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

Институт математики, физики и информационных технологий
(наименование института полностью)

Кафедра «Прикладная математика и информатика»
(наименование кафедры)
01.03.02 Прикладная математика и информатика
(код и наименование направления подготовки, специальности)
Системное программирование и компьютерные технологии
(направленность (профиль)/специализация

БАКАЛАВРСКАЯ РАБОТА

на тему Реализация алгоритма решения задачи оптимального распределения ресурсов на основе метода динамического программирования

Студент(ка) И.В. Дубровин _____

Руководитель Н.А. Сосина _____

Консультант Е.В. Косс _____

Допустить к защите
Заведующий кафедрой к.т.н., доцент, А.В. Очеповский _____

«_____» _____ 20___ г.

Тольятти 2018

АННОТАЦИЯ

Тема выпускной квалификационной работы: «Реализация алгоритма решения задачи оптимального распределения ресурсов на основе метода динамического программирования»

Работа была выполнена студентом Тольяттинского Государственного Университета, института математики, физики и информационных технологий, группы ПМИб-1402, Дубровиным Ильей Владимировичем.

Выпускная квалификационная работа посвящена реализации алгоритма решения задачи оптимального распределения ресурсов.

Цель работы − обоснование метода динамического программирования для решения задачи распределения ресурсов, решение задачи распределения ресурсов аналитически и с помощью программных средств.

Объект исследования – задача оптимального распределения ресурсов на основе динамического программирования.

Предмет исследования – алгоритм решения задачи распределения ресурсов методом динамического программирования.

Задачи работы:

1) изучить общий подход динамического программирования;

2) разработать алгоритм для решения задачи распределения ресурсов;

3) выполнить программную реализацию разработанного алгоритма.

Актуальность работы заключается в решении проблемы эффективного распределения ресурсов, возникающей в области управления большими экономическими и производственными системами.

Результатом работы является программа решения задачи распределения ресурсов.

Бакалаврская работа содержит пояснительную записку объемом 43 страниц, включая 7 иллюстраций, 9 таблиц, список литературы из 21 наименований.

ABSTRACT

The title of the graduation work is «Developing algorithm based on dynamic programming for solving the optimal resources allocation problem».

The aim of this work is the use of the dynamic programming method for solving the resource allocation problem.

The object of the study is a mathematical model for solving the resources allocation problem.

The subject of the study is optimizing the resource allocation problem with the help of dynamic programming.

In the first part the theoretical basis of the method of dynamic programming, Bellman's principle of optimality and the main recurrence relations are discussed. The application of the method of dynamic programming in applied problems is also considered.

In the second part we outline the resource allocation problem. Match attention is given to an analytical solution of the described problem.

The third part is about the design and implementation of the algorithm and interface for a program that solves the resource allocation problem.

The result of the graduation work is the developed algorithm and the program that solves the resource allocation problem.

The graduation work consists of an explanatory note on 43 pages, introduction, three parts including 7 figures, 9 tables, the list of 21 references including 7 foreign sources.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Одной из важнейших задач управления экономическими и производственными системами является задача распределения ресурсов.

Ресурс является источником или запасом, из которого производится прибыль. Ресурсы обычно классифицируются на основе их доступности, а также возможности их возобновления. Обычно ресурсами являются материалы, энергия, услуги, персонал, знания или другие активы, которые преобразуются для получения прибыли предприятия и компании.

В экономике задача распределения ресурсов заключается в распределении доступных ресурсов для различных целей. В контексте макроэкономики ресурсы могут распределяться различными способами, в том числе, такими как рынки или централизованное планирование.

В управлении проектами распределение ресурсов или управление ресурсами - это планирование действий и ресурсов, востребованных в проектах с учетом их доступности.

Для решения задач оптимального распределения ресурсов используется алгоритм, основанный на методе динамического программирования.

Реализация данного алгоритма представляет научно-практический интерес.

Таким образом, **актуальность бакалаврской работы** обусловлена необходимостью решения задач оптимального распределения ресурсов, возникающих в управлении экономическими и производственными системами, на основе метода динамического программирования.

Объект исследования – задача оптимального распределения ресурсов на основе динамического программирования.

Предмет исследования – алгоритм решения задачи распределения ресурсов методом динамического программирования.

Целью выпускной квалификационной работы является обоснование метода динамического программирования для решения задачи распределения

ресурсов, решение задачи распределения ресурсов аналитически и с помощью программных средств.

Для достижения поставленной цели необходимо решить следующие задачи:

4) изучить общий подход динамического программирования;

5) разработать алгоритм для решения задачи распределения ресурсов;

6) выполнить программную реализацию разработанного алгоритма.

Выпускная квалификационная работа состоит из введения, трёх глав, заключения, списка используемых источников.

В главе 1 рассматривается общая постановка задачи динамического программирования. В главе 2 приводятся задачи распределения ресурсов и методы их решения. В главе 3 разрабатываются алгоритм и интерфейс программы. В заключении представлены результаты и выводы о выполненной работе.

ГЛАВА 1 МЕТОД ДИНАМИЧЕСКОГО ПРОГРАММИРОВАНИЯ

1.1 Общая постановка задачи динамического программирования

Динамическое программирование - это метод математической оптимизации и метод компьютерного программирования. Этот метод был разработан Ричардом Беллманом в 1950-х годах и нашел применение во многих областях: от аэрокосмической техники до экономики [3].

В динамическом программировании задача разделяется на подзадачи, и решения этих подзадач объединяются вместе для достижения общего решения основной задачи. При использовании подходов, таких как «разделяй и властвуй», подзадача может быть решена несколько раз. Динамическое программирование должно решить каждую из этих подзадач только один раз, тем самым уменьшив количество вычислений, а затем сохранит результат, избегая повторного вычисления на более позднем этапе, когда требуется решение для этой подзадачи.

Динамическое программирование используется для решения задач оптимизации (например, поиск кратчайшего пути), где может существовать множество решений, но интерес представляет только поиск оптимального решения для одного и того же (существует множество решений, которые достигают оптимального значения).

Эти проблемы должны иметь свойства перекрывающихся подзадач. Иными словами, решаемая задача может быть разбита на подзадачи, которые многократно используются, причем рекурсивный алгоритм решает одну и ту же подзадачу много раз, а не создает новую подзадачу. Например, числа Фибоначчи и оптимальная подструктура. В конечном итоге, оптимальное решение может быть построено из оптимальных решений подзадач.

Следует отметить, что во многих случаях алгоритмы перебора могут работать быстрее, чем алгоритмы динамического программирования, но они не гарантируют оптимальное решение задачи.

Если подзадачи могут быть вложены рекурсивно в более крупные задачи, так, что к ним могут быть применимы методы динамического

программирования, то существует связь между значением большей задачи и значениями подзадач. В литературе по оптимизации это соотношение называется уравнением Беллмана.

1.2 Уравнение Беллмана

Рассмотрим следующую постановку задачи.

$$V\ x\ = \sup_{y} F\ x, y\ + \beta V\ y\ , \tag{1}$$

где:

– $y\ \in U\ x\ ;$

– уравнение функции (1) называется уравнением Беллмана;

– x – переменная состояния;

– $G\ x\ =\ \ y \in U\ x\ : V\ x\ = F\ x, y\ + \beta V(y)$ – оптимальная политика. В ней представляются все значения y, которые достигают максимума в выражении (1);

– если $G(x)$ – единственное значение (уникальный оптимум), то G называется функцией политики.

Уравнение Беллмана впервые было применено к теории управления техническими системами и другим темам прикладной математики, а затем стало важным инструментом в экономической теории.

Большинство задач, которое может быть решено с использованием теории оптимального управления, также может быть решено путем анализа соответствующего уравнения Беллмана. Однако термин «уравнение Беллмана» обычно относится к уравнению динамического программирования, связанному с задачами оптимизации дискретного времени. В задачах непрерывной оптимизации аналогичное уравнение представляет собой уравнение в частных производных, которое обычно называют уравнением Гамильтона-Якоби-Беллмана.

1.3 Применение метода динамического программирования в решении прикладных задач

К особенностям математической модели динамического программирования относятся:

- формализация задачи оптимизации в качестве итогового многошагового процесса управления;
- определение критерия оптимальности операции или показателя эффективности целевой функцией, являющейся аддитивной от любого шага оптимизации;
- зависимость выбора управляющего воздействия x_k на каждом шаге исключительно от состояния самой системы на этом шаге с учетом отсутствия влияния на предшествующие шаги (отсутствие обратной связи);
- состояние системы S_k после каждого шага управления зависит исключительно от предшествующего состояния данной системы и управляющего воздействия x_k (нет последействия), причем оно может быть сохранено в виде уравнения состояния системы;
- зависимость управления x_k на каждом шаге от конечного числа управляющих переменных, а состояния системы S_k - от конечного числа параметров;
- определение оптимального управления (вектора X) как последовательности оптимальных управлений, число которых устанавливает количество шагов задачи.

