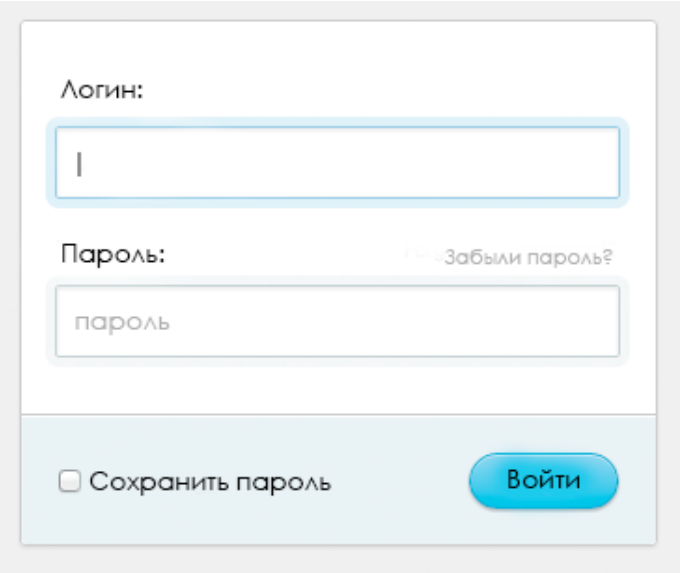

Рис 4.1. Форма логина системы.

Если пользователь ввел неправильные данные, то появится сообщение о том, что логин или пароль неверные (см. рис. 4.2).

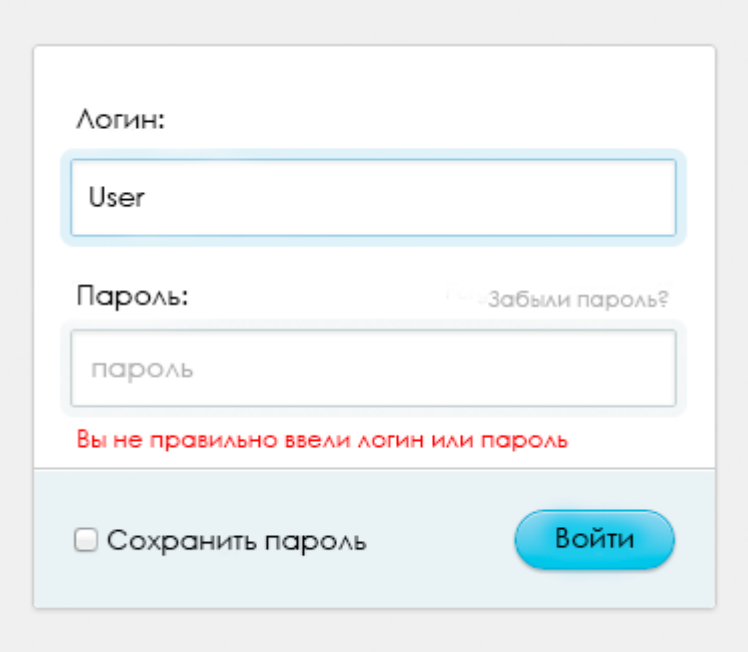

Рис 4.2. Форма логина с ошибкой входа.

При успешном входе в свой аккаунт перед пользователем открывается окна, в котором область разделена на четыре части (см. рис. 4.3). Это сделано для удобства работы. Каждая из них может меняться по размеру если двигать разделительные полосы.

