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

Институт **математики, физики и информационных технологий**

Кафедра **«Прикладная математика и информатика»**

01.03.02 ПРИКЛАДНАЯ МАТЕМАТИКА И ИНФОРМАТИКА

СИСТЕМНОЕ ПРОГРАММИРОВАНИЕ И КОМПЬЮТЕРНЫЕ ТЕХНОЛОГИИ

БАКАЛАВРСКАЯ РАБОТА

на тему **Реализация алгоритма стохастического программирования**

Студент	П.С. Мавринсий	
Руководитель	Г.А. Тырыгина	
Руководитель аннотации	Н.В. Ященко	

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

«_____» _____ 20___ г.

Тольятти 2017

АННОТАЦИЯ

Выпускная квалификационная работа посвящена вопросу реализации алгоритма стохастического программирования.

Целью работы является максимизация средней прибыли летного парка авиакомпании.

Предметом исследования является максимизация средней прибыли с помощью алгоритма стохастического программирования.

Объектом исследования является экономическая деятельность авиакомпании.

Структура работы представлена введением, тремя главами, заключением и списком литературы.

Во введении определены актуальность темы, цели и задачи, поставленные в работе, а также объект и предмет исследования.

В первой главе рассматриваются модели стохастического программирования и способы их решения. Во второй главе разрабатывается алгоритм решения поставленной задачи. В третьей главе представлен программный код на языке программирования Java. В заключении представлены результаты и выводы о выполненной работе.

Результатом выпускной квалификационной работы будет, является программа, которая способна максимизировать среднюю прибыль авиакомпании.

В работе представлено 1 таблица, 3 рисунка и 1 приложение, список использованной литературы содержит 28 источников. Общий объем выпускной квалификационной работы составляет 72 страницы.

ABSTRACT

The title of the graduation work is «Implementation of Stochastic Programming Algorithms».

The aim is developing the maximizing average profit fleet of the airline.

The subject of the study is the maximization of the average profit using the stochastic programming algorithm.

The object of study is the economic activity of the airline.

The structure of the work includes introduction, three parts, conclusion.

In the first part, models of stochastic programming and methods for their solution are considered. In the second part an algorithm for solving the problem is developed. The third part develops the program interface based on the algorithm.

In the conclusion of this work was the algorithms of stochastic programming. The mathematical model of the problem management summer Park airlines to maximize average profit. On the basis of the mathematical model program code was created in the programming language JAVA. This program has a practical significance for any airline and can be used to maximize average profit.

The result of graduation work is the program that is able to maximize the average profit of the airline.

The work presents 1 tables, 3 figures. The list of references contains 28 sources. The total amount of the graduation work is 72 pages.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Стохастическое программирование это подход, позволяющий учитывать неопределенность в оптимизационных моделях.

В то время как детерминированные задачи оптимизации формулируются с использованием заданных параметров, реальные прикладные задачи обычно содержат некоторые неизвестные параметры. Когда параметры известны только в пределах определенных границ, один подход к решению таких проблем называется робастной оптимизацией. Этот подход состоит в том, чтобы найти решение, которое является допустимым для всех таких данных и в некотором смысле оптимально.

В моделях стохастического программирования используются знания распределения вероятностей для данных или их оценок. Наиболее широко используемой и хорошо изученной является двухступенчатая линейная модель стохастического программирования. На первом этапе принимаются некоторые меры, после чего происходит случайное событие, влияющее на результат первого этапа. На втором этапе можно принять корректирующее решение, которое компенсирует любые нежелательные эффекты, возникающие в результате решения первой стадии.

Актуальность: в этой работе происходит рассмотрение задачи максимизации средней прибыли авиационной компании при наличии некоторых вероятностных ограничений при фиксированном временном расписании полетов. Разработанный алгоритм можно эффективно использовать для решения более общих задач по оптимизации деятельности авиационной компании.

Объект исследования: экономическая деятельность авиакомпании**.**

Предмет исследования: максимизация средней прибыли авиакомпании с помощью алгоритма стохастического программирования.

Цель: максимизация средней прибыли летного парка авиакомпании

Задачи работы:

1. описать математические модели стохастического линейного программирования;
2. проанализировать алгоритм максимизации средней прибыли авиакомпании;
3. на основе полученных алгоритмов создать программу.

Структура работы представлена тремя главами, заключением и списком литературы.

В первой главе рассматриваются модели стохастического программирования и способы их решения. Во второй главе разрабатывается алгоритм решения поставленной задачи. В третьей главе представлен программный код на языке программирования Java. В заключении представлены результаты и выводы о выполненной работе.

В работе представлено 1 таблица, 3 рисунка, список использованной литературы содержит 28 источников. Общий объем выпускной квалификационной работы составляет 72 страницы.

ГЛАВА 1 МОДЕЛИ СТОХАСТИЧЕСКОГО ПРОГРАММИРОВАНИЯ

1.1 Использование стохастического программирования в прикладных задачах

В задачах стохастического оптимального управления модель функционирования системы описывается некоторой функцией, характеризующей эволюцию системы, которая содержит одновременно вектор управления, вектор состояния и набор случайных факторов. В общем случае эта функция может быть нелинейной. Однако исследователи, изучающие динамические системы со случайными воздействиями, как правило, не рассматривают функцию эволюции общего вида, а концентрируются на частных случаях, которые описывают некоторую физическую задачу, например, задачу управления космическим аппаратом. В упомянутых задачах функция эволюции системы содержит при некотором зафиксированном значении вектора текущего состояния скалярное произведение вектора управления на случайный вектор, то есть функция эволюции системы является линейной по случайным величинам, а при реализациях случайных величин является линейной по управлению. Поскольку скалярное произведение является билинейной функцией, то подобную модель функционирования системы можно также назвать билинейной. Рассмотрим задачу оптимального капиталовложения, и задачу корректирования космического аппарата.

В зависимости от целей инвестора в различных работах по задаче оптимального капиталовложения, как правило, рассматриваются следующие критерии оптимальности портфеля ценных бумаг: логарифмический критерий (максимальная средняя скорость роста капитала), впервые исследованный в работе Дж. Келли [27] вероятностный критерий (вероятность того, что капитал превысит некоторый наперед заданный желаемый уровень), изучению различных свойств которого посвящена монография Ю.С. Кана и А.И. Кибзуна [8] VaR-критерий (максимальный уровень капитала, гарантированный с

заданным уровнем надежности), одной из первой работ по которому является работа Ф. Джориона [26] CVaR-критерий (максимальное среднее значение капитала, если капитал инвестора окажется ниже некоторого гарантированного уровня).

Авторы, исследующие логарифмический критерий в задаче формирования портфеля ценных бумаг, как правило, рассматривают только два финансовых инструмента: без рисковый актив, имеющий нулевую дисперсию доходности, и рисковый актив, имеющий определенное распределение. Были приведены оптимальные стратегии в случае, когда рисковый актив имеет равномерное и нормальное распределение доходности. Если же рисковых активов больше одного, то для получения критериальной функции необходимо выполнить много громоздких вычислений. По этой причине вместо непосредственного поиска критериальной функции предлагается оптимизация некоторой функции, аппроксимирующей критериальную. Так же был найден аналитический вид приближенного решения, полученный при помощи разложения критериальной функции в ряд Тейлора.

В рассматриваемой задаче с дискретным распределением доходностей, критерием в форме математического ожидания доходности портфеля ценных бумаг в некоторый будущий момент времени и с не большей чем заданная вероятностью того, что доходность окажется ниже некоторого зафиксированного уровня. Исходная задача была сведена к задаче смешанного целочисленного линейного программирования. В ней максимизировалась линейная по управлению функция, а вероятностное ограничение состояло из произвольного числа неравенств. В работе использовалось дискретное распределение доходностей, рассматривался вероятностный критерий и ограничение на среднюю доходность портфеля. Как и исходная задача была сведена к задаче смешанного целочисленного линейного программирования. В работе предлагается оптимальная стратегия по вероятностному критерию, когда доходности имеют нормальное распределение, параметры которого определяются сценариями с неизвестными заранее вероятностями.

Квантильный или VaR-критерий получил широкое распространение не только в теории, но и на практике. В частности, VaR-критерий был включен в как способ оценки рисков ведения банковской деятельности. Если же рассматривать теоретические аспекты оптимального капиталовложения, то в любой произвольной инвестиционной стратегии сопоставляется наименьшее значение VaR-критерия, получаемое как решение задачи полуопределенного программирования, в случае если о распределении доходностей известны только вектор математических ожиданий и ковариационная матрица доходностей. В работе Ю.С. Кана [9] была найдена оптимальная по квантильному критерию стратегия в случае нормального распределения доходностей и отсутствия ограничения на то, что нельзя брать деньги в долг, т.е. в случае запрета операций «shortsales».

Поскольку VaR-критерий не показывает уровень капитала инвестора вне заданного уровня надежности, то часто выбирают CVaR-критерий или рассматривают ограничение, связанное с CVaR. Причем в отличие от VaR CVaR принадлежит классу когерентных мер риска, введенному в работе. В задачу с CVaR критерием и ограничением на среднюю доходность портфеля предлагают решать приближенно: исходная задача сводится к задаче линейного программирования при помощи дискретизации вероятностной меры. Если же доходности распределены нормально, то находится аналитический вид критерия. Предлагается алгоритм решения модифицированной задачи Марковица, где в отличие от классической постановки максимизируется квантиль некоторого уровня и имеется ограничение на значение интегральной квантили.