Основные свойства задач, в которых можно применять метод динамического программирования:

- задача должна допускать интерпретацию как многошаговый процесс принятия решения;
- задача должна быть определена для любого числа этапов (шагов) и иметь структуру, не зависящую от их числа;

• при рассмотрении задачи на каждом шаге должно быть задано множество параметров, описывающих состояние системы;

• выбор решения (управления) на каждом шаге не должен оказывать влияния на предыдущие решения.

Примеры задач, решаемые методом динамического программирования.

Пример № 1. Задача замены оборудования.

Задача формулируется следующим образом.

Необходимо определить оптимальную стратегию использования оборудования в период времени длительностью n лет, при условии, что прибыль за каждые i лет ($i = 1, m$) от использования оборудования возраста t лет должна быть максимальной.

Известны следующие данные:

$f(t)$– выручка от реализации продукции, произведенной за год на оборудовании возраста t лет;

$g(t)$ – годовые затраты, зависящие от возраста оборудования t;

$q(t)$ − остаточная стоимость оборудования возраста t лет;

S – стоимость нового оборудования.

Возраст оборудования определяется как период эксплуатации оборудования после последней замены, выраженный в годах.

Пример №2. Задача о рюкзаке.

Задача формулируется следующим образом.

Имеется определенный набор предметов из конечного числа видов, для которых указана ценность. Известен и ограничен объем рюкзака.

Задача заключается в определении количества предметов каждого вида, которое необходимо поместить в, чтобы их суммарная ценность была максимальной, при условии, что их суммарный объем меньше объема рюкзака.

Операция, которая рассматривается в данной задаче, является многоэтапной.

Каждый этап будет связан с укладкой предметов отдельного вида в рюкзак. В этом случае, на -ом этапе будет выполняться поиск решения задачи, тем самым определяя, какое количество предметов k-го вида требуется положить в рюкзак, если уже какая-то часть общего объема занята другими предметами 1-го, 2-го, ..., $(k-1)$-го видов. В данной задаче выполняется переход от одного этапа к другому по перечню предметов.

Пример №3. Задача о прокладке выгодного пути от одного пункта к другому.

Задача формулируется следующим образом.

Из пункта А до пункта В необходимо проложить дорогу, таким образом, чтобы суммарные затраты на постройку участка были минимальными.

Указанную задачу можно представить, как многошаговый процесс.

Необходимо данный отрезок от А до В разделить на n частей, провести через точки деления прямые, которые будут перпендикулярны отрезку АВ.

Если прямые окажутся неподалеку друг от друга, то в этом случае отрезки пути, а также соединяющие их линии, считаются прямолинейными.

Таким образом, можно связать каждый этап с выбором направления перехода с одной прямой на другую.

Затраты на построение всего пути будут определяться, как затраты на строительство отдельных участков, которые соединяются прямыми, расположенные по соседству.

Вывод по главе 1

В первой главе выпускной квалификационной работы была описана общая постановка задачи динамического программирования.

Было рассмотрено уравнение Беллмана, как основное соотношение для решения оптимизационных задач методом динамического программирования.

Были рассмотрены особенности математической модели динамического программирования. Так же представлены примеры прикладных задач имеющие необходимые свойства для применения к ним метода динамического программирования.

ГЛАВА 2 ЗАДАЧА ОПТИМАЛЬНОГО РАСПРЕДЕЛЕНИЯ РЕСУРСОВ

2.1 Общая постановка задачи оптимального распределения ресурсов.

Задача формулируется следующим образом.

Имеется определенное количество ресурсов, к которым относятся денежные ресурсы и материальные ресурсы (трудовые ресурсы, сырье, оборудование и т. п.).

Указанные ресурсы требуется распределить между несколькими объектам их использования, разделяя на отдельные промежутки планового периода или части имеющихся ресурсов по различным объектам таким образом, чтобы в итоге получилось оптимальное распределение, гарантирующее максимальную суммарную эффективность. В качестве показателей эффективности можно использовать полученную прибыль, суммарные затраты или объемы готовой продукции.

Описание задачи оптимального распределения ресурсов.

В течении n лет между p предприятиями необходимо распределить имеющееся начальное количество ресурсов (средств) S_0. Выделив в -ом году i-му предприятию средства $x_{ki}(k = 1,n; i = 1,p)$, предприятие получит прибыль в размере $f_{ki}(x_{ki})$ и к концу года ресурсы возвращаются в количестве $g_{ki}(x_{ki})$. На следующем этапе распределения полученная прибыль может быть использована полностью или частично, либо не использована вовсе. Требуется определить оптимальный способ распределения ресурсов (средств), которые следует выделить -му предприятию в k-ом году, чтобы суммарная прибыль от p предприятий за n лет была максимальной [8].

Таким образом, показателем эффективности всего процесса распределения будем считать суммарную прибыль от p предприятий за n лет:

$$Z = \sum_{i=1}^{p} \sum_{k=1}^{n} f_{ki}(x_{ki}) \qquad (2.1)$$

Величина S_{k-1} характеризует количество ресурсов в начале -го года, т.е. является параметром состояния системы. Управлением на k-ом шаге является выбор переменных $x_{k1}, x_{k2}, \dots, x_{kp}$, характеризующих средства, выделяемые в k-ом году -му предприятию.

Предположим, что прибыль на -ом этапе распределения не используется. Тогда уравнение состояния имеет вид:

$$S_k = S_{k-1} - \sum_{i=1}^{p} x_{ki} + \sum_{i=1}^{p} g\ x_{ki} \qquad (2.2)$$

Если же некоторая часть прибыли используется на следующем этапе распределения, то к правой части уравнения (2.2) прибавляется полученная величина прибыли.

Необходимо определить np переменных x_{ki} удовлетворяющих условиям (2.2) и максимизирующих функцию (2.1). Вычислительная схема динамического программирования начинается с определения функции $Z_k^*(S_{k-1})$, обозначающей прибыль, начиная с -го года до конца текущего периода, при оптимальном разделении средств между p предприятий, если в k-ом году распределялось S_{k-1} средств. Функции $Z_k^*(S_{k-1})$ для $k = 1,2,\dots,n-1$ удовлетворяют рекуррентным соотношениям (1), которые примут вид:

$$Z_k^*\ S_{k-1}\ = \max_{0 \le \sum_{i=1}^{p} x_{ki} \le S_{k-i}} \sum_{i=1}^{p} f_{ki}\ S_{k-1}, x_k\ + Z_k^*(S_k) \qquad (2.3)$$

При $k = n$ согласно (1) получаем:

$$Z_n^*\ S_{n-1}\ = \max_{0 \le \sum_{i=1}^{p} x_{ni} \le S_{n-1}} \sum_{i=1}^{p} f_{ni}(x_n) \qquad (2.4)$$

Процесс поиска решения состоит в последовательном вычислении уравнений (2.4) и (2.3) для всех допустимых состояний системы S_k , $(k = n -$

$1, n-2, \ldots, 1$). Причем каждое уравнение зависит от p переменных и является задачей оптимизации функции.

В итоге задача с np переменными сводится к ряду из n задач, каждая из которых содержит p переменных.

2.2 Решение задач оптимального распределения ресурсов.

Задача 1.

На период n лет составляется план работы двух предприятий ($p = 2$).

Имеется определенное количество средств, которые составляют S_0. Средства в размере x, выделенные предприятию П1, к концу года приносят прибыль в количестве $f_1(x)$ и возвращаются в количестве $g_1(x) < x$, аналогично для предприятия П2, приносят прибыль в количестве $f_2\ x$ и возвращаются в количестве $g_2(x) < x$. В конце года оставшиеся ресурсы снова перераспределяются между предприятиями П1 и П2, полученный доход в предприятия не вкладываются, новые средства не поступают.

Требуется выбрать оптимальный способ распределения начальных средств на n лет.