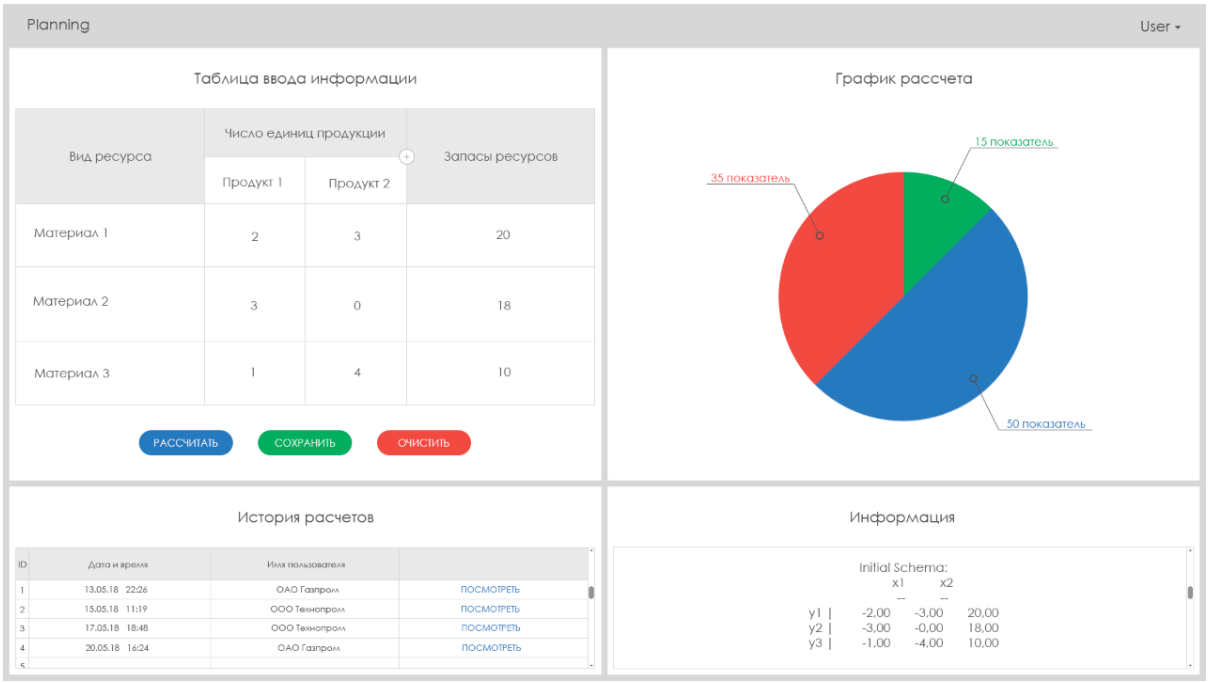

Рис 4.3. Полная форма системы.

В самом вверху отображается меня приложения и ник под которым зашел пользователь. Здесь отображается информация о пользователе и страницах. Это не так важно, как части самого приложения. В первом окне находится таблица, которую можно менять по размеру и изменять в ней данные, нажав два раза на ячейку таблицы. После того как заполнится таблица пользователю необходимо нажать кнопку отправить и все данные пройдут расчеты без какого-либо человеческого фактора. Все действия расчетов автоматизированы. После всех расчетов данные отображаются в остальные три части окна.

В графике отображается информация о продуктах, какие и сколько необходимо сделать для получения большей прибыли и меньших затрат.

Рис 4.4. График результаты расчетов.

Все результаты сохраняются в таблицу по который можно фильтровать данные. По нажатию на запись в таблицу, отображаются расчеты из абазы данных которые раннее производились. Это дает лучше проанализировать прошлые результаты, а не вводить данные по новой.

Рис 4.5. Форма истории расчетов.

В четвертом окне выводятся математические расчеты. Они выводятся в качестве текста, для специалистов.

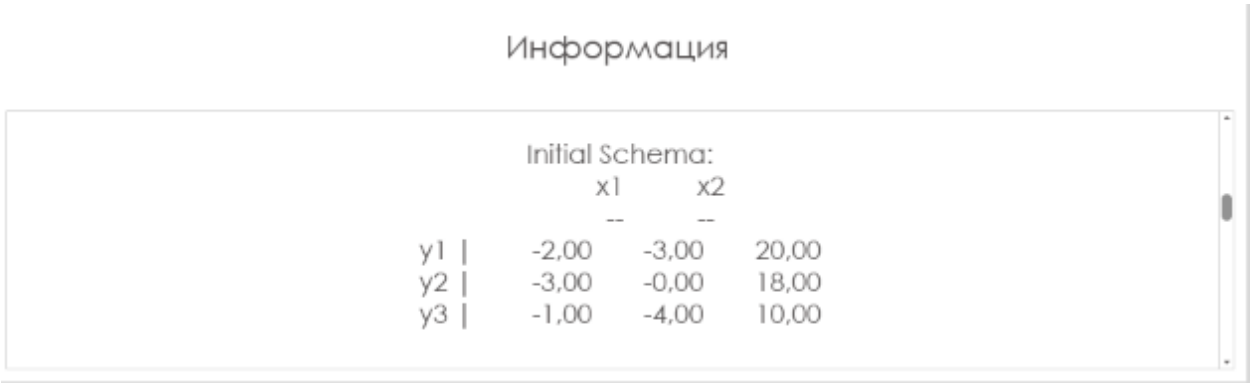

Рис 4.6. Форма вывода расчетов.