Следует отметить, что для формирования портфеля ценных бумаг могут использоваться и другие критерии. Использование одношаговых моделей может привести к «биржевому парадоксу», когда при многократном использовании одношаговой стратегии в среднем капитал инвестора стремится к бесконечности, а вероятность разорения стремится к единице. Использование двух шаговой или многошаговой моделей может позволить избежать данного

парадокса, поскольку при использовании таких моделей имеется возможность в некоторый момент времени инвестиционного горизонта ребалансировать портфель ценных бумаг и, таким образом, учесть изменяющуюся ситуацию на рынке.

Учет возможности ребалансировки портфеля ценных бумаг существенно усложняет процесса поиска решения. Поэтому исследователи, использующие в качестве критерия оптимальности вероятность или VaR, как правило, рассматривают двух шаговые задачи с одним рисковым активом на каждом шаге. В работе А.И. Кибзуна [7] доходность рисковой ценной бумаги на каждом шаге описывалась равномерным распределением с фиксированной и одинаковой левой границей носителя меры, равной минус единице, а правые концы носителя доходности на первом и втором шагах отличались и их можно было варьировать. При этом оптимальное управление портфелем ценных бумаг выбиралось таким образом, что стратегия первого шага совпадала со стратегией второго шага. Доходность рисковой ценной бумаги также полагалась равномерно распределенной, однако управление на втором шаге искалось в классе позиционных стратегий, то есть в зависимости от текущего значения капитала на момент ребалансировки. На каждом шаге распределение доходности полагалось усеченной нормально распределенной случайной величиной, а управление на втором шаге, полученное при помощи доверительного метода, зависело от реализации доходности на первом шаге. В работе А.И. Кибзуна [12] была приближенно решена задача многошаговой квантильной оптимизации и найдено некоторое позиционное управление в случае одной рисковой и одной без рисковой ценной бумаги на каждом шаге, также, как и в, при помощи доверительного метода. Отметим, что двух шаговые задачи стохастического оптимального управления похожи по структуре функционирования системы на двухэтапные задачи стохастического программирования. Однако если в двух шаговой задаче имеется два вектора управления и два вектора случайных факторов, то в двухэтапной задаче также имеется два вектора управления, но лишь один вектор случайных факторов.

Поэтому можно сделать вывод о том, что двух шаговые задачи несколько сложнее двухэтапных. При этом большая часть исследований двухэтапных задач проводится в случае использования математического ожидания в качестве критерия. Также представлены различные постановки двухэтапных задач стохастического линейного программирования, в которых представлены различные методы решения двухэтапных задач стохастического линейного программирования.

Параллельно с двухэтапными задачами развиваются двухуровневые задачи стохастического программирования, главным отличием которых от двухэтапных является то, что в двухуровневых задачах стратегии на первом и втором этапах могут быть подчинены разным целям, а в двухэтапных – единой. Общую постановку двухуровневой задачи стохастического программирования можно найти в работе Р.М. Ковачевича [28].

Если же использовать критерии, связанные с моментными характеристиками случайных величин, то удается получить результаты для многошаговых задач, в которых предполагается случай более чем одной коррекции инвестиционного портфеля на протяжении инвестиционного горизонта. Так, в качестве критерия использовалась взвешенная дисперсия, а именно сумма дисперсий капиталов, вкладываемых в произвольное число активов, в каждый момент времени инвестиционного горизонта с некоторыми заданными весами; средний капитал в терминальный момент времени должен был быть выше некоторой заданной отметки. Предшествующее выбранные управления, зависящие от момента времени, корректировались линейно в зависимости от того, насколько реализация доходностей в прошлый момент времени отличается от средних доходностей. В качестве критерия использовалось математическое ожидание квадрата разности капитала и желаемого уровня капитала в терминальный момент времени. В случае отсутствия транзакционных издержек, то есть платы за покупку или продажу активов, и наличия только одного ограничения на управление, заключающегося в том, что капитал, получаемый в результате торговли, в каждый момент

времени вкладывается полностью в финансовые инструменты, было найдено оптимальное управление на каждом шаге, выбираемое в классе позиционных стратегий, которое оказалось линейным по состоянию. Если же имеются дополнительные ограничения на управления, то предлагается субоптимальное решение, получаемое при помощи проектирования полученного решения на требуемое множество допустимых управлений.

Часто для формирования портфеля ценных бумаг в многошаговых задачах используется математическое ожидание от той или иной функции полезности, являющейся решением уравнения от капитала инвестора в терминальный период времени. Так, например, было найдено оптимальное аналитическое решение при использовании экспоненциальной функции полезности, а доходности подчинялись авторегрессии первого порядка с гауссовским белым шумом. Так же предлагались процедуры для поиска оптимального управления для различных функций полезности, в том числе логарифмической и степенной.

Другой физической задаче задаче управления космическими аппаратами, также посвящено множество работ с различными постановками задачи в том числе в непрерывном времени. Часто в качестве критерия оптимальности выступает математическое ожидание. Рассмотрена задача оптимальной коррекции бокового отклонения космического аппарата в области случайных сил, где динамика системы, описываемая дифференциальным уравнением 2-го порядка, на управление налагались интегральные ограничения, а критерием качества является математическое ожидание квадрата фазовых координат в некоторый конечный момент времени. Рассматривалась проблема коррекции высоты и скорости в предположении ошибок тестового импульса, независимо от его размера, критерием была ожидаемая полная коррекция скорости. Однако для поиска оптимального управления с использованием доверительного метода, считается, что возмущения являются нормальными. В то же время производительность коррекции ошибок не может иметь распределение Гаусса. В работе Дж. С. Кана [1] рассматривалась проблема оптимальной

двухимпульсной коррекции спутника с помощью ошибок на высоком импульсе двигателя, выполняющих импульс, распределенных по единому закону. Для поиска оптимального управления был применен метод динамического программирования. В работе А. И. Кибзуна [12] решает проблему коррекции положения стохастической системы по критерию квантиля.

Рассматривая выбор критерия оптимальности, следует отметить, что, обычно, интересен для инвестора определенный уровень доходности, например, 10% годовых, или же порог капитала, который необходимо достичь, следовательно, важно использовать вероятностный критерий. Полет должен осуществлять коррекцию с вероятностью, близкой к единице, то есть лучше использовать вероятностный критерий контроля качества, а не средний. Поэтому для поиска оптимального управления лучше использовать вероятностный критерий. Следует отметить, что вероятность применяется в других практических задачах, таких как оценка различных рисков на железнодорожном транспорте, как указано в национальном.

Однако при использовании вероятностного критерия в задаче поиска оптимального управления с использованием метода динамического программирования как в задаче корректирования космического аппарата, так и в задаче оптимального капиталовложения сопряжено с большими трудностями в поиске аналитического выражения функций Беллмана. Также можно было бы воспользоваться доверительным методом, вычислив для различных уровней надежности гарантированное значение капитала, после чего выбрать подходящее значение капитала и соответствующую ему инвестиционную стратегию в качестве решения. Однако, для различных распределений уже для случая одного рискового актива на каждом шаге трудно подобрать доверительное множество, близкое к оптимальному, не говоря уже о большем количестве рисковых активов на каждом шаге.

К тому же на практике доходность в отличие от реализации случайной величины, подчиняющейся нормальному закону, не может оказаться меньше минус единицы. Поэтому в задаче оптимального капиталовложения логичным

представляется использование распределения с ограниченным носителем, например, можно использовать равномерное распределение. Из работы А.И. Кибзуна [8] известно, что равномерное распределение при минимальных предположениях о виде закона распределения оказывается наихудшим по значению критериальной функции для лица, принимающего решения. Более того, равномерное распределение встречается в задаче оптимального капиталовложения: равномерное распределение является одним из немногих, «полезных для приложений задачи распределения активов», рассматриваются характеристики портфеля ценных бумаг, имеющего равномерное распределение. Однако равномерное распределение также имеет свои недостатки: оптимальное значение критерия может оказаться неоправданно невысоким. Кроме того, плотность равномерного распределения симметрична относительно математического ожидания, что не обязательно соответствует истине. С другой стороны, оптимальный портфель, полученный с использованием равномерного распределения доходностей, можно считать гарантирующим.

1.2 Математические модели стохастического программирования.

В стохастическом программировании рассматриваются проблемы принятия решений при случайных факторах, которые нужно учитывать в математических моделях.

Далее рассмотрим задачу нелинейного программирования:

Требуется найти вектор X, для которого:

$$f(X) \to \min \quad (1)$$

при ограничении:

$$g_i(X) \le 0, i = \overline{1,m}, \quad (2)$$

Задачи стохастического программирования появляются в случаях, когда функции $f(X)$, $g_i(X)$ находятся в зависимости от случайных параметров ω. При этом подразумевается, что ω - элемент пространства состояний природы (или пространства случайных параметров) Ω. Тогда задача стохастического программирования формулируется следующим образом:

Минимизировать:

$$f(X, \omega) \quad (3)$$

при следующих условиях:

$$g_i(X, \omega) \le 0, i = \overline{1,m}\ , \quad (4)$$

Постановка задач по стохастическому программированию значительно зависит наличия возможности во время выбора решений осуществить уточнение состояние природы ω через некоторые наблюдения. По этой причине различают задачи перспективного и оперативного стохастического программирования.

В задачах оперативного стохастического программирования принимается решение по окончанию определенного эксперимента над состоянием самой природы ω, которое зависит от результатов проведенного эксперимента и является случайным вектором. Эти задачи возникают, к примеру, при быстром планировании, медицинской диагностике. В задачах по перспективному стохастическому программированию принятие решения *x* необходимо для того, чтобы провести различные наблюдения за состоянием природы и по этой причине является детерминированным. Эти проблемы появляются в перспективном технико-экономическом планировании, в задачах проектирования, когда системным параметрам необходимо выбирать

определенную детерминированную ценность, в основе которой лежит расчет на определенном диапазоне возможных возмущений.