Рассмотрим процесс распределения ресурсов как многошаговый с количеством шагов n. Номер каждый шаг равен номеру года. Два предприятия с вложенными в них средствами представляют собой управляемую систему. Система имеет один параметр состояния S_{k-1} ($k = 1, n$) − количество средств, которые нужно распределить в начале -го года. На каждом шаге имеем две переменные управления: x_{k1} и x_{k2} − количество средств, выделенных предприятию П1 и П2 соответственно. Так как средства ежегодно распределяются полностью, то $x_{k2} = S_{k-1} - x_{k1}$ ($k = 1, n$). На каждом шаге задача становится одномерной. Обозначим x_{k1} через x_k, тогда:

$$x_{k2} = S_{k-1} - x_k.$$

Показатель эффективности на k-ом шаге равен $f_1\ x_k\ + f_2\ S_{k-1} - x_k$ − прибыль полученная от двух предприятий в течении k-го года

Показатель эффективности всего процесса – это прибыль, полученная от двух предприятий в течении n лет, равная:

$$Z = \sum_{k=1}^{n} [f_1(x_k) + f_2(S_{k-1} - x_k)]. \quad (2.5)$$

Уравнение, характеризующее состояние системы, представляет собой остаток средств S_k после распределения на -ом шаге и запишется следующим образом:

$$S_k = g_1\ x_k\ + g_2(S_{k-1} - x_k) \quad (2.6)$$

Введем величину $Z_k^*(S_{k-1})$, которая представляет собой возможную оптимальную прибыль, полученную после распределения средств в размере S_{k-1} между двумя предприятиями, начиная с -го года до конца планируемого периода. Получим основные оптимальные уравнения для этих функций:

$$\begin{gathered} Z_n^*\ S_{n-1}\ = \max_{0 \le x_n \le S_{n-1}} f_1\ x_n\ + f_2\ S_{n-1} - x_n\ ; \\ Z_k^*\ S_{k-1}\ = \max_{0 \le x_k \le S_{k-1}} f_1\ x_k\ + f_2\ S_{k-1} - x_k\ + \\ Z_{k+1}^*(S_k)\ , \end{gathered} \quad (2.7)$$

где S_k − находится из уравнения состояния (2.6).

Решаем поставленную задачу для следующих условий:

$$S_0 = 10000; n = 4;$$

$$f_1\ x\ = 0{,}4x; f_2\ x\ = 0{,}3x;$$

$$g_1\ x\ = 0{,}5x; g_2\ x\ = 0{,}8x.$$

Если x_k и $S_{k-1} - x_k$ – количество средств, выделенных предприятиям П1 и П2 в -м году, то общая прибыль, полученная от двух предприятий, равна:

$$Z_k = 0{,}4x_k + 0{,}3S_{k-1} - 0{,}3x_k = 0{,}1x_k + 0{,}3S_{k-1},$$

а уравнение состояния (1.6) принимает вид:

$$S_k = 0{,}5x_k + 0{,}8\ S_{k-1} - x_k\ = 0{,}8S_{k-1} - 0{,}3x_k$$

Основные функциональные уравнения (2.7) запишутся следующим образом:

$$Z_4^*\ S_3\ = \max_{0 \le x_4 \le S_3}\ 0{,}1x_4 + 0{,}3S_3\ ;$$

$$Z_k^*\ S_{k-1}\ = \max_{0 \le x_k \le S_{k-1}}\ 0{,}1x_k + 0{,}3S_{k-1} + Z_{k+1}^*(0{,}8S_{k-1} - 0{,}3x_k) \quad k = 1{,}2{,}3\ .$$

I этап. Условная оптимизация.

4-й шаг. Условный оптимальный доход равен:

$$Z_4^*\ S_3\ = \max_{0 \le x_4 \le S_3}\ 0{,}1x_4 + 0{,}3S_3\ = 0{,}4S_3$$

Так как показатель эффективности $Z_4(S_3)$ является линейной функцией относительно x_4 и эта переменная входит в выражение со знаком плюс, то данный показатель достигает максимума в конце интервала $0 \le x_4 \le S_3$, т.е. при $x_4 = S_3$.

3-й шаг:

$$Z_3^*\ S_2\ = \max_{0 \le x_3 \le S_2}\ 0{,}1x_3 + 0{,}3S_2 + 0{,}4(0{,}8S_2 - 0{,}3x_3)$$

$$= \max_{0 \le x_3 \le S_2}\ -0{,}02x_3 + 0{,}62S_2\ = 0{,}62S_2\ .$$

Коэффициент при x_3 отрицателен, поэтому максимум в этой линейной функции относительно x_3 достигается в начале интервала$0 \le x_3 \le S_2$, т.е.

$$x_3 = 0.$$

2-й шаг:

$$Z_2^*\ S_1\ = \max_{0 \le x_2 \le S_1}\ -0{,}086x_2 + 0{,}796S_1\ = 0{,}796S_1;$$

$$x_2 = 0.$$

1-й шаг.

$$Z_1^*\ S_0\ = \max_{0 \le x_1 \le S_0} -0{,}1388x_1 + 0{,}9368S_0\ = 0{,}9368S_0;$$

$$x_1 = 0.$$

Результат условной оптимизации:

$$Z_1\ S_0\ = 0{,}9368S_0, x_1 = 0;$$
$$Z_2\ S_1\ = 0{,}796S_1, x_2 = 0;$$
$$Z_3\ S_2\ = 0{,}62S_2, x_3 = 0;$$
$$Z_4\ S_3\ = 0{,}4S_3, x_4 = S_3.$$

II этап. Безусловная оптимизация.

Полученное значение эффективности на первом шаге $Z_1\ S_0\ = 0{,}9368S_0$, зная что начальное количество средств $S_0 = 10000$ то максимальный суммарный доход составит 9368.

Зная $x_1 = 0$, находим $S_1 = 8000$ используя S_1 и $x_2 = 0$ получаем $S_2 = 6400$. Аналогично $x_3 = 0, S_3 = 5120$ из чего следует, что $x_4 = 5120$ т.к. $x_4 = S_3$. Получаем что, средства следует распределить как указано в таблице 2.1.

Таблица 2.1 – Результат распределения средств

Год	Предприятие	
	П1	П2
1	0	10000
2	0	8000
3	0	6400
4	5120	0

Задача 2.

Денежные средства в количестве 100 млн ден.ед. необходимо распределить между четырьмя предприятиями (П1, П2, П3, П4). Средства выделяются частями, кратными 20. Для каждого предприятия разработан план реализации выделенных средств. В таблице 2.2 указан прирост выпуска

продукции $f_p\ x_k$, $(p = 1{,}4, k = 1{,}6)$ который получит предприятие при реализации выделенных средств.

Таблица 2.2. – Прирост выпуска продукции

Выделенные средства, млн ден.ед.	П1	П2	П3	П4
	Прирост выпуска продукции на предприятиях, шт.			
	$f_1(x_k)$	$f_2(x_k)$	$f_3(x_k)$	$f_4(x_k)$
0	0	0	0	0
20	9	10	6	12
40	16	18	12	17
60	22	20	25	20
80	25	28	30	27
100	30	35	37	36

Данную задачу рассмотрим, как многошаговый процесс. Шагом процесса распределения будем считать выделение средств предприятиям: первый шаг – выделение средств предприятию П1, второй – П2, и т.д. таким образом весь процесс разбивается на четыре шага. За состояние процесса распределения примем количество имеющихся средств. Начальное состояние системы $S_0 = 100$. Выбранное управление на каждом шаге – это количество денежных средств, выделяемых предприятию u_k , $k = 1{,}4$. Критерием эффективности для каждого шага – прирост выпуска продукции, полученный предприятием $f_k(x_k)$, $k = 1{,}4$. Критерий эффективности всего процесса– это суммарный прирост выпуска продукции всех предприятий:

$$Z = \sum_{k=1}^{4} f_k(x_{k-1}, u_k),$$

где x_{k-1} − возможные остатки средств (множество состояний), которыми располагает производства (система S) перед k-м шагом ;

u_k – возможные суммы, выделяемые -му предприятию (множество управлений, на k-м шаге для перевода системы S в одно из состояний множества x_k);

f_k – значение прироста выпуска продукции на -м предприятии.

I этап. Условная оптимизация.

Шаг 1.

Допустим, что все имеющиеся средства выделяются одному предприятию. Получаем, что максимально возможный прирост выпуска продукции на этом предприятии, соответствующий выделенной сумме, отвечает определенное (единственное) значение дохода из составленного плана, поэтому можно записать, что:

$$Z_1(x) = f_1(x).$$

Шаг 2.