Подводя итогам апробации можно сделать выводы что, система успешно прошла возложенные испытания. При тестировании на большом количестве пользователей система показала высокую отказоустойчивость. Также были проведены тесты сравнения систем, которые показали, что реализованная система быстрее и на 1% точнее.

Заключение

В ходе выполнения магистерской диссертации была спроектирована и разработана система планирования производства продукции на основе доработанного симплекс метода для решения задачи планировании производства. Также в ходе были проведены испытания системы с другими приложения в данном области и показало, что разработанное приложение в ходе магистерской диссертации быстрее и показало, что точность системы была на 1% выше других. В результате чего были решены следующие задачи:

1. Изучены теоретические основы линейного программирования;
2. Изучены теоретические основы математического моделирования задач линейного программирования;
3. Проанализированы методы решения задач линейного программирования;
4. Построена математическая модель задачи планирования производства;
5. Реализовано решение задачи планирования производства с использованием симплекс метода;
6. Проектирование и реализация программного модуля задачи планирования производства;
7. Апробация системы планирования производства.

В первой главе были проанализированы и история развития экономико-математического планирования производства, и необходимость решения задач планирования. Был произведен обзор основных алгоритмов решения задач линейного программирования применительно к задачам планирования производства.

Во второй главе была составлена задача планирования производства симплекс методом. Был описано математическая модель и решение задачи. Также был представлен алгоритм для реализации системы.

В третьей главе была разработана система для планирования производства на основе дополненного симплекс метода. Также была спроектирована архитектура системы, которая была применена при реализации планирования производства.

В четвертой главе произведено тестирование веб приложения. Система показала продуктивные результаты.

Список использованной литературы

1. F.A. Ficken, Dover Books on Mathematics - The Simplex Method of Linear Programming (Dover Books on Mathematics), Paperback – June 17, 2015

2. Karl Heinz Borgwardt, Algorithms and Combinatorics (Book 1) - The Simplex Method: A Probabilistic Analysis (Algorithms and Combinatorics), Softcover reprint of the original 1st ed. 1987 Edition

3. Аверьянова С.Ю. Содержательные задачи линейного программирования и их решение с помощью ЭТ MS EXCEL и пакета MATHCAD: учебное пособие/С. Ю. Аверьянов, Н.В. Растеряев. - Южный федеральный университет. – Ростов-на-Дону: Издательство ЮФУ, 2014. – 132 с.

4. Ашихмин В.Н. и др. Введение в математическое моделирование, М: Логос, 2005г.

5. Ашманов С.А. Линейное программирование / С.А. Ашманов. - М.: Прогресс 2016. - 976 с.

6. Акулич И.Л., «Математическое программирование в примерах и задачах», Москва «Высшая школа» 1993г.

7. Бурда А.Г. Математическое моделирование процессов расширенного воспроизводства и вычислительное экспериментирование производственных параметров крестьянских (фермерских) хозяйств при различных нормах накопления / А.Г. Бурда, Е.А. Метельская // Программные системы и вычислительные методы. – 2013. – № 3. – С. 285– 294.

8. Бурда А.Г. Параметризация, моделирование и оптимизация эффективного использования производственного потенциала АПК Кубани / А.Г. Бурда, Г.П. Бурда // Труды Кубанского государственного аграрного университета. – 2015 – № 2.

9. Бурда А.Г. Экономико-математические методы и модели: учебное пособие (курс лекций)/ А.Г. Бурда, Г.П. Бурда. - Краснодар КубГАУ 2015. –

179 с.

10. Б.Банди, Основы линейного программирования. М: Высшая мат.,1989г.

11. Банди Б. Методы оптимизации. Вводный курс –М.. Радио и связь, 1988.-128 с.

12. Васильев Ф. П. Линейное программирование / Ф.П. Васильев, А.Ю. Иваницкий. - М.: Факториал Пресс, 2015. - 352 с.

13. Выгодчикова И.Ю. Введение в линейное программирование: учебное пособие./ И.Ю. Выгодчикова. - Саратов: Издательский центр «Наука», 2014.-47с.