Задачи по стохастическому программированию обычно задаются в одной из следующих форм:

минимизировать:

$$M_\omega f(X,\omega) = F(X) \qquad (5)$$

при условиях:

$$M_\omega g_i(X,\omega) = G_i \le 0, i = \overline{1,m} \quad , \qquad (6)$$

где M_ω – операция математического ожидания;

минимизировать

$$P\{f(X,\omega) \ge a\} \qquad (7)$$

при ограничениях

$$P\{g_i(X,\omega) \le 0\} \ge P_i, i = \overline{1,m}, \quad , \qquad (8)$$

где a, P_i – некоторые числа; P – вероятность.

Возможные комбинации задач (5), (6) и (7), (8).

К примеру, необходимо осуществить поиск минимума (5) при условиях (8) или минимума (7) при условиях (6). Не учитывая кажущееся различие в задачах (5), (6) и (7), (8), они могут быть сведены к общей формулировке, вида (5), (6). Для этого требуется ввести характеристические функции:

$$h_0(X,\omega) = \begin{cases} 1, & \textit{если} \quad f(X,\omega) \ge a \quad ; \\ 0, & \textit{в противном случае}; \end{cases} \qquad (9)$$

$$h_0(X,\omega) = \begin{cases} 1, & \textit{если} \quad g_i(X,\omega) \ge a \quad ; \\ 0, & \textit{в противном случае} \; ; \end{cases} \qquad (10)$$

для которых:

$$M_\omega h_0(X,\omega) = P\{f(X,\omega) \ge a\}; M_\omega h_i(X,\omega) = P\{g_i(X,\omega) \le 0\} \qquad (11)$$

Задача (7), (8) приводится к следующему виду

Минимизировать:

$$M_{\omega} f_0(X,\omega) \quad (12)$$

при условии:

$$M_{\omega} f_0(X,\omega) \geq P_i, i = \overline{1,m} \quad (13)$$

Существует 2 основных подхода к процессу решения задач по стохастическому программированию:

1) непрямые методы, заключающиеся в нахождении функций $F(X)$, $G_i(X)$ и решении эквивалентной задачи НП вида (5), (6);

2) прямые методы стохастического программирования, которые основаны на информации о значении функций $f(X,\omega)$, $g_i(X,\omega)$, получаемой после проведения экспериментов.

Одноэтапные задачи стохастического программирования включают в себя задачи, где решения принимаются, основываясь на известных стохастических характеристиках распределения случайных параметров условий задачи до наблюдения за их реализациями. Это должно быть лучшим в среднем смысле, решением.

Формулирование задач стохастического программирования различается по трем характеристикам:

1) характер решений;

2) выбор показателя качества решения (критерия);

3) метод декомпозиции ограничений в задаче.

Ограничение на функции просмотра. В проблеме стохастического программирования обычно берут такие функционалы, как ожидание или дисперсия целевой функции, или вероятность превышения целевой функции некоторого порога.

Задача с целевой функцией формы $M\{c^T x\}$ называют М-моделями, задачи, где необходимо минимизировать дисперсию $M\{(c^T x \geq \overline{c}^T)^2\}$ называют

V-моделями, стохастические задачи, где максимизируется вероятность $P\{c^T x \geq \overline{c}^T x = k_0\}$, - P-моделями.

В последнюю группу моделей входят задачи, где нужно минимизировать порог *k*, который не должен быть больше заданной вероятностьюα, к примеру:

минимизировать *k* при условии

$$P\{c^T x \leq k\} \geq \alpha. \qquad (14)$$

Ограничения могут быть представлены в одной из следующих форм:

а) $P\{\sum a_i j x_j \leq b_i\} \geq a_i \; i = 1,2,.....,m; 0 \leq a \leq 1;$

б) $P\{Ax \leq B\} \geq a, 0 \leq a \leq 1;$

в) $P\{\sum_{j=1}^{n} a_{ij} x_j \leq b_{ik}, i_k \in I\} \geq a_k \;, \quad 0 \leq a_k \leq 1; k = \overline{1,K}$

Рассмотрим варианты моделей задач стохастического программирования в один этап.

1. Пусть задана задача линейного стохастического программирования с вероятностными ограничениями типа а):

максимизировать $M\{c^T x\}$ (15)

при условиях:

$$P\{\sum_{j=1}^{n} a_{ij} x_j \leq b_i\} \geq a_i \;, \; i = \overline{1,m}; \qquad (16)$$

$$x_j \geq 0, \; j = \overline{1,n}. \qquad (17)$$

При детерминированной матрице $A = |a_{ij}|$ и случайном векторе $b = |b_i|$ задача (15)–(17) сводится к эквивалентной детерминированной задаче ЛП следующим образом.

Пусть $P(b_1, b_2, ..., b_m)$ – совместная плотность распределения составляющих b_i случайного вектора b. Находим плотность распределения b_i:

$$P(b_i) = \underbrace{\int_{-\infty}^{\infty} .... \int_{-\infty}^{\infty}}_{m-1} p(b_1, b_2 ... b_m) db_1 db_2 ... db_{i-1} db_{i+1} ... db_m \qquad (18)$$

Вычислим $\tilde{b}_i$ из уравнения:

$$\int_{bi}^{\infty} P(b_i db_i = \alpha_i, i = \overline{1,m} \ . \tag{19}$$

Если решение уравнения (19) не единственное, то в качестве $\tilde{b}_i$ выберем наибольший корень.

Получается, что при этом условия (12) эквивалентны неравенствам:

$$\sum_{j=1}^{n} a_{ij} x_j \le \tilde{b}_i,$$

где $\tilde{b}_i$ удовлетворяет соотношениям (19). Это значит, что задаче стохастического программирования (10), (12), (17) будет эквивалентна следующая детерминированная задача ЛП:

$$\bar{c}^T x \to \max \tag{20}$$

при условиях:

$$\sum_{j=1}^{n} a_{ij} x_j \le \tilde{b}_i, i = \overline{1,m} \ , \tag{21}$$

$$x_j \ge 0 \ , \tag{22}$$

где $\bar{c} = M$; $\tilde{b}_i$ – корень уравнения $F(\tilde{b}_i) = 1 - \alpha_i$, или $\tilde{b}_i = F^{-1}(1-\alpha_i)$, $F_i(b_i)$ – функция распределения случайной величины b_i.

Для стохастической задачи (10), (12), (17) с детерминированной матрицей *A* можно записать двойственную задачу с вероятностными ограничениями.

Рассмотрим задачу:

$$\tilde{b}^T y \to \min \tag{23}$$

при условиях:

$$P(A^T y \ge c) \ge \beta, \tag{24}$$

$$y \ge 0. \tag{25}$$

Ее решение определяется в виде детерминированного вектора. Пусть $G_j(\xi)$ – функция распределения случайного коэффициента C_j функции (10), т.е.

$G_j(\xi) = P\{C_j \le \xi\}$. Если $G_j(\xi) = \beta_j$, то запись $\xi = G_j^{-1}(\beta_j)$ эквивалентна записи $\xi = \min\{y \mid G_j(y) \ge \beta_j\}$. Задача (23)–(25) может быть в виде

$$\tilde{b}^T y \to \min \tag{26}$$

при условии:

$$A^T y \ge G^{-1}(\beta), y \ge 0\ . \tag{27}$$

Сравнивая данную задачу с исходной (10), (12), (17), убеждаемся, что при $\beta = G(\tilde{C})$ следующие 2 одноэтапные задачи стохастического программирования с вероятностными ограничениями - двойственная пара:

$$G^{-1}(\beta)x \to \max, P\{A_x \le b\} \ge a, x \ge 0; \tag{28}$$

$$P^{-1}(1-\alpha)y \to \min, P\{A^T y \ge c\} \ge \beta, Y \ge 0. \tag{29}$$

2. Рассмотрим общий случай, если A – это случайная матрица, то элементы матрицы A, составляющие вектора b – не зависят друг от друга, нормально распределенные случайные величины: $a_{ij} \in N(\bar{a}_{ij}, \sigma_{ij}^2), b_i \in N(\tilde{b}_i, \theta_i^2)$, т.е. a_{ij} – случайная нормально распределенная величина с математическим ожиданием $\bar{a}_{ij}$ и дисперсией $\sigma^2{}_{ij}$. Пусть, кроме этого, при (16), $a_i \ge 0{,}5 i = \overline{1,m}.$

Отобразим, что при данных предположениях стохастическая задача (10), (12), (17) будет принимать вид детерминированной задачи выпуклого программирования с целевой линейной функцией, а также с квадратичными ограничениями.

На самом деле, при принятых допущениях невязка i го условия – случайная величина $\delta_i(x) = \sum_{j=1}^{n} a_{ij} x_j - b_j$ – нормально распределенная величина с математическим ожиданием:

$$\delta_j(x) = \sum_{j=1}^{n} \bar{a}_{ij} x_j - \bar{b}_j \tag{30}$$

и с дисперсией:

$$\sigma_j^3(x) = \sum\nolimits_{j=1}^{n} \sigma_{ij} x^3 + \theta\,, \tag{31}$$

То есть:

$$\delta_j(x) \in N\left(\delta_j(x), \sigma^3(x)\right). \tag{32}$$

Тогда условия $P\left\{\sum_{j=1}^{n} a_{ij}x_j \le b_i\right\} \ge a_i$ будут эквивалентны следующим неравенствам:

$$P\{\delta_i(x) \le 0\} \ge a_i, i = \overline{1,m}, \quad (33)$$

или

$$\frac{1}{\sqrt{2\pi}\sigma_i(x)} \int_{-\infty}^{0} \exp\left\{-\frac{(\xi - \bar{\delta}(x))^2}{2\sigma_i^3(x)}\right\} d\xi \ge a_i, i = \overline{1,m}. \quad (34)$$

Обозначив $\Phi(t) = \frac{1}{\sqrt{2\pi}} \int_{-\infty} e^{-\frac{\xi^2}{2}} d\xi$, последнее неравенство (34) отобразим в виде: $\Phi\left(-\frac{\bar{\delta}_i(x)}{\sigma_i(x)}\right) \ge a_i$,

Откуда: $\bar{\delta}(x) + \Phi^{-1}(a_i)\sigma_i(x) \le 0$.