На данном шаге средства выделяются второму предприятию с учетом остатка средств после первого шага. Выделив второму предприятию количество средств x_2, прирост продукции на нем составит $f_2(x_2)$. Остаток средств выделяемые другому предприятию $(x - x_2)$ в зависимости от величины x_2 позволит увеличить прирост выпуска продукции до максимального значения $F_1(x - x_2)$. При данном условии функциональное уравнение характеризующее суммарный прирост продукции на двух предприятиях имеет вид:

$$F_2(x) = \max_{0 \le x \le 100} f_2(x_2) + F_1(x - x_2).$$

Условное распределение на втором шаге будет следующим образом таблица 2.3.

Таблица 2.3 – Распределение на втором шаге

x	0	20	40	60	80	100
$F_2(x)$	0	10	19	27	34	40
u_2	0	20	20	40	40	40

Шаг 3.

На этом шаге средства выделяются третьему предприятию с учетом второго шага. Выделив третьему предприятию сумму x_3, прирост выпуска продукции на нем составит $f_3(x_3)$. При этом условии функциональное уравнение суммарного прироста продукции на трех предприятиях имеет вид:

$$F_3(x) = \max_{0 \le x \le 100} (f_3\ x_3\ + F_2\ x - x_3\ ).$$

Распределение на третьем шаге будет следующим образом таблица 2.4

Таблица 2.4 – Распределение на третьем шаге

x	0	20	40	60	80	100
$F_3(x)$	0	10	19	27	35	44
u_3	0	0	0	0	60	60

Шаг 4.

Средства выделяются четвертому предприятию с учетом предыдущего шага. Выделив четвертому предприятию средства в размере x_4, прирост выпуска продукции на нем составит $f_4(x_4)$. Функциональное уравнение суммарного прироста продукции на четырех предприятиях имеет вид:

$$F_4\ x\ = \max_{0 \le x \le 100} (f_4\ x_4\ + F_3\ x - x_4\ ).$$

Распределение на третьем шаге будет следующим образом таблица 1.4

Таблица 2.5 – распределение на третьем шаге

x	0	20	40	60	80	100
$F_4(x)$	0	12	22	31	39	47
u_4	0	20	20	20	20	20

Из таблицы 2.5 получаем, что распределив все средства (100 млн. ден. ед.), максимальный прирост выпуска продукции на четырех предприятиях составит 47 шт..

II этап. Безусловная оптимизация.

Из таблицы 2.4 видно, что $u_4 = 20$ зная это можно определить остальные распределения суммы для оставшихся предприятий. Пологая что $S_0 = 100$ получаем, что $S_3 = 80$ из таблицы 2.3 следует $u_3 = 60$, аналогично $S_2 = 60$ из таблицы 2.2 получаем $u_2 = 20$. На П1 средства не выделяются $u_1 = 0$. В таблице 2.5 отражено оптимальное распределение для задачи 2.

Таблица 2.6 – Решение задачи 2.

Предприятие	П1	П2	П3	П4
Выделяемая сумма	0	20	60	20

Вывод по главе 2

В данной главе была сформулирована постановка задачи оптимального распределения ресурсов. Так же представлена математическая модель данной задачи.

Аналитически решена задача оптимального распределения ресурсов с применением метода динамического программирования.

ГЛАВА 3 РЕАЛИЗАЦИЯ АЛГОРИТМА РЕШЕНИЯ ЗАДАЧИ ОПТИМАЛЬНОГО РАСПРЕДЕЛЕНИЯ РЕСУРСОВ

3.1 Анализ требований к программной реализации алгоритма решения задачи оптимального распределения ресурсов

Для определения требований будем использовать классификацию FURPS+ [5].

FURPS − это метод проверки приоритетных требований после понимания потребностей и задач клиента.

Аббревиатура FURPS − с английского функциональность, удобство использования, надежность, производительность и поддерживаемость.

С течением времени возникла необходимость в увеличению типов предъявляемых требований, что привело к появлению версии данной технологий FURPS +.

Эта технология сделала классификацию требований более внимательной к пониманию различных типов нефункциональных требований.

Система классификаций разделяется на функциональные и нефункциональные требования.

Функциональные требования определяют то, что система должна делать.

Нефункциональные требования регламентируют внутренние и внешние условия или атрибуты функционирования системы.

Рассмотрим основные положения данной технологии с учетом особенностей разрабатываемой программы.

1) Functionality, функциональность:
- решение задачи оптимального распределения ресурсов;

2) Usability, удобство использования:
- наличие справочной информации;
- отсутствие функциональной избыточности.

3) Reliability, надежность:

- допустимая частота/периодичность сбоев: 1 раз в 300 часов;
- среднее время сбоев: 1 час;
- возможность восстановления системы после сбоев: 1 час;
- режим работы 7/24/365.

4) Performance, производительность:
- допустимое количество одновременно работающих пользователей: 5;
- время реакции на возникновение аварийной ситуации – 10 мс.

5) Supportability, поддерживаемость:
- дистанционное администрирование;
- время устранения критических проблем: в течение 10 часов.

6) Проектные ограничения:
- использование метода динамического программирования;
- низкая стоимость владения;
- реализация с помощью современных Web-технологий.

С учетом вышеперечисленных требований разработана диаграмма вариантов использования разрабатываемой программы (use case), которая изображена на рисунке 3.1.

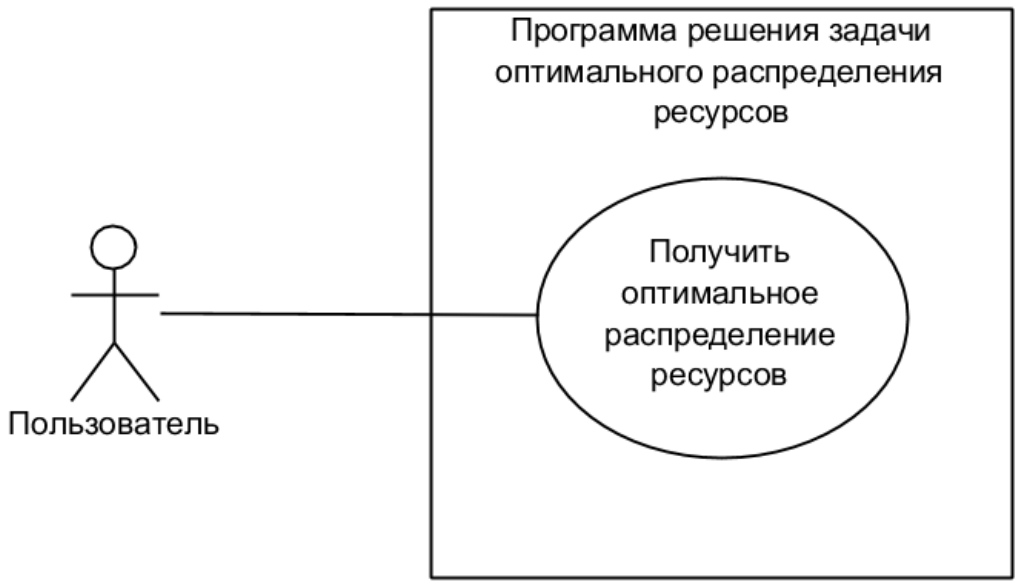

Рисунок 3.1 − Диаграмма вариантов использования программы

На диаграмме представлены следующие элементы:

• граница программы, очерченная прямоугольником;

• основной актер – Пользователь;

• вариант использования – Получить оптимальное распределение ресурсов.

Спецификации прецедентов изображены в виде таблицы (таблица 3.1).

Таблица 3.1 – Спецификация прецедентов

Прецедент: получить оптимальное распределение ресурсов
ID: 2
Краткое описание: Выбрать оптимально распределение ресурсов в зависимости от введенных данных
Главный актер: Пользователь
Второстепенные актеры: нет
Предусловия: нет
Основной поток: 1. Прецедент запускается после ввода данных и нажатия на кнопку распределения ресурсов Пользователем. 2. Программа выполняет решение задачи оптимального распределения ресурсов. 3. Программа выводит результат в виде таблицы распределения
Постусловия: нет
Альтернативные потоки: нет

Для построения диаграммы вариантов использования использовался программный продукт Visual Paradigm for UML 8.0.

Данный продукт представляет собой CASE-средство, обеспечивающее поддержку задач моделирования информационных систем и бизнес-процессов на основе языка UML 2.0 с возможностью последующей генерации программного кода и проектирования базы данных моделируемой системы.

3.2 Проектирование программы для решения задачи оптимального распределения ресурса

Процесс проектирования программы является одной из ключевых стадий ее жизненного цикла.

Проектирование программы осуществляется с учетом требований, сформулированных в предыдущем разделе.