14. Вентцель Е.С. Исследование операций: задачи, принципы, методологии. М.: Изд-во «Наука», 1980.

15. Введение в исследование операций. 6-е издание.: Пер. с англ. - М.: Издательский дом «Вильямс», 2001. - 912 с.: ил. - Парал. тит. англ.

16. Гаас С. Линейное программирование.- М… ГИМФМЛ, 1961-304 с.

17. Гилл Ф., Мюррей У., Райт М. Практическая оптимизация. – М.. Мир, 1985.- 512 с.

18. Гордон М.П., Тишкин Е.М. ,Усков Н.С. Как осуществить экономическую доставку товара отечественному и зарубежному покупателю. Москва. "Транспорт" 1993.

19. Ершов А.Т., Карандаев И.С., Шананин Н.А. Планирование производства и линейное программирование. МИУ, М., 1981.

20. Заславский Ю.Л. Сборник задач по линейному программированию.- М.. Наука, 1969.- 256.

21. Исследование операций в экономике: Учебное пособие для вузов/ Н.Ш. Кремер, Б.А. Путко, И.М. Тришин, М.Н. Фридман; Под ред. проф. Н.Ш. Кремера. - М.: ЮНИТИ, 2001. - 407 с.

22. Ильясов И.И. Система Эвристических приемов решения задач. -М.: РОУ 1992.

23. Конюховский П.В. Математические методы исследования операций

в экономике. - СПб: Питер, 2002. - 208 с.: ил. - (Серия "Краткий курс"). Раздел 4.2 Метод Гомори.

24. Кузнецов Ю.Н., Кузубов В.И., Волощенко А.Б. «Математическое программирование», Москва «Высшая школа» 1980г.

25. Кузнецов Б.Т, Математические методы и модели исследования операций. М: Юнити,2005г.

26. Калихман И.Л. Сборник задач по математическому программированию. Изд. 2-е, доп. И перераб. М., “Высшая школа”, 1975.-270 с.

27. Калихман И.Л. Сборник задач по линейной алгебре и программированию.- М.. Высшая школа, 1969.-160 с.

28. Линейное программирование. Ашманов С.А. - М.: Наука. Главная редакция физико-математической литературы, 1981. - 340 с.

29. Леоненков А. Решение задач оптимизации в среде MS Excel – СПб..БХВ- Петербург, 2005.- 704 с.. ил.

30. Маркс К., Энгельс Ф. Соч. Изд. 2-е, т.26, ч.1, с.345.

31. Малеко Е.М. Математические методы и модели исследования операций: Учеб. пособие. - Магнитогорск: МаГУ, 2003. - 100 с.

32. Математические методы в программировании: Учебник. - М.: ИД «ФОРУМ»: ИНФРА-М, 2006. - 224 с.: ил. - (Профессиональное образование).

33. Математическое программирование: Учебное пособие. - 2-е издание, переработанное и дополненное / Кузнецов Ю.Н., Кузубов В.И., Волощенко А.Б. - М.: Высшая школа, 1980. - 300 с.

34. Петти В. Экономические и статистические работы. М., Соцэкгиз, 1940, с. 156.

35. Партыко Т.Л., И.И. Попов. Математические методы, М: Форум, 2003г.

36. Схрейвер А. Теория линейного и целочисленного программирования: в 2-х т. Т.2: Пер с англ. - М.: Мир, 1991. - 342с., ил. Раздел Целочисленное линейное программирование.

37. Сдвинков О.А. математика в MS Excel 2002- М… Солон-Пресс, 2004-192 с.. ил.

38. Смирнов Э.А. Разработка Управленческих решений. - М.: ЮНИТИ, 2000.

39. Советский энциклопедический словарь / Гл. Ред. А.М.Прохоров. - 3-е изд. - М.: Советская энциклопедия, 1985.

40. Томас Х. Кормен и др. Глава 29. Линейное программирование // Алгоритмы: построение и анализ = INTRODUCTION TO ALGORITHMS. — 2-е изд. — М.: «Вильямс», 2006. — С. 1296. — ISBN 0-07-013151-1

41. Фалмер Р. М. Энциклопедия современного управления, т. 4 М.: Финансы и статистика, 1992.