Учитывая выражения для $\bar{\delta}(x)$, $\sigma_i(x)$, получим

$$\Phi^{-1}(a_i)\left(\sum_{j=1}^{n} \sigma_{ij}^2 x_j^2 + \theta_i^2\right)^{\frac{1}{2}} \le \bar{b}_i - \sum_{j=1}^{n} \bar{a}_{ij} x_j \text{ , i= } i = \overline{1,m} \quad (35)$$

Опираясь на допущения $a_i \ge 0.5$. По этой причине $\Phi^{-1}(a_i) \ge 0$, и можно убедиться, что область, которая определяется условиями (35) является выпуклой.

Такой же результат будет, если случайные элементы строки i го условия a_{ij} между собой коррелированы.

Введем обозначения:

$$\nu_{ij} M\{b_i - \bar{b}_i(a_{ij} - \bar{a}_{ij})\}; \ \nu_{ijk} = M\{(a_{ij} - \bar{a}_{ij})(a_{ik} - \bar{a}_{ik})\} \quad (36)$$

Тогда получим:

$$\Phi^{-1}(a_i)\left(\sum_{i} \sum_{k} \nu_{ijk} x_j x_k - 2\sum_{j} \nu_{ij} x_j + \theta_i^3\right)^{\frac{1}{3}} \le \bar{b}_i - \sum_{i=1}^{n} \bar{a}_{ij} x_j \quad (37)$$

Если матрица $V_{i=}\|v_{ijk}\|, j,k=\overline{1,n}$ определенная положительно, и $a_i \geq 0.5$, $i=\overline{1,m}$, то допустимое множество решений, задаваемых (37), будет являться выпуклым.

Итак, при допущениях, принятых выше, с вероятностными ограничениями линейная стохастическая задача (10), (12), (17) сводится к детерминированной задаче по выпуклому программированию вида:

$$\sum_{j=1}^{n} \bar{c}_j x_j \to \max \tag{38}$$

с условием:

$$\Phi^{-1}(a_\text{ʃ})\left(\sum_{\text{ʃ}} \sum_{k} v_{ijk} x_j x_k - 2\sum_{j} v_{ij} x_j + \theta_\text{ʃ}^3\right)^{\frac{1}{3}} \leq \bar{b}_\text{ʃ} - \sum_{\text{ʃ}=1}^{n} \bar{a}_{\text{ʃ}\,j} x_j \qquad i=\overline{1,m}. \tag{39}$$

3. Рассмотрим задачу по стохастическому программированию, которая задана Р-моделью:

минимизировать k (40)

при условии

$$P\left\{\sum_{j=1}^{n} c_j x_j \leq k\right\} \geq \alpha_0 . \tag{41}$$

Будем учитывать, что коэффициенты c_i, $j=\overline{1,n}$, распределены нормально с математическим ожиданием $\bar{c}_j$ и корреляционной матрицей $C=\left|c_{ij}\right|$, где $c_{tj}=M\left\{(c_t-\bar{c}_t)(c_j-\bar{c}_j)\right\}$. Линейная форма $c^T x=\sum_{j-1}^{n} c_j x_j$ при принятых допущениях распределена с математическим ожиданием $\bar{c}^T x$ и дисперсией $\sum_{t=1}^{n}\sum_{j=1}^{n} c_{ij} x_t x_j$. По этой причине соотношение (41) вида:

$$\Phi\left(\frac{k-\sum_{j=1}^{n} \bar{c}_j x_j}{\sqrt{\sum_{t=1}^{n}\sum_{j=1}^{n} c_{tj} x_t x_j}}\right) = \alpha_0 . \tag{42}$$

Отсюда значит, что минимизация k с условием (41) будет эквивалентна минимизации:

$$k(x) = \sum_{j=1}^{n} \bar{c}_j x_j + \Phi^{-1}(\alpha_0)\sqrt{\sum_{t=1}^{n}\sum_{j=1}^{n} c_{tj} x_t x_j} \ . \qquad (43)$$

При $\alpha \geq 0$ k(x) - выпуклая вниз функция по переменным x_j. Получается, что при допущениях задаче стохастичного программирования вида:

минимизировать k (44)

при условиях:

$$P\{\bar{c}^T x \leq k\} \geq a_0 \ , \qquad (45)$$

$$p\{\sum_{j=1}^{n} a_{ij} x_j \leq b_i\} \geq a_i \ , \ i = \overline{1,m} \ , \qquad (46)$$

соответствует следующий детерминированный эквивалент:

$$k(x) = \sum_{j=1}^{n} \bar{c}_j x_j + \Phi^{-1}(a_0)\sqrt{\sum_{t=1}^{n}\sum_{j=1}^{n} c_{tj} x_t x_j} \to \min \qquad (47)$$

С условием:

$$\Phi^{-1}(a_i)(\sum_t \sum_j v_{itj} x_t x_j - 2\sum_j v_{ij} x_j + \theta_i^2)^{\frac{1}{2}} \leq \bar{b}_i - \sum_{j-1}^{n} \bar{a}_{ij} x_j \ , \ i = \overline{1,m} \qquad (48)$$

Задача (47), (48) – это задача по выпуклому программированию. Для ее решения можно применить теорему Куна-Таккера, или же один из вариантов метода допустимых направлений и другие методы НП.

1.3 Постановка задачи реализации алгоритма управления летним парком авиакомпании

Существует большое количество работ, в которых при решении различного рода экономических задач используются методы стохастического программирования. Исторически сложилось два направления в стохастическом программировании при моделировании экономических систем. К первому относятся модели, в которых оптимизируется какой-либо средний показатель эффективности работы экономической системы, например, средняя прибыль или средние издержки компании. Как правило, это двухэтапные задачи с

критерием в форме математического ожидания. Второе направление содержит модели, позволяющие получать гарантированный с заданной вероятностью результат. К ним относятся задачи с вероятностными ограничениями, а также двухэтапные и одноэтапные задачи квантильной оптимизации. Достаточно редко встречаются модели, где критерий в виде математического ожидания и вероятностные ограничения используется совместно.

В этой работе анализируется цель максимизации среднего дохода авиакомпании при существовании ряда вероятностных ограничений при постоянном временном расписании полетов. Задача осложнена целочисленностью переменных оптимизации. В данной работе даётся краткое подтверждение построенной модели и приводится алгоритм нахождения субоптимального решения. Рассматриваемая модель не претендует на абсолютную уникальность и не является общей моделью для оптимизации данного процесса работы авиакомпании. Разные маркетинговые поиски эффективности использования маршрутов и управления пассажиропотоками учитываться не будут при создании этой модели, а данные параметры (состав самолетного парка и временное расписание) будут считаться фиксированными, созданный алгоритм будет эффективно применен при решении более общих задач оптимизации деятельности летного парка в целом.

Рассмотрим работу парка авиакомпании, который работает на рынке пассажирских авиаперевозок. Предположим, что летный парк компании состоит из некоторого числа разных самолетов, благодаря которым компания будет обслужить заданные рейсы. Аэропорт, в котором располагается главный офис компании, и все службы занимающиеся ремонтными работами, называется базовым. Все рейсы будут совершаться по составленной схеме: Базовый аэропорт - Аэропорт назначения - Базовый аэропорт. Расписание рейсов имеет повторяющееся действия, а именно через неопределенный промежуток времени все рейсы будут совершаться заново. Длительность одного цикла называется базовым периодом (как правило это неделя). В один и тот же аэропорт в течение базового периода может совершаться несколько

рейсов. Время вылета каждого рейса будет зафиксировано. Эти рейсы устанавливаются временем вылета, расчетным временем выполнения, продолжительностью полета, ценой билета пассажира и расходами, которые связанны с выполнением рейса. Предположим, что на каждый рейс будет приходить некоторое количество пассажиров. Авиакомпания состоит из самолетов различной вместимости, каждый из которых определяется определенными эксплуатационными расходами. Также кроме вместимости каждый самолет устанавливаются себестоимостью, расходами на обслуживаемый ремонт, изношенностью, а также принадлежностью к какой-либо авиакомпании (самолет, который находится в собственности или взятый в аренду). После того как самолёт совершил рейс в течение какого-то времени (так называемое “время послеполетного обслуживания") самолеты должны пройти контроль на выявление и устранение поломок которые произошли во время рейса. Время на устранение таких поломок будет зависеть от конкретного самолета и неисправностей, которые произошли во время рейса. Поломки можно разделить на два типа: которые влияют на дальнейшую возможность использования самолёта (крупные поломки) и не влияющие (незначительные поломки). Незначительные поломки могут либо устраняться в течение некоторого времени послеполетного обслуживания, либо накапливаться. Накопленные незначительные поломки должны быть устранены в процессе плановых ремонтов самолетов, которые предусмотрены условиями эксплуатации воздушных аппаратов. Расписание плановых ремонтных работ, а так же время на устранение поломок известно заранее. Если количество мелких поломок превышает допустимую норму, то технический контроль должен поставить самолёт на внеплановый ремонт. Продолжительность внеплановых ремонтов и время становится известно за неделю до проведения работ. При значительных поломках самолет должны снять на время проведения ремонта со своих запланированных рейсов, которые будут выполняться арендованным у других компаний самолетов или самолетами собственного пользования, после вынужденного пересмотра лётного расписания (изменение рейсов). Время на

устранение поломки и продолжительность ремонта неизвестна. Рейс может отправляться с неопределенной задержкой по времени, которая вызвана разными причинами: некачественной работой сотрудников аэропорта (например, затянувшимся досмотром пассажиров) или же технической поломкой самолета. Также, самолёт может задержаться из-за несвоевременного прибытия в аэропорт по причине задержки на предыдущем рейсе. Время задержки может быть от одной секунды до нескольких часов. Лётное расписание нужно составлять так, чтобы авиакомпания могла получала, как можно больше прибыли. Так как рейсы цикличны и повторяются каждую неделю, то нужно составить расписание только на одну неделю. Так же нужно стремиться к максимальному уменьшению продолжительности и числа задержек, которые возникают из-за непредвиденных поломок самолетов, связанных с поздним прибытием самолета в аэропорт (на другие задержки лётное расписание никакого влияния не оказывает). Требование к уменьшению задержек возникают из-за их отрицательного воздействия на престиж компании, в следствии чего уменьшается прибыть.