На рисунке 3.2 представлена блок-схема алгоритма, реализуемого проектируемой программой.

Предварительно опишем входные данные программы.

Входными данными программы являются:

1) начальное состояние системы S (количество распределяемых средств);

2) количество вариантов эффективного вложения n (размер множества управлений x);

3) количество объектов распределения (предприятий) p;

4) критерии эффективности для каждого предприятия относительно вложения.

Выходными данными является результат в виде оптимального распределения ресурса и суммарная эффективность.

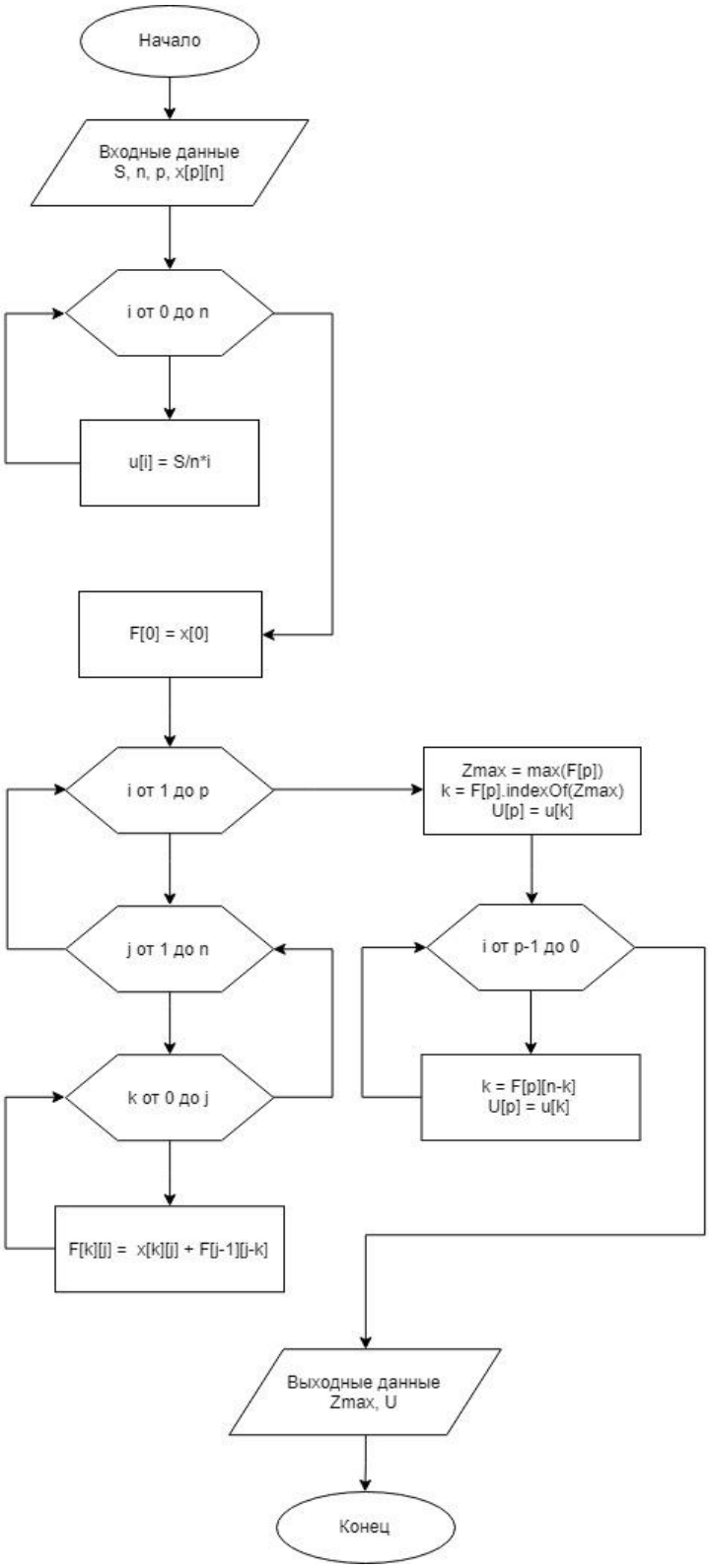

Рисунок 3.2 – Блок-схема алгоритма программы

3.3 Выбор языка программирования

Для выбора средства разработки программы опишем и сравним нижеследующие популярные языки программирования.

Java − это язык программирования, разработанный Джеймсом Гослингом в 1995 году для Sun Microsystems.

Это объектно-ориентированный язык, похожий на C++, но с расширенными и упрощенными функциями. Java имеет свободный доступ и может работать на всех платформах [19].

Java является параллельной средой программирования, позволяющей выполнять множество операций. Помимо этих функций, Java также является независимым языком программирования, который следует принципу «Write once, Run where». Это означает, что скомпилированный код может работать на всех платформах, поддерживающих java.

Говоря простыми словами, это вычислительная платформа, где вы можете разрабатывать приложения.

Основные преимущества Java:

- Простота освоения.
- Независимость от платформы: любое приложение, написанное на языке Java, может быть легко перенесено на другую платформу.
- Объектная ориентированность: Java - это объектно-ориентированный язык программирования. В данном языке все элементы интерпретируются как объекты, и все операции выполняются с использованием этих объектов.
- Безопасность: Java является защищенным языком, потому что весь код преобразуется в байтовый код после компиляции, который не читается человеком. Кроме того, Java не использует явные указатели и запускает программы внутри так называемой «песочницы» (sandbox), чтобы предотвратить любые действия из ненадежных источников. Это позволяет разрабатывать безвирусные, без вмешательства пользователя системы / приложения.

• Динамичность: Java динамичен по своей природе, поскольку он обладает способностью адаптироваться к развивающейся среде.

• Надежность: Java обладает мощной системой управления памятью. Это помогает устранять ошибки, когда проверяется код во время компиляции и времени выполнения.

• Высокая производительность: Java достигает высокой производительности благодаря использованию байт-кода, который можно легко перевести на собственный машинный код.

• Java поддерживает автоматическую сборку мусора. А это значит, что вам не надо освобождать вручную память от ранее использовавшихся объектов, так как сборщик мусора это сделает автоматически за вас;

По данным 2016 г. язык Java является одним из самых популярных языков программирования.

Есть несколько типов платформ Java:

• Java SE (Standard Edition). Предназначена для создания небольших приложений в масштабах малого предприятия;

• Java EE (Enterprise Edition). Нацелена на создание более сложных приложений.

C++ − мощный язык программирования, стандартизованный международной организацией стандартизации (ISO) в 1998 году.

C++ обладает объектно-ориентированными функциями программирования, включая возможность определять классы и функции и используется во многих популярных программных приложениях, например, таких, как Firefox, Adobe Photoshop, MySQL и Microsoft Office [20].

Преимущества:

• объектная ориентированность;

• независимость от платформы;

• использование различных парадигм программирования;

• одинаково эффективен как язык низкоуровневого программирования и язык высокого уровня;

• обеспечение высокой производительности и эффективности управления памятью и др.

Недостатки:

• низкий уровень безопасности;

• сложность разработки программ высокого уровня;

• не поддерживает сбор мусора;

• не поддерживает встроенные потоки и др.

Python − это многостраничный, универсальный, интерпретируемый, высокоуровневый язык программирования.

Python позволяет программистам использовать разные стили программирования для создания простых или сложных программ, получения более быстрых результатов и написания кода на языке, сильно приближенном к языку человеческого общения.

К некоторым из популярных систем и приложений, которые в разное время использовали Python для разработки относятся: Google, YouTube, BitTorrent, Google App Engine, Eve Online, Maya и iRobot и др. [21].

Для проведения сравнительного анализа языков программирования все их важные характеристики собраны в таблице 3.2.

Знаки «+» и «-» обозначаются соответствие и несоответствие выбранному критерию.

Таблица 3.2 – Сравнительный анализ языков программирования

Критерии оценки	Java	C++	Python
Перегрузка функций	+	+	-
Кроссплатформенность	+	-	+
Наличие опыта разработчика	+	+	-
Итого (+)	3	2	1

На основе проведенного анализа был выбран язык Java.

3.4 Выбор графического фреймворка для реализации интерфейса программы

Фреймворк (от англ. framework − структура) − программная платформа, определяющая структуру программного обеспечение, облегчающее разработку. В пункте 3.2 на основе анализа языков программирования был выбран язык Java. Для разработки интерфейса на языке Java существует несколько программных платформ для реализации графического интерфейса, самые популярные из них: AWT, Swing и JavaFX [19].