42. Хемди А. Таха Глава 3. Симплекс-метод // Введение в исследование операций = Operations Research: An Introduction. — 7-е изд. — М.: «Вильямс», 2007. — С. 95-141. — ISBN 0-13-032374-8.

43. Шадрина Н.И. Решение задач оптимизации в Microsoft Excel 2010 : учеб. пособие / Н.И. Шадрина, Н.Д. Берман. – Хабаровск: Издательство Тихоокеан. гос. университета, 2016. – 101 с.

44. Шапкин А.С., Мазаева Н.П. Математичаские методы и модели исследования операций: Учебник.- М.. Издательско-торговая корпорация “Дашков и К°”, 2003.

45. Эрв Мате, Даниель Тиксье. Материально-техническое обеспечение деятельности предприятия. Москва. Прогресс. 1993

46. Эддоус М. И др. Методы принятия решений. - М.: Аудит, ЮНИТИ, 1997.

47. Юдин Д.Б. Задачи и методы линейного программирования. Задачи транспортного типа / Д.Б. Юдин, Е.Г. Гольштейн. - М.: Либроком, 2013. - 184 с.

ПРИЛОЖЕНИЕ 1

```
public class SimplexSolver {

    private double[][] schema;
    private int aimIndex;
    private int cIndex;
    private String[] head;
    private String[] side;

    public Double solve(SimplexProblem problem) {
        problem.convertEqualsConstraints();

        for(SimplexConstraint c : problem.getConstraints()) {
            if (c.getConstraintType() == ConstraintType.GreaterThanEquals)
                c.convertInequation();
        }

        if(problem.getOptimisationType() == OptimisationType.Min)
            problem.convertInequation();

        head = new String[problem.getCoefficients().length - 1];

        for (int i = 0; i < head.length; i++)
            head[i] = "x" + (i + 1);

        side = new String[problem.getConstraints().length];

        for (int i = 0; i < side.length; i++)
            side[i] = "y" + (i + 1);
```

```
        schema = new double[problem.getConstraints().length+1][problem.getCoefficients().length];

        aimIndex = schema.length-1;
        cIndex = schema[aimIndex].length-1;

        for(int y = 0; y < aimIndex; y++)
            schema[y] = problem.getConstraints()[y].getSlackVariables();

        schema[aimIndex] = problem.getSlackVariables();

        long maxSteps = getMaxIterations(problem.getCoefficients().length - 2, problem.getConstraints().length);

        printSchema("Initial Schema");

        int negCCount = getNegativeCCount();
        if(negCCount > 0) {
            dualAlgorithm(negCCount);
        }

        int stepCount = 0;
        while (nextStep(stepCount++)) {
            if(stepCount > maxSteps) {
                System.out.println("Cylcic Problem! Not solvable!");
                return null;
            }
        }
```

```
    System.out.println();

    for (int i = 0; i < head.length; i++) {
        String xName = "x" + (i + 1);
        int index = getIndexOf(xName);
        System.out.println(xName + "\t= " + schema[index][cIndex]);
    }

    if(negCCount > 0) {
        schema[aimIndex][cIndex] *= -1;
    }

    System.out.println("Result: " + schema[aimIndex][cIndex]);
    System.out.println();

    return schema[aimIndex][cIndex];
}

private int getNegativeCCount() {
    int c = 0;
    boolean[] negC = getNegativeC();
    for(int i = 0; i < negC.length; i++) {
        if (negC[i])
            c++;
    }
    return c;
}


private boolean[] getNegativeC() {
```

```
            boolean[] isNegative = new boolean[schema.length-1];
            for(int i = 0; i < schema.length - 1; i++) {
               isNegative[i] = schema[i][cIndex] < 0;
            }

            return isNegative;
         }

         private boolean nextStep(int stepCount) {
            MatrixPos aimFunctionPos = getPositiveAimFunctionComponent();
            if(aimFunctionPos == null)
               return false;
            System.out.println("Aim Column (" + aimFunctionPos.x + "): " +
schema[aimIndex][aimFunctionPos.x]);
            MatrixPos                    pivotIndex                    =
getPositionOfSmallestQuotient(aimFunctionPos.x);
            System.out.println("Pivot (" + pivotIndex.y + "|" + pivotIndex.x + "): " +
schema[pivotIndex.y][pivotIndex.x]);
            swap(pivotIndex);
            printSchema("Step " + stepCount);
            return true;
         }

         private void swap(MatrixPos pivotIndex) {