ГЛАВА 2 РАЗРАБОТКА АЛОГОРИТМА МАКСИМИЗАЦИИ СРЕДНЕЙ ПРИБЫЛИ ЛЕТНЕГО ПАРКА АВИАКОМПАНИИ

2.1. Математическая модель функционирования летного парка авиакомпании

В разработанной математической модели детерминированные такие показатели, как:

K – общее число самолетов компании;

L – число рейсов, совершаемых за базовый период времени (число вылетов из базового аэропорта). Все рейсы нумеруются в порядке их вылета из базового аэропорта;

ω – базовый период времени (неделя);

c_{ik} – себестоимость полета k-го самолета на i-м рейсе, $k=\overline{1,K}$, $i=\overline{1,L}$;

τ_i – время вылета i-го рейса из базового аэропорта, $i=\overline{1,L}$;

z_i – количество посадочных мест в k-м самолете (вместимость), $k=\overline{1,K}$;

p_i – средняя цена билета (средний тариф) на одного пассажира на i-м рейсе, $i=\overline{1,L}$.

Чтобы рассчитать стоимость операций, модель была разработана для ранжирования всех самолетов для эффективного использования с учетом таких факторов, как частота основных сбоев самолета, стоимость его обслуживания, стоимость аренды, лизинговые платежи и пр. Чтобы описать эту модель, мы вводим следующие дополнительные обозначения:

t_i^0 – продолжительность полета на i-м рейсе, $i=\overline{1,L}$;

$\tilde{t}_{ik}$ – расчетное время выполнения i-го рейса k-м самолетом, $k=\overline{1,K}$, $i=\overline{1,L}$;

c_k^0 – часть себестоимости транспортного средства за рассматриваемый базовый период времени, который не зависит от эффективности эксплуатации, $k=\overline{1,K}$ (стоимость собственного самолета, арендные и лизинговые оставляющие для самолета, который был взят в лизинг и пр.);

c_{ik}^1 – расходы на использование k-го самолета на i-м рейсе, не зависящие от продолжительности полета, $k=\overline{1,K}$, $i=\overline{1,L}$ (сборы в аэропорту и т.п.);

c_{ik}^2 – расходы на использование k-го самолета на i-м рейсе, зависящие от продолжительности полета, $k=\overline{1,K}$, $i=\overline{1,L}$ (топливо, часть арендных или лизинговых выплат для самолета, взятого в лизинг, бортпитание, з/п экипажа и т.п.);

c_k^3 – средняя стоимость ремонта, приходящаяся на час полета для k-го самолета, $k=\overline{1,K}$;

c_k^4 – стоимость аренды самолета другой компании на i-й рейс при отсутствии своих свободных исправных самолетов в базовом аэропорту, $i=\overline{1,L}$;
c_k^5 – средняя стоимость часа ремонта k-го самолета, $k=\overline{1,K}$;

q_k – вероятность того, что k-й самолет окажется неисправным к моменту начала текущего рейса, $k=\overline{1,K}$;

p_k – вероятность того, что k-й самолет будет иметь крупную поломку после очередного рейса, $k=\overline{1,K}$.

В дополнение к детерминированным переменным в разработанной модели учитывались случайные факторы, влияющие на производительность компании. К ним относятся: летательный аппарат с временной поддержкой полета, отказ самолета, значительный ущерб авиации, ремонт самолетов после повреждения, а также пассажирские перевозки. После обработки статистической информации сделаны следующие предположения о распределениях этих случайных величин.

Y_i – случайная величина, имеющая экспоненциальное распределение с параметром λ_i и характеризующая продолжительность задержки самолета на i-м рейсе, произошедшей по вине служб аэропорта, $i = \overline{1, L}$;

V_k – случайная величина, имеющая распределение Бернулли с параметром q_k и характеризующая неисправность k-го самолета к началу рейса, $k = \overline{1, K}$;

T_k – случайная величина, имеющая распределение Бернулли с параметром m_k и характеризующая крупную поломку k-го самолета после рейса, $k = \overline{1, K}$;

W_k – случайная величина, имеющая распределение Вейбулла и характеризующая продолжительность устранения поломки, обнаруженной на k-м самолете, $k = \overline{1, K}$;

$t_{ik} = \tilde{t}_{ik} + Y_i$ – фактическое время выполнения i-го рейса k-м самолетом с учетом случайного времени задержки Y_i, $t \hat{=} \{t_{ik}\}$, $k = \overline{1, K}$, $i = \overline{1, L}$;

X_i^1, X_i^2 – пассажиропоток на i-м рейсе соответственно в аэропортах отправления и назначения, $i = \overline{1, L}$. Заметим, что случайные величины X_i^1, X_i^2 могут значительно варьироваться в параметрах закона распределения из-за ряда факторов, в частности, наличия транзитных потоков пассажиров. Например, от базового аэропорта до порта назначения, в среднем, может летать меньше пассажиров, чем обратно.

Из предположений о законах распределения используемых случайных величин являются результатом обработки большого статистического материала с использованием специальной методики, описание которой не является предметом этой работы. Однако следует отметить, что предлагаемое распределение случайных величин V_k и T_k (распределение Бернулли) весьма универсальное по своей природе и может быть адекватно использовано для описания неисправностей и серьезных сбоев самолетов всех типов.

Введем матрицу оптимизационных переменных следующего вида: $u = \{u_{ik}\}$, $k = \overline{1,K}$, $i = \overline{1,L}$,

где элемент u_{ik} матрицы u принимает значения “1” если k-й самолет задействован на i-м рейсе, и “0” в противном случае.

С учетом введенных обозначений прибыль компании может быть представлена следующим образом:

$$R(u) = D(u) - C(u); \tag{41}$$

где $D(u)$ – доходная часть, $C(u)$ – расходная часть,

$$D(u) = \sum_{i=1}^{L}\sum_{k=1}^{K} u_{ik}\rho_i(\min\{z_k, X_i^1\} + \min\{z_k, X_i^2\} \tag{42}$$

$$C(u) = \sum_{k=1}^{K} c_k^0 + \sum_{i=1}^{L}\sum_{k=1}^{K} u_{ik}(c_{ik}^1 + (c_{ik}^2 + c_k^3)t_i^0) + \sum_{i=1}^{L}\sum_{k=1}^{K} u_{ik}(V_k c_i^4). \tag{43}$$

Поясним значение некоторых величин, входящих в модель расчета себестоимости. Величина c^3_k позволяет учесть стоимость ремонтных работ для каждого самолета. Случайная величина V_k введена для учета потерь, связанных с невозможностью самолета выполнить свой запланированный рейс из-за крупной поломки. Для выполнения этого полета придется арендовать самолет другой авиакомпании или перераспределить график использования самолета в своем парке.

Представленная модель для расчета расходных деталей позволяет частично учитывать вероятный сбой самолета и оценивать реальные затраты на обслуживание парка воздушных судов в течение всего базового периода. Разработка общей модели является частью расходов, является предметом отдельного исследования, результаты которого здесь не перечислены. По сути, это подмодель, связанная с функционированием флота компании и не учитывающая другие затраты, такие как ремонт зданий и сооружений компании.

В качестве критерия для оптимизации этой математической модели математическое ожидание прибыли компании (средняя прибыль).

В результате построения математической модели оптимизации сводится к решению задач стохастического программирования большой размерности с булевыми переменными. Особенность проблемы заключается в максимизации математического ожидания при наличии вероятностных ограничений и дискретных переменных оптимизации. Однако эта формулировка естественна и адекватна реальности. Итак, наконец, модель функционирования флота авиакомпании выглядит следующим образом:

$$u^* = \arg\max_u M[R(u)] \qquad (44)$$

при детерминированных ограничениях:

$$\sum_{k=1}^{K} u_{ik} = 1,\ u_{ik} \in \{0,1\},\ i = \overline{1,L}, \qquad (45)$$

$$u_{ik} = 0,\ i \in I_2,\ k \in K_2\ , \qquad (46)$$

и вероятностных ограничениях:

$$P\{\sum_{k=1}^{K} u_{ik} z_k \geq X_i^1\} \geq \alpha_i^1,\ \ i \in I_1, \qquad (47)$$

$$P\{\sum_{k=1}^{K} u_{ik} z_k \geq X_i^2\} \geq \alpha_i^1,\ i \in I_1, \qquad (48)$$

$$P\{G_i(u,Y) \leq 0\} \geq \alpha_i^1, \qquad (49)$$

Здесь P – вероятность соответствующего события; M – оператор математического ожидания;

α^1_i, α^2_i – заданные вероятности выполнения соответствующих ограничений, $i \in I_1$;

I_1 – множество особых (разрабатываемых, перспективных и т.д.) рейсов в неделю, по которым невзирая на расходную часть требуется перевезти максимальное число пассажиров с заданными вероятностями;

I_2 – множество рейсов, на которых не все самолеты из летного парка могут быть задействованы в силу ограниченности своих технических

характеристик или ограниченности возможностей аэропорта города назначения;

K_i – множество самолетов, которые не могут быть задействованы на i-м рейсе.