AWT − Abstract Window Toolkit разработанная Sun Microsystems. Слово abstract в названии указывает, что в качестве стандартных компонентов используются встроенные компоненты той системы, где запускается приложение. Библиотека AWT имеет следующие достоинства:

- часть JDK;
- скорость работы;
- графические компоненты похожи на стандартные.

Так же имеет ряд недостатков:

- использование встроенных в систему компонентов налагает ограничения на использование их свойств. Некоторые компоненты могут вообще не работать на некоторых платформах;
- стандартных компонентов AWT очень немного, часто приходится разрабатывать компоненты самостоятельно;
- программа выглядит по-разному на разных платформах, что может повлиять на правильность отображения компонентов.

Swing − разработана компанией Sun Microsystems вслед за AWT. В отличии от AWT полностью написана на Java и не использует системные графические компоненты, что является плюсом в кроссплатформенности языка Java. Как и все программные платформы имеет свои достоинства:

- часть JDK;

• Swing популярный фреймворк, имеет много информации в интернете и печатных изданиях;

• почти каждая среда разработки (IDE) имеет встроенный редактор форм.

А так же недостатки:

• окно программы с большим количеством компонентов работает медленно и подтормаживает;

• трудность в разработке сложных интерфейсов в редакторе форм.

JavaFX – это программная платформа для создания и развертывания настольных приложений, а также мощных интернет-приложений, которые могут работать на самых разных устройствах [18].

JavaFX поддерживает настольные компьютеры и веб-браузеры в Microsoft Windows, Linux и macOS.

До версии 2.0 JavaFX разработчики использовали статически типизированный декларативный язык под названием JavaFX Script для создания приложений JavaFX.

Поскольку JavaFX Script был скомпилирован в байт-код Java, программисты могли также использовать Java-код. Приложения JavaFX могут запускаться на любом рабочем столе, на котором можно запускать Java SE или на любом мобильном телефоне, который может запускать Java ME.

Недостатки:

• фреймворк находится в стадии разработки, поэтому случаются падения и некоторые ошибки приложений;

• JavaFX не имеет такого широкого распространения как Swing.

Для проведения сравнительного анализа рассмотренных платформ все их важные характеристики собраны в таблице 3.3.

Диапазон значений 0-10 определяет степень соответствия характеристики выбранному критерию.

Таблица 3.3 – Сравнительный анализ графических фреймворков

Критерии оценки	AWT	Swing	JavaFX
Быстродействие	5	8	10
Большая база компонентов	4	7	10
Наличие редактора форм	0	10	10
Опыт использования фреймворка	5	6	10
Итого	14	31	40

На основе проведенного анализа (таблица 3.3) была выбрана библиотека JavaFX.

3.5 Разработка кода программы для решения задачи оптимального распределения ресурсов

В качестве средства разработки кода программы использован язык программирования Java 8.

Преимуществом восьмой версии Java являются:

- наличие лямбда выражений (анонимных функций)
- встроенная библиотека Stream API, которая позволяет работать с данными в функциональном стиле [19].

Разработка велась в IntelliJ IDEA это интегрированная среда разработки программного обеспечения на Java от компании JetBrains.

В процессе разработки программы использован паттерн проектирования Model – View – Controller, MVC.

Архитектура паттерна MVC (Модель-Представление-Контроллер) помогает разделить приложения на логические единицы. Проще говоря, эта парадигма отделяет бизнес-логику от логики интерфейса. Эта архитектура делает приложение более эффективным.

Модель представляет собой уникальный объект - это может быть один объект или, скорее, структура. Между объектами и данными объекта существует взаимно однозначное отношение. Это модель, которая реагирует на запросы, исходящие из представления о ее статусе или состоянии. Таким образом, обработка данных происходит только в модели, что обеспечивает согласованность внутренних данных. Представление используется для представления графической визуализации пользовательского интерфейса. Оно представляет входные и выходные данные в интерфейсе с использованием различных элементов, таких как кнопки, меню, диалоговые окна и т. д. Контроллер обеспечивает связь между пользовательским интерфейсом (представлением) и логикой обработки приложения (моделью). Контроллер использует методы модели для извлечения информации об объекте приложения, для изменения статуса объекта и для представления об этом изменении. В некотором смысле контроллер позволяет пользователю вносить изменения и видеть результаты.

Диаграмма пакетов программы представлена на рисунке 3.3.

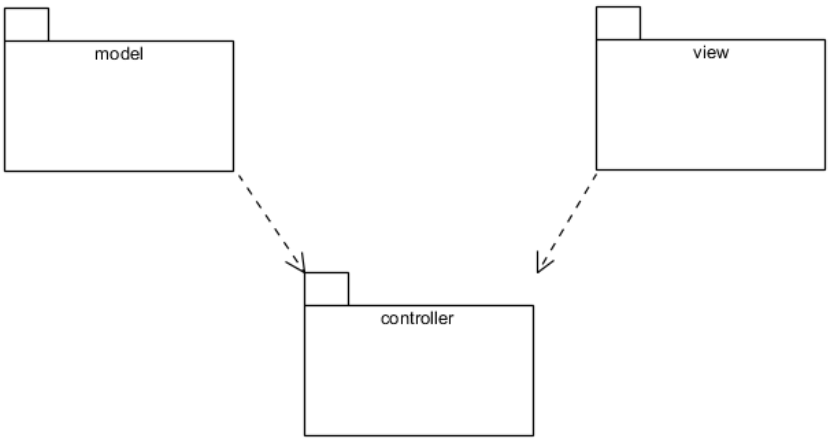

Рисунок 3.3 – Диаграмма пакетов программы

Основные классы программы:

• ResourceAllocationApp − исполняющий класс программы с методом main().

• ResourceAllocationSceneController – класс контроллер главной формы, содержит поля связанные с элементами на форме.

• ResourceAllocationSceneControllerImpl – в классе содержится вся логика работы с формой и алгоритм решения задачи.

• ResourceAllocationSceneModel – содержит поля для значений из объектов формы.

• FactoryModel – класс модель содержит поля, такие как номер объекта и список критерий эффективности.

• FactoryEffectModel – класс модель описывает полученный прирост дохода от вложенных средств.

• CalculationResultModel – класс модель полученного результата оптимального распределения.

На рисунке 3.4 представлена диаграмма разработанных классов.