            schema[pivotIndex.y]      =      Equation.shift(schema[pivotIndex.y],
pivotIndex.x);
            swapVariableName(pivotIndex.y, pivotIndex.x);
            double[] values = schema[pivotIndex.y];
            for(int y = 0; y < schema.length; y++) {
               if(y != pivotIndex.y) {
```

```
            double[] eq = schema[y];
            schema[y] = Equation.plugIn(eq, values, pivotIndex.x);
        }
    }
}

private MatrixPos getPositionOfSmallestQuotient(int x) {
    double min = Double.MAX_VALUE;
    int bestY = -1;

    for(int y = 0; y < aimIndex; y++) {
        double aiq = schema[y][x];

        if (aiq >= 0)
            continue;

        double q = Math.abs(schema[y][cIndex] / aiq);
        if(q < min) {
            min = q;
            bestY = y;
        }
    }

    assert bestY != -1;
    return new MatrixPos(bestY, x);
}

private MatrixPos getPositiveAimFunctionComponent() {
    for(int x = 0; x < cIndex; x++) {
        if(schema[aimIndex][x] > 0)
```

```
            return new MatrixPos(aimIndex, x);
        }

        System.out.println("no new element found to switch!");
        return null;
    }

    private void printSchema() {
        printSchema("Schema");
    }

    private void swapVariableName(int s, int h) {
        String tmp = side[s];
        side[s] = head[h];
        head[h] = tmp;
    }

    private int getIndexOf(String varName) {
        for (int i = 0; i < aimIndex; i++)
            if (side[i].equals(varName))
                return i;

        return 0;
    }

    private void printSchema(String message) {
        System.out.println(message + ":");
        System.out.print("\t");
        for (int i = 0; i < head.length; i++)
            System.out.format("%12s", head[i]);
```

```
            System.out.println();

            System.out.print("\t");
            for (int i = 0; i < head.length; i++)
               System.out.format("%12s", "--");
            System.out.println();

            for(int y = 0; y < schema.length; y++) {
               if(y == schema.length-1)
                  System.out.print("z  | ");
               else
                  System.out.print(side[y] + " | ");

               for(int x = 0; x < schema[y].length; x++) {
                  System.out.format("%12.2f", schema[y][x]);
               }
               System.out.println();
            }
            System.out.println();
         }

         private void dualAlgorithm(int negCCount) {
            System.out.println("Problem can only be solved with dual algorithm!\n");
            double[] saveZ = schema[aimIndex].clone();
            int saveCIndex = cIndex;
            boolean[] negativeC = getNegativeC();
            int ci = 0;

            schema[aimIndex] = new double[schema[aimIndex].length];
```

```
for(int n = 0; n < negCCount; n++) {
   for (int i = 0; i < schema.length; i++) {
      schema[i] = SimplexUtil.insertAt((schema[i]), 0, 1);
      if(i == aimIndex) {
         head = SimplexUtil.insertAt(head, 0, "q"+n);
         cIndex++;
      }
   }
}

schema[aimIndex][0] = -1;
printSchema("Dual Schema");

for(int i = negativeC.length - 1; i >= 0; i--) {
   if(negativeC[i]) {
      swap(new MatrixPos(i,ci));
      ci++;
   }
}

printSchema("Pre Simplex Dual");
int stepCount = 0;
while (nextStep(stepCount++)){}
System.out.println("Dual algorithm stopped!");

for(int i = 0; i < head.length; i++) {
   if(head[i].startsWith("q")) {
      for(int j = 0; j < schema.length; j++) {
         schema[j] = SimplexUtil.removeAt(schema[j], i);
```

```
                }

                head = SimplexUtil.removeAt(head, i);
                i--;
            }
        }

        int indexX1 = getIndexOf("x1");
        schema[aimIndex] = Equation.plugIn(saveZ, schema[indexX1], 0);
        cIndex = saveCIndex;
        printSchema("Dual-Fixed Simplex Schema");
    }

    private long getMaxIterations(int varCount, int inequationCount) {
        int n = varCount + inequationCount;
        int k = inequationCount;

        return faculty(n) / (faculty(n-k) * faculty(k));
    }

    private long faculty(int n) {
        long m = 1;
        for (int i = 1; i <= n; i++) {
            m *= i;
        }
        return m;
    }
}
```