Рассмотрим подробнее поставленную задачу. Целевая функция представляет собой среднюю прибыль компании. Ограничения (47)–(48) отражают требование по ряду рейсов перевезти практически всех потенциальных пассажиров с заданными вероятностями, которые задаются экспертами. Заметим, что ввиду случайности пассажиропотока перевезти абсолютно всех пассажиров невозможно. Ограничения (45) означают, что каждый рейс должен обслуживать один и только один самолет. Ограничения (46) представляют собой запрет на использование определенных воздушных судов на определенных маршрутах, которые могут быть связаны с техническими характеристиками самого воздушного судна, или с невозможностями такого типа воздушных судов в аэропорту назначения. Вероятностное ограничение (49) отражает требование реализовать с заданной вероятностью всех технических ограничений, связанных со временем послевоенного обслуживания со случайной временной задержкой каждого воздушного судна. Это учитывает тот факт, что циклический характер расписания полетов, то есть необходимые временные соединения, расписание рейсов начинается в течение базового периода времени с окончанием полетного графика основного периода времени. Условие (49) физически означает, что график полета должен быть составлен таким образом, чтобы все летательные аппараты с вероятностью не были более низкими, чтобы вернуться в базовый аэропорт и пройти послеполетное техническое обслуживание до начала вылета следующего рейса. Таким образом, в этой модели значение tik включает в себя время полета на i-м рейсе с учетом временной службы и случайной задержки воздушных служб, связанных с воздушным судном. Таким образом, ограничение (49) не учитывает задержки, вызванные разбивкой воздушных судов, и задержки, связанные с поздним прибытием воздушных судов в

базовом аэропорту. Это связано с тем, что повреждение самолета происходит нечасто и попытка учесть их в ограничении (49) приводит к значительному увеличению количества t_{ik}. В результате таких попыток график полета самолета после каждого полета в течение долгого времени должен стоять в базовом аэропорту, для компенсации временного приёма, но могут быть задержки из-за поломки, и флот используется очень неэффективно. Возможная потеря авиационных сбоев воздушных судов, частично учтенная в расходной части (43), что приводит к интенсивному использованию более надежных с точки зрения возможного ущерба для воздушных судов.

По той же причине не учитываются и задержки, связанные с поздним прибытием воздушных судов в базовом аэропорту.

Эта математическая модель может использоваться не только для максимизации прибыли, но и для максимизации доходов и минимизации затрат, а также для оптимизации любой свертки последних двух критериев, взятых с соответствующими весами в зависимости от их тяжести.

2.2 Алгоритм решения поставленной задачи

Заметим, что, зная распределения случайных величин X_i^{1}, X_i^{2}, математические ожидания величин $\min\{z_k, X_i^{1}\}$, $\min\{z_k, X_i^{1}\}$ можно найти аналитически, а для ограничений (47)–(48) может быть выписан детерминированный эквивалент (50) в виде детерминированных линейных неравенств.

Согласно доверительному подходу (50) для нахождения гарантирующего решения задачи (44)–(49), т.е. доставляющего нижнюю оценку для максимального значения средней прибыли, достаточно решить следующую задачу:

$$u = \arg\max_{u} M[R(u)] \tag{50}$$

при ограничениях:

$$\sum_{k=1}^{K} u_{ik} z_k \geq x_{\alpha i}^1,\ i \in I_1, \tag{51}$$

$$\sum_{k=1}^{K} u_{ik} z_k \geq x_{\alpha i}^2,\ i \in I_1, \tag{52}$$

$$\sum_{k=1}^{K} u_{ik} = 1,\ u_{ik} \in \{0,1\},\ \ i = \overline{1,L}, \tag{53}$$

$$u_{ik} = 0,\ k \in K_i,\ i \in I_2, \tag{54}$$

$$\max_{y \in S_\alpha} G(u, y) \leq 0, \tag{55}$$

где $M_R(u) \hat{=} \sum_{i=1}^{L} \sum_{k=1}^{K} \tilde{C}_{ik} u_{ik} + C^0$ - математическое ожидание прибыли компании с найденными аналитически коэффициентами $\tilde{C}_{ik}$ u C^0, $k = \overline{1,K}$, $i = \overline{1,L}$, $x_{\alpha i}^1$, $x_{\alpha i}^1$ – квантили уровня a_i^1, a_i^2 распределения случайных величин X_i^1, X_i^2, $i \in I_1$, S_α - доверительное множество с вероятностной мерой α для случайного вектора Y, составленного из случайных величин Y_i, $i = \overline{1,L}$.

Выберем доверительное множество в виде прямоугольника

$$S_\alpha = \{y : 0 \leq y_i \leq y_{\alpha i}, i = \overline{1,L}\}, \tag{56}$$

где $y_{\alpha i}$ – квантиль уровня α_1 случайной величины Y_i.

Так как Y_i независимы (продолжительность задержки, связанной с плохой работой служб аэропорта и произошедшей на одном рейсе, не зависит от продолжительности задержки того же типа, случившейся на другом рейсе) и P{ $Y_1 \leq 0$}=0, то для того, чтобы доверительное множество имело вероятностную меру , положим $\alpha_1 = \sqrt[L]{\alpha}$ .i

Поскольку доверительное множество выбрано в виде прямоугольника, то ограничение (55) можно записать в виде

$$G_{ijk}(u_{ik}, u_{jk}, y_i) \leq 0 \text{ для всех } y_i \in [0; y_{i\alpha}],\ j = \overline{1,L}, k = \overline{1,K},\ i = \overline{1,L} \tag{57}$$

Рассмотрим зависимость G_{ijk} (u_{ik} , u_{jk} , y_i) от j и k для некоторого фиксированного $i=\overline{1,L}$. Пусть в начале для некоторых j и k выполняется условие $u_{ik} = u_{jk} = 1$. Тогда легко проверить, что максимум G_{ijk} (u_{ik} , u_{jk} , y_i) для данных j и k достигается при $y_i = y_{\alpha 1}$. Для всех оставшихся j и k выполняются условия $u_{ik} \neq 1$ или $u_{ik} \neq u_{ij}$. В этом случае легко проверить, что для таких j и k выполняется G_{ijk} (u_{ik} , u_{jk} , y_i) ≤ 0 при любых $y_i \in [0; y_{i\alpha}]$, в том числе и для $y_i = y_{i\alpha}$. В результате ограничения (57) можно переписать в следующем виде:

$$G_{ijk}(u_{ik}, u_{jk}, y_{i\alpha}) \le 0,\ j=\overline{1,L}, k=\overline{1,K},\ i=\overline{1,L} \qquad (58)$$

Таким образом, задача (50)–(54), (58) нахождения гарантирующего решения задачи (44)–(49) является задачей линейного программирования большой размерности (при большом значении K L) с булевыми переменными. Размерность задачи не позволяет напрямую (без модификаций) использовать “классические” методы линейного целочисленного программирования [11].

Однако, на практике, обычно, K × L велико (K > 10, L > 100). Для этих значений K и L не получилось разработать алгоритм, находивший бы оптимальную для задачи (50)–(54), (58) стратегию за доступное время. Однако получилось создать алгоритм, который находит за небольшой временной период (удовлетворяющую всем ограничениям) стратегию для задачи (50)–(53), (58), при которой значение y критериальной функции (50) рядом с оптимальным. При малом значении величины K × L (менее 400) алгоритм осуществляет поиск оптимальной стратегии для задачи (50)–(54), (58).

В разработанном субоптимальном алгоритме применяется идея метода границ и ветвей, состоящая в неявном переборе допустимых комбинаций y переменных оптимизации. Алгоритм заключается в решении последовательности вспомогательных детерминированных задач типа (50)–(54), (58), но гораздо меньшей размерности.

Опишем субоптимальный алгоритм подробнее, который позволяет получать хорошую допустимую стратегию для задачи (50)–(54), (58).

Идея алгоритма заключается в следующем. Необходимо сначала решить задачу составления летного оптимального расписания для первых l_{m} рейсов, последовательно расположенных. Самолет, используемый в первом рейсе, запоминается. Далее решается задача по составлению оптимального летного расписания для первых l_{m} + 1 рейсов, последовательно расположенных, с учетом применения на первом рейсе запомненного самолета. Самолет, используемый на 2-ом рейсе, запоминается. Затем происходит снова решение задачи составления оптимального летного расписания для первых l_{m} + 2 рейсов, последовательно расположенных с учетом применения на 1-ых 2-ух рейсах самолетов, которые были ранее запомнены. И т.д., пока все рейсы не будут заполнены.