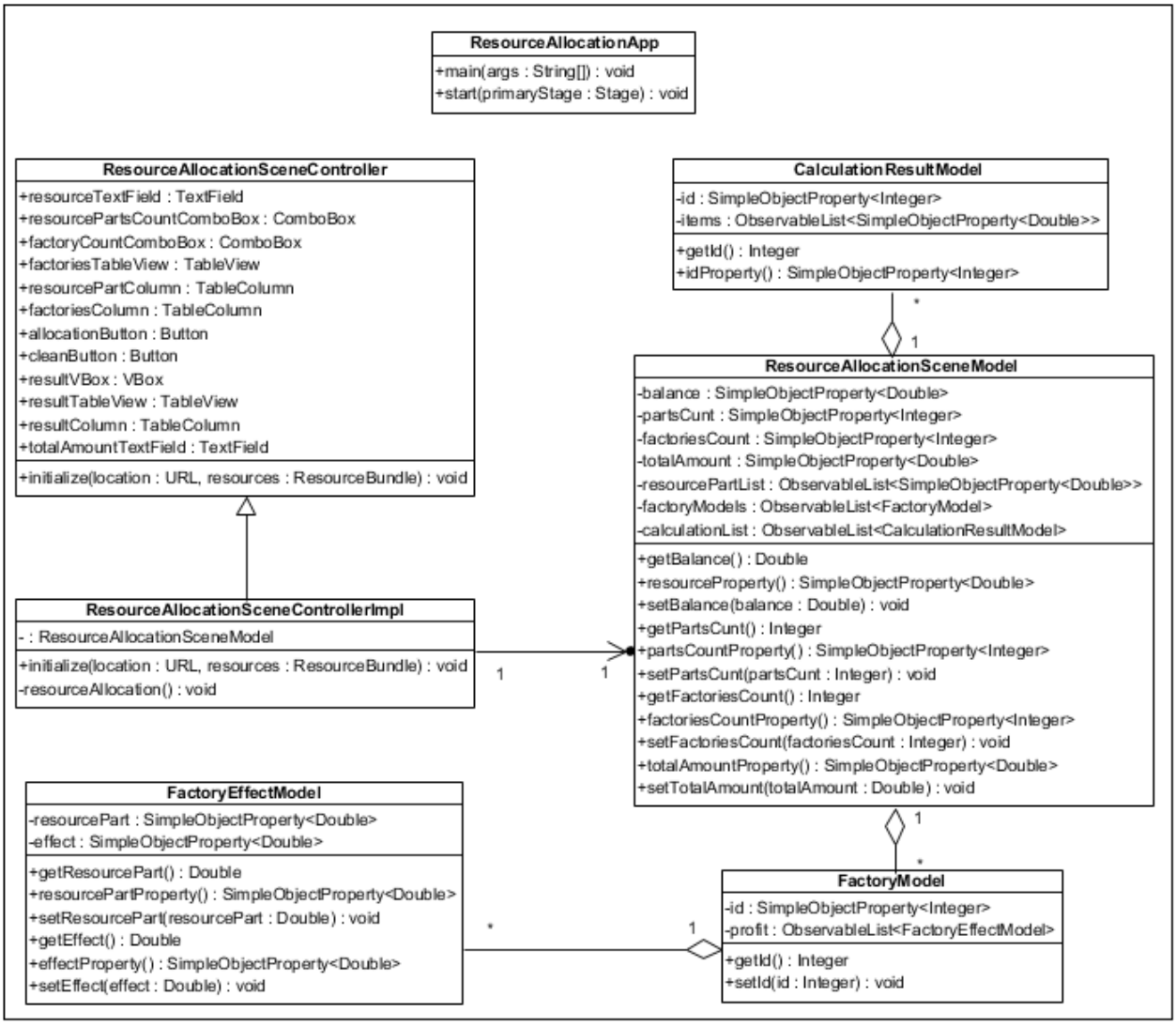

Рисунок 3.4 – Диаграмма классов программы

Решение задачи распределения ресурсов разбит на два этапа:

- условная оптимизация
- безусловная оптимизация

Фрагмент кода реализации алгоритма первого этапа решения задачи:

```
int p = model.getFactoriesCount(); //Количество предприятий
int n = model.getPartsCunt(); //Количество вариантов вложения
double[][] F = new double[p][n]; //Критерии эффективности
int[][] u = new int[p][n]; //Управление на каждом шаге
for (int i = 0; i < n; i++) {
    F[0][i] = model.getFactoryModels().get(0).getProfit().get(i).getEffect();
    u[0][i] = i;
}
for (int i = 1; i < p; i++) {
    FactoryModel factory = model.getFactoryModels().get(i);
    for (int j = 0; j < n; j++) {
        double[] a = new double[j + 1];
        for (int k = 0; k <= j; k++) {
            a[k] = factory.getProfit().get(k).getEffect() + F[i - 1][j - k];
        }
        double max = a[0];
        u[i][j] = 0;
        for (int l = 1; l < a.length; l++) {
            if (a[l] > max) {
                max = a[l];
                u[i][j] = l;
            }}
            F[i][j] = max;
  }
}
```

Код реализации второго этапа:

```
double[] U = new double[p]; //Оптимальные управления
double Fmax = F[p - 1][0]; //Суммарная эффективность
int k = u[p - 1][0];
for (int i = 1; i < n; i++) {
    if (F[p - 1][i] > Fmax) {
        Fmax = F[p - 1][i];
        k = u[p - 1][i];
    }
}
U[p - 1] = model.getResourcePartList().get(k).get();
n = n - 1;
for (int j = p - 2; j >= 0; j--) {
    n = n - k;
    k = u[j][n];
    U[j] = model.getResourcePartList().get(k).get();
}
```

Реализация «view» компонентов выполнена в виде файла в формате графической разметки FXML.

Разработанный интерфейс программы представлен на рисунке 3.5.

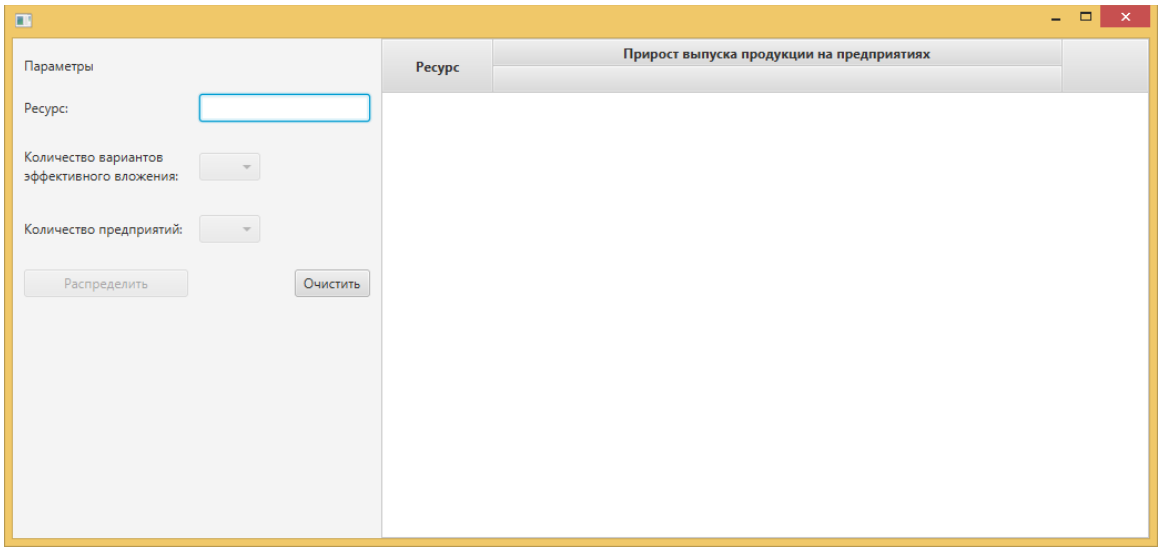

Рисунок 3.5 – Интерфейс программы

Логика интерфейса построена таким образом, что пользователь не сможет сделать расчет пока не будут введены все начальные данные.

Для демонстрации работы программы было использовано условие задачи 2 из пункта 2.2 второй главы работы.

Результат вычислений (работы программы) изображен на рисунке 3.6.

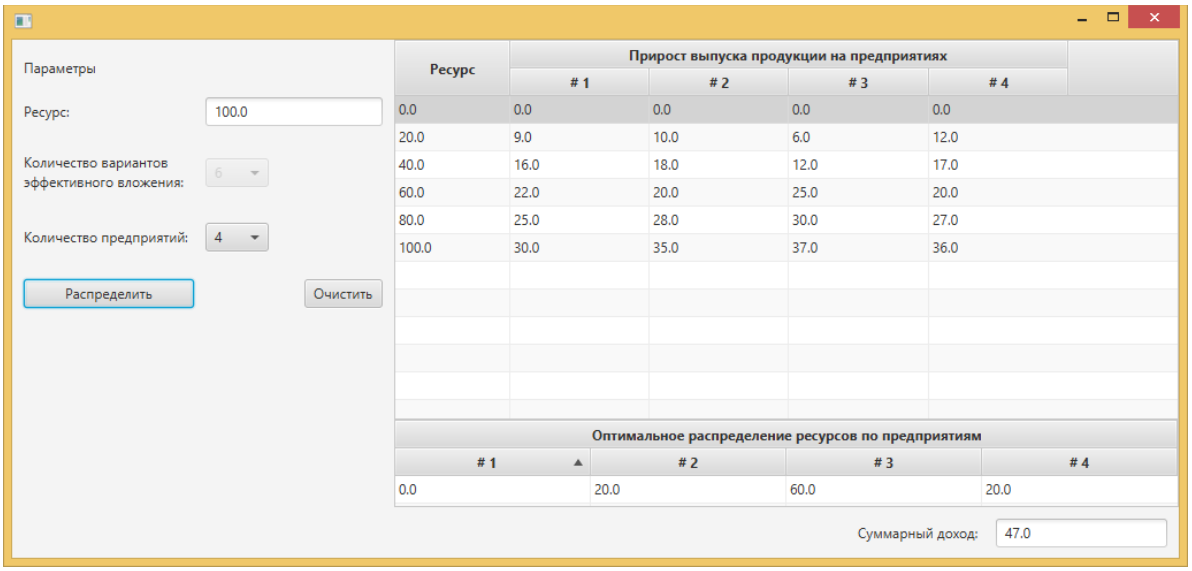

Рисунок 3.6 – Результат работы программы

3.6 Тестирование и отладка программы для решения задачи оптимального распределения ресурсов

Для тестирования и отладки разработанной программы использован метод модульного тестирования.