1. На первом шаге алгоритма (m = 1) для матрицы $\tilde{u}$ u полагается $\tilde{u}_{\text{ik}}$ = 0, $i = \overline{1, L}$;

2. На каждом m-м шаге алгоритма задается глубина просмотра l_{m};

3. Формируется множество индексов для m-го шага:

$$J_m \hat{=} \{m, m+1, \ldots, \min\{L, l_m\}\} \cup \{1, 2, \ldots, \max\{0, m + l_m - L\}\}$$

Данное множество соответствует рейсам, для которых на m-м шаге формируется летное расписание;

4. Решается задача формирования оптимального летного расписания для l_{m} рейсов из множества J_{m} с учетом уже задействованных на первых m − 1 рейсах самолетов.

$$\sum_{i \in I_m} \sum_{k=1}^{K} \tilde{C}_{ik} u_{ik} \to \max_{\tilde{u} \in U_m} \tag{59}$$

где множество U_{m} определяется следующими ограничениями:

$$\sum_{k=1}^{K} u_{ik} z_k \geq x_{\alpha i}^1, \; i \in I_1 \cap J_m \tag{60}$$

$$\sum_{k=1}^{K} u_{ik} z_k \geq x_{\alpha i}^2, \; i \in I_1 \cap J_m \tag{61}$$

$$\sum_{k=1}^{K} u_{ik} = 1, \; u_{ik} \in \{0,1\}, \; i \in J_m \tag{62}$$

$u_{ik} = 0,\ i \in I_2 \cap J_m,\ k \in K_i$ (63)

$u_{ik} = 0$, если $i \notin I_m$ и $\tilde{u}_{ik} = 0$, (64)

$u_{ik} = 1$, если $\tilde{u}_{ik} = 1$

5. Если номер текущего шага m < L $- l_{\text{m}}$, то полагаем, что $\tilde{u}_{ik} := 1$ для $u_{ik} = 1$; иначе $\tilde{u}_{ik} := 1$ для $u_{ik} = 1$; где $i \in J_m$, переход к п. 7;

6. m := m + 1, переход к п. 2;

7. Конец работы алгоритма.

Матрица $\tilde{u}$ принимается в качестве субоптимального решения задачи. При этом критерий равен $\tilde{\text{R}} = \sum_{i=1}^{L} \sum_{k=1}^{K} \tilde{C}_{ik} \tilde{u}_{ik} + C^0$.

Время работы изложенного алгоритма и близость решения к оптимальному зависят от настроечных параметров l_{m}. При увеличении параметра l_{m} летное расписание, получаемое с помощью алгоритма, становится ближе (в смысле критерия) к оптимальному, однако при этом время работы алгоритма существенно увеличивается. Если взять l_{m} = L, то алгоритму удается найти точное решение задачи (50)–(53). Однако при больших L и K в этом случае время его работы становится через-чур велико, в следствии чего пропадает возможность использовать на практике очень большие значения l_{m}. Оптимальные (с точки зрения соотношения “время работы – точность решения") значения l_{m} для задачи свои и существенно зависят от количества самолетов K.

Причина, по которой произошло увеличение времени счета алгоритма, если увеличивается l_{m} – зависимость размерности задачи по линейному целочисленному программированию с булевыми переменными (59)–(65), которая решается в п.4 алгоритма, от величины l_{m} .

В основе решения задачи (59)–(65) находится идея (применительно к специфике у рассматриваемой задачи) метода границ и ветвей. Осуществляется неявный перебор всевозможных разрешенных комбинаций у оптимизационных переменных с осуществлением выбора наилучшей. Причем происходит явное

перебирание исключительно небольшой части комбинаций, что значительно уменьшает время счета, если сравнивать с явным полным перебором.

Для описания алгоритма решения задачи (59)–(65) введем обозначения:

$\tilde{u}^{ij}$ – матрица размерности L × K, у которой все элементы равны нулю, кроме j-го элемента i-й строки, равному единице;

J – некоторое множество индексов (рейсов);

$\tilde{Q}(J)$ – множество матриц размерности L × K, причем каждая матрица $\tilde{q}(J) \in \tilde{Q}(J)$ состоит из нулей и единиц, а элементы $\tilde{q}(J)$ удовлетворяют ограничениям:

$\sum_{k=1}^{K} \tilde{q}_{ij}(J) = 1$, если $i \in J_m$, остальные равны нулю.

Назовем $\tilde{u}^{ij}$ допустимой матрицей для некоторой матрицы $\tilde{q}(J) \in \tilde{Q}(J)$, если ни одно из ограничений (60), (61), (63)–(65) не нарушается при подстановке в них вместо матрицы u матрицы $\tilde{u}^{ij} + \tilde{q}(J)$;

$\tilde{U}q(J)$ - множество всех допустимых матриц для некоторой матрицы $\tilde{q}(J) \in \tilde{Q}(J)$;

$C_i^* \triangleq \max_{k \in F_i} \tilde{C}_{ik}$ - наибольший возможный вклад в критерий индекса (рейса) I для некоторой $\tilde{q}(J) \in \tilde{Q}(J)$, где $F_i = \{k : \tilde{u}^{ik} \in \tilde{U}q(J)\}$.

Описание алгоритма решение задачи (59)-(60) приведем для шага m=1:

1. Положим $S = -\infty$; i=1, r=0, J=∅;

2. Если $S' = \sum_{i \in I_m \setminus J} \tilde{C}_i^* + \sum_{i=1}^{L} \sum_{k=1}^{K} \tilde{C}_{ik} \tilde{q}_{ik}(J)$ - оценка сверху величины значения критерия задачи для $\tilde{q}(J)$;

3. Если S’≤ S, то перейти к п. 4; иначе перейти к п. 7;

4. J := J \{r}; $\tilde{q}(J) := \tilde{q}(J) - \tilde{u}^{ik}$;

5. Выбираем $\tilde{u}^{ik} \in \tilde{U}q(J)$ такое, что *j = min {l:l>r}*. Если выбрать не удается, то перейти к п.6; иначе r:=j; J:=J+{r}, $\tilde{q}(J) := \tilde{q}(J) + \tilde{u}^{ik}$, переход к п.2;

6. Если i > 1, то i:=i-1; переход к п.4; иначе переход к п.9;

7. Если i ≤ l_m, i:=i+1, r:=0, переход к п.5; иначе перейти к п.8;

8. Полагаем что S = S', $u^* = \tilde{q}(J)$, переход к п.4;

9. Конец алгоритма; u^*, S – решение задачи

Благодаря изложенному алгоритму можно найти точное решение задачи (59)–(65). Нужно добавить, что помимо величины l_m на время нахождения решения оказывает существенное влияние число ограничений (60), (61), (63). Чем из большего числа элементов состоит множество I_1 и I_2, тем быстрее функционирует алгоритм.

В результате уменьшения время счета можно позволить более точную (чем в п. 2 алгоритма) оценка больше величины S. Точность данной оценки уменьшается из-за увеличения величины `$_m$. Увеличить точность можно, к примеру, за счет использования симплекс-метода для того, чтобы получить оценки S в отказе от оптимизационных целочисленности переменных. Благодаря более точной оценки S можно увеличить величину l_m сохранив время счета. Из-за увеличения l_m произойдет нахождение путей решения задачи (50)–(53), которое неподалеку от оптимального.

После получения расписания полетов необходимо уточнить стоимость прибыли компании, которая была найдена в расписании рейсов. Необходимость разъяснения возникает из-за того, что такие случайные факторы, как значительный ущерб авиации и продолжительность их устранения, не учитывались непосредственно при подготовке расписаний полетов (в противном случае это было слишком перестрахование) и учитывались косвенно, Добавив некоторые штрафные коэффициенты в расходную часть. Уточнение прибыли можно получить путем статистического моделирования функционирования флота компании, когда вы уже нашли расписание полетов. В симуляции после каждого полета моделируется случайной величиной, характеризующей основной ущерб самолета, а также случайное время его разрешения. Если самолет с отказом не может совершить ваш следующий рейс, график пересчитывается или арендуется самолетом другой компании для выполнения рейсов. При проведении статистического моделирования

функционирования флота компании в дополнение к значениям средней прибыли, получаемой компанией, также могут быть получены дополнительные характеристики деятельности компании.

ГЛАВА 3 РАЗРАБОТКА ПРОГРАММЫ

3.1 Выбор программного обеспечения

Под программным обеспечением понимается совокупность программ на носителях данных и программных документов, предназначенная для отладки, функционирования и проверки работоспособности автоматизированной системы.

На начальных этапах процесса проектирования ПО должны быть приняты принципиальные решения, во многом определяющие этот процесс, а также качество и трудоемкость разработки. К таким решениям относят:

→ выбор подхода к разработке;

→ выбор архитектуры программного обеспечения;

→ выбор систем управления базами данных;

→ выбор языка и среды программирования.

Следующим шагом необходимо язык программирования. Существует множество языков программирования, различающихся по предоставляемым возможностям. Выбор языка программирования для решения конкретной задачи существенно влияет на качество программы, получаемой в результате разработки, а также на скорость и удобство самого процесса разработки. С точки зрения возможностей, предоставляемых для непосредственной реализации алгоритма, а также свойств исполняемых программ, получаемых в процессе разработки, можно определить несколько критериев выбора языка программирования:

- невысокая цена;
- портативность разработанного продукта под различные системы;
- простота настройки;
- скорость выполнения команд;
- нацеленный на работу в Интернет;
- универсальный и ясный синтаксис.

Всем необходимым требованиям соответствует несколько языков:

– PHP;

– Perl;

– Java;

– ColdFusion.

Для обоснования выбора наиболее подходящего языка необходимо рассмотреть каждый из них.

PHP является языком еще довольно молодым, он (точнее, его интерпретатор) установлен уже на порядка миллиона серверов по всему миру, и цифра продолжает расти. Новое поколение PHP должно вообще стереть все преимущества Perl перед PHP, как с точки зрения быстродействия обработки программ, так и с точки зрения синтаксиса. Наконец, большинство PHP-сценариев работают быстрее аналогичных им программ, написанных на Perl. PHP является самым молодым, перспективным и быстроразвивающимся из языков программирования для Internet, доля его использования по сравнению с другими языками быстро растет [16].