В компьютерном программировании модульное тестирование - это метод тестирования программного обеспечения, посредством которого отдельные единицы исходного кода, наборы одного или нескольких компьютерных программных модулей вместе со связанными данными управления, процедурами использования и рабочими процедурами проверяются на предмет того, подходят ли они для применения в тестируемой программе.

Интуитивно можно рассматривать модуль как самую маленькую тестируемую часть приложения.

В процедурном программировании модуль тестирования может быть целым программным модулем, но чаще всего это отдельная функция или процедура.

В объектно-ориентированном программировании модуль тестирования часто представляет собой целый интерфейс, такой как класс, но может быть индивидуальным методом.

Модульные тесты - это короткие фрагменты кода, созданные программистами для тестирования по методу «белого ящика».

В рассматриваемом случае для модульного тестирования программной реализации алгоритма распределения ресурса выбрана платформа JUnit.

Полученный опыт при работе с модульным тестированием важен в разработке концепций тестирования программного обеспечения.

Следует отметить, что среда программирования IntelliJ IDEA, поставляемая с фреймворком JUnit, имеет все необходимые средства для отладки программного кода.

Пример реализации простого теста представлен на рисунке 3.7.

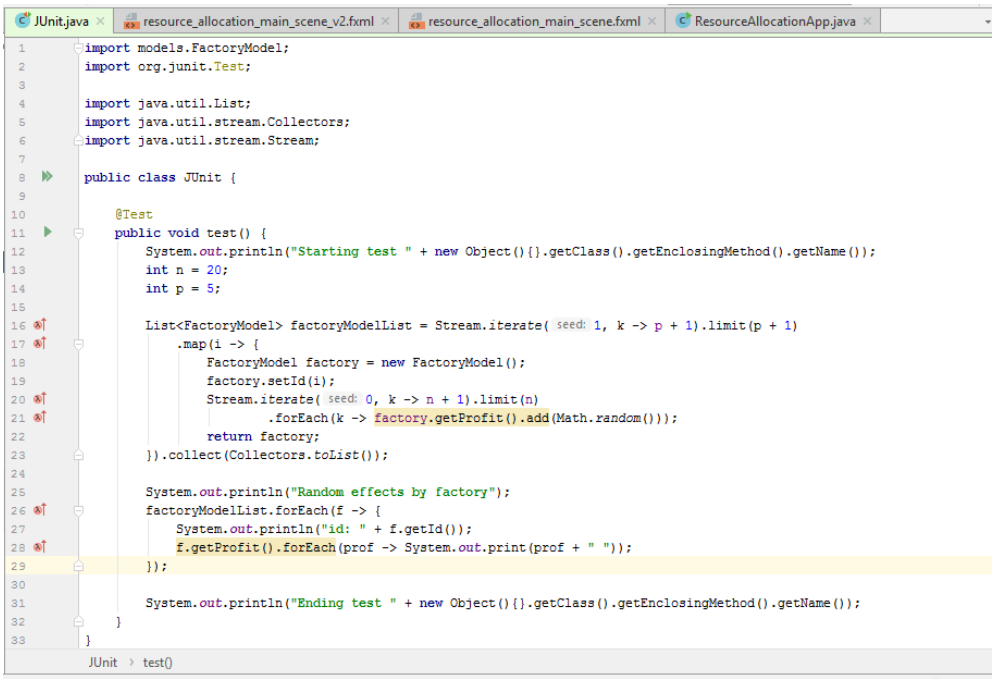

Рисунок 3.7 – Реализация простого теста

Таким образом, модульное тестирование разработанной программы подтвердило ее работоспособность.

Вывод по главе 3

Произведен анализ требований к программной реализации алгоритма решения задачи распределения ресурсов

На основе выделенных требований спроектирована программа решения задачи оптимального распределения ресурсов. Описаны входные и выходные данные программы.

Проанализированы и выбраны язык программирования и графический фреймворк для программной реализации алгоритма решения поставленной задачи.

Разработан и протестирован код программы. Так же представлен разработанный графический интерфейс.

ЗАКЛЮЧЕНИЕ

Тема бакалаврской работы посвящена актуальной проблеме реализации алгоритма решения задачи оптимального распределения ресурсов на основе метода динамического программирования.

В ходе выполнения бакалаврской работы достигнуты следующие результаты:

1. Проанализированы общие подходы динамического программирования к решению задач оптимального распределения ресурсов экономических и производственных систем.

2. Аналитически решена задача оптимального распределения ресурсов с помощью метода динамического программирования.

3. Выполнена реализация алгоритма решения задачи распределения ресурсов. Данная задача была решена в ходе проектирования программы и ее интерфейса.

Таким образом, основным результатом процесса проектирования является программная реализация алгоритма решения задачи оптимального распределения ресурсов методом динамического программирования.

Результаты работы могут быть рекомендованы для решения задач управления ресурсами экономических и производственных систем.

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

Научная и методическая литература

1. Акулич И.Л. Математическое программирование в задачах и упражнениях: учебное пособие / И.Л. Акулич. СПб.: Лань, 2011. − 352 с.

2. Бабаков Н.А. Теория линейных систем автоматического управления / Н.А. Бабаков, А.А. Воронов. Москва: Высшая школа, 1986. − 367 с.

3. Беллман Р.Э., Дрейфус С. Прикладные задачи динамического программирования / Р.Э. Беллман, С. Дрейфус, [пер. с англ. Н.М. Митрофановой]. Москва: Наука, 1965. − 458 с.

4. Бронов С.А. Методы оптимизации в САПР : конспект лекций для спец. 230104.65 / С. А. Бронов. Красноярск: НИИ АММ, 2011. − 126 с.

5. Гома Х. UML-проектирование систем реального времени параллельных и распределенных приложений: [Пер. с англ.] / Х. Гома. 2-е изд. Москва: ДМК Пресс, 2011. − 704с.

6. Ермакова В.И. Сборник задач по высшей математике для экономистов / В.И. Ермакова. Москва: Инфра-М, 2003. − 575 с.

7. Каллихман И.Л., Войтенко М.А. Динамическое программирование в примерах и задачах / И.Л. Каллихман, М.А. Войтенко. Москва: Высшая школа, 1979. − 124 с.

8. Лежнев А.В. Динамическое программирование в экономических задачах / А.В. Лежнев. Москва: БИНОМ. Лаб. знаний, 2010. − 176 с.

9. МакГрат, Майк Программирование на Java / Майк МакГрат; [пер. с англ. М. А. Райтмана]. - 5-е изд. Москва: Э, 2016. − 190 с.

10. Некрасова М.Г. Методы оптимизации и теория управления: учебное пособие / М.Г. Некрасова. Комсомольск-на-Амуре: КНАГТУ, 2007. − 132 с.

11. Окулов С.М. Динамическое программирование / С.М. Окулов, О.А. Пестов. Москва: БИНОМ. Лаборатория знаний, 2012. − 260 с.

12. Самаров К.Л. Математика. Учебно-методическое пособие для студентов по разделу "Элементы теории графов. Динамическое

программирование. Сетевое планирование" / К.Л. Самаров. Москва: Учебный центр "Резольвента", 2009. − 26 с.

13. Шапкин А.С. Математические методы и модели исследования операций / А.С. Шапкин, В.А. Шапкин. Москва: Дашков и Ко, 2017. − 398 с.

14. Ширяв В.И. Исследование операций и численные методы оптимизации: учебное пособие / В.И. Ширяв. Москва: Ленанд, 2017. − 224 с.

Литература на иностранном языке

15. Antoniou A., Lu W.-S. Practical Optimization. Algorithms and Engineering Applications. Springer, 2007, − 675 p.

16. Gembicki F.W. Vector Optimization for Control with Performance and Parameter Sensitivity Indices. Ph.D. Dissertation, Case Western Reserve Univ., Cleveland, Ohio, 1974. – 354 p.

17. Gill P.E., W. Murray, M.A. Saunders, and M.H. Wright. Procedures for Optimization Problems with a Mixture of Bounds and General Linear Constraints. ACM Trans. Math. Software, Vol. 10, 1984. 282 − 298 pp.

18. Heckler M. JavaFX 8: Introduction by Example / M. Heckler. Apress, 2014. – 420 p.

19. Herbert Schildt, Java The Complete Reference, 8th Edition / Herbert Schildt. McGraw-Hill Education, 2011. – 1153 p.

20. Richard L. Halterman, Fundamentals of C++ Programming / Richard L. Halterman, 2018. – 766 p.

21. Richard L. Halterman, Learning to program with Python / Richard L. Halterman, 2011. – 283 p.