Perl является интерпретатором и появляется гораздо раньше, чем в Интернете. Широко распространенный Perl получил потому, что он был доступен на каждом веб-сервере, так как почти все они работали под UNIX, а альтернатива была более сложным языком. С. Perl теряет изящество PHP, хотя PHP предлагает аналогичную функциональность Perl в значительной степени более низкую избыточность и относительную простоту синтаксиса. Избыточность Perl заключается в том, что он предназначен для различных приложений, которые не могут повлиять на синтаксис.

Различать язык Java и технологию Java. Язык Java - это C-подобный язык, который был разработан как «улучшенный C ++». Технология Java включает в себя клиентские и серверные части, а также доступ к базам данных, поэтому наиболее корректно сравнивать Java с Apache / PHP / MySQL. Технология Java была разработана как кросс-платформенная технология, позволяющая создавать веб-приложения масштаба предприятия. Основными преимуществами этой технологии являются переносимость и кросс-

платформенный объектно-ориентированный язык, который позволяет создавать сложные и крупные приложения. Среди недостатков - медленное выполнение, потребление больших объемов памяти (плата за кросс-платформенную переносимость) и сложность разработки веб-приложений. Использование Java для разработки довольно простого приложения, вряд ли оправдано, и при работе над серьезными задачами Java стоит дороже.

Пакет ColdFusion, разработанный Allaire, предназначен для быстрой разработки и интерактивных динамических веб-документов, обрабатывая информацию, полученную из базы данных. Недостатком ColdFusion является низкая допустимость. ColdFusion работает только на четырех платформах: Win32, Solaris, HP / UX и Linux. Кроме того, ColdFusion является коммерческой разработкой. ColdFusion довольно необычен. Преимущество ColdFusion - хорошая среда разработки.

Для более объективного выбора необходимо провести сравнительный анализ рассмотренных языков по необходимым критериям. Результаты проведенного анализа можно увидеть в таблице 2.1 причем наличие того или иного критерия будем обозначать числом-баллом в интервале от 0 (не поддерживается) до 10 (полная поддержка).

Таблица 2.1- Анализ языков программирования

Критерии	PHP	Perl	Java	ColdFusion
Быстродействие	10	5	10	5
Простота синтаксиса	5	5	8	5
Межплатформенная переносимость	10	10	10	5
Доступность (цена)	10	10	10	3
Итого:	35	30	38	18

Из приведенного выше сравнения языков, можно сделать вывод, что Java обладает всеми необходимыми требованиями к языку программирования разрабатываемой системы, по сравнению с другими языками программирования.

3.2 Обзор и обоснование выбора среды разработки Eclipse

В настоящее время существует большое множество систем IDE, универсальных и ориентированных на определённый язык разработки, платных и бесплатных, с закрытыми исходным кодом и с открытыми исходным кодом. Кроме того, писать программы можно и в текстовом редакторе, а компилировать их с помощью командной строки и компилятора, поэтому приемлема разработка очень простых проектов и в блокноте.

Наиболее популярные текстовые редакторы – Notepad++, SublimeText, Bred2, VIm, Emacs – с поддержкой «подсветки» синтаксиса и другими особенностями, характерными для IDE. Среди полноценных интегрированных сред разработки, используемых при написании кода на Java, используют программные пакеты NetBeans, Eclipse, и IntelliJ IDEA. Все пакеты мощные и достаточно давно используются в профессиональной разработке ПО. Причиной для этого послужили такие критерии как:

- бесплатное распространение (для IntelliJ IDEA Community edition);

- поддержка J2SE, J2EE, J2ME;
- высокая популярность среди разработчиков;
- обеспечение средствами профилирования и отладки.

NetBeans и Eclipse являются классическими системами, а IntelliJ новая, но, имеющая высокий прирост популярности, IDE.

Пользовательский интерфейс Eclipse отличается удобным интерфейсом, лояльной системой подсказок, «линзированием» (быстрый показ фрагмента кода в специальной «линзе», при наведении мыши на пометки справа), встроенными возможностями работы с системами контроля версий git, cvs, github, mercurial, subversion. Удобная настройка и взаимодействие с ftp, sftp, ssh и удобный отладчик. Хороший набор предустановленных шаблонов проектов и поддержка плагинов, позволили использовать Eclipse в данной работе для разработки ПО.

3.3 Разработка графического интерфейса и модулей программы

В разработанной программе используется Model-View-Controller это схема разделения интерфейса и управляющей логики на три отдельных компонента: модель, представление и контроллер.

Схема проектируемой программы представлена на рисунке 3.1

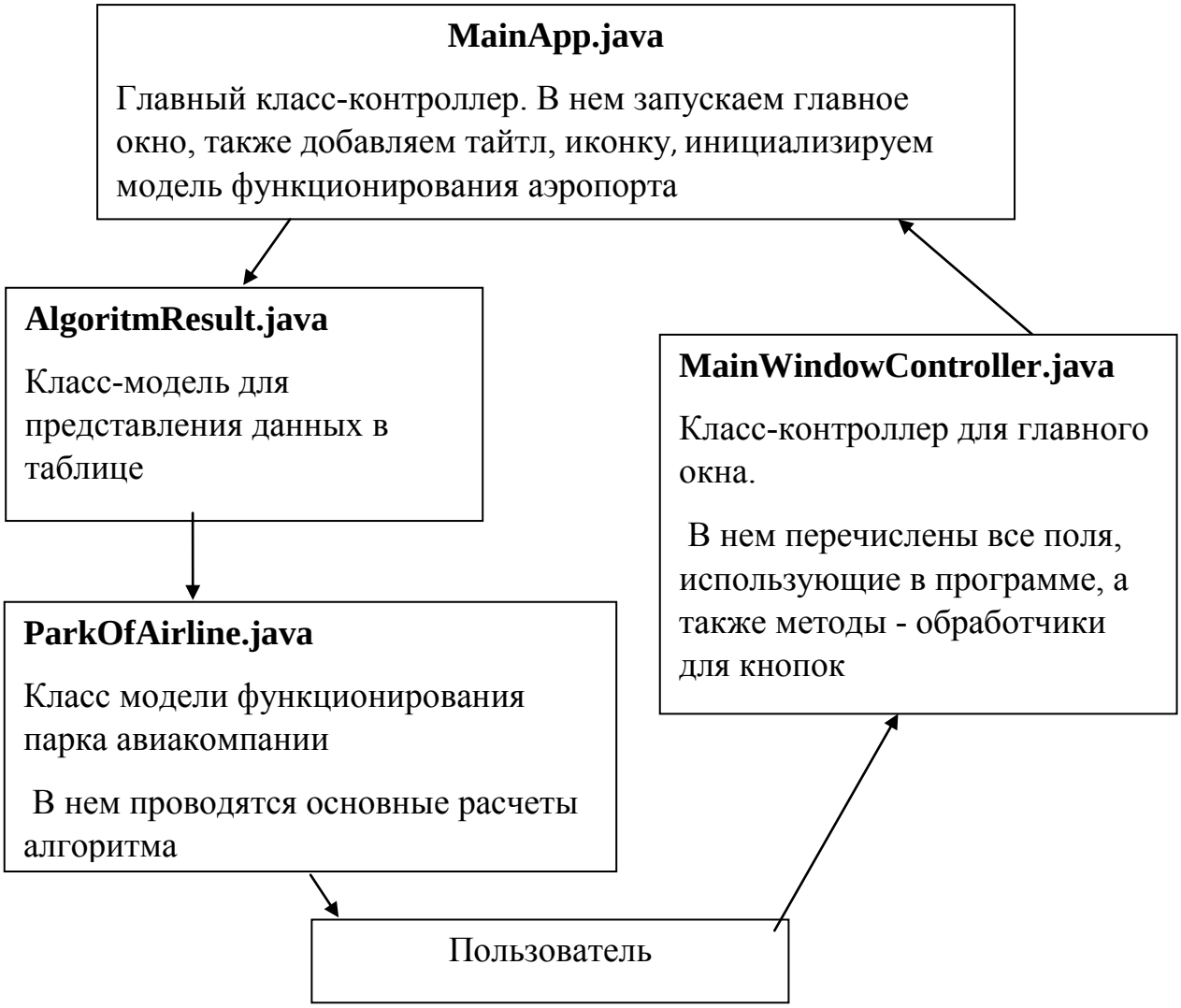

Рисунок 3.1 – Схема проектируемой программы

Основной алгоритм программы реализован в модуле ParkOfAirline.java:

```
package ru.mathnet.airplanecompany.model;
import java.util.ArrayList;
import java.util.List;
/**
 * Класс модели функционирования парка авиакомпании
 * В нем проводятся основные расчеты алгоритма
 *
 */
public class ParkOfAirline {
```

```
//Входные параметры
private int numberOfAirplanes;
private int numberOfFlights;
private float avarageNumberOfSeats;
private float avarageTicketPrice;
private float avarageCostOfFlight;
//Дополнительные параметры для упрощения расчетов
private double[][] ticketPrice;
private double[][] costOfFlights;
private int[] basePassangerTraffic;
private int[] destPassangerTraffic;
private int[] numberOfSeats;
//Конструктор класса
public ParkOfAirline(int numberOfAirplanes, int numberOfFlights, float avarageNumberOfSeats,
                float avarageTicketPrice, float avarageCostOfFlight) {
        this.numberOfAirplanes = numberOfAirplanes;
        this.numberOfFlights = numberOfFlights;
        this.avarageNumberOfSeats = avarageNumberOfSeats;
        this.avarageTicketPrice = avarageTicketPrice;
        this.avarageCostOfFlight = avarageCostOfFlight;
                init();
}
```

Код остальных модулей программы представлен в Приложении А

Программа написана в среде разработке Eclipse.

Основной интерфейс программы представлен на рисунке 3.2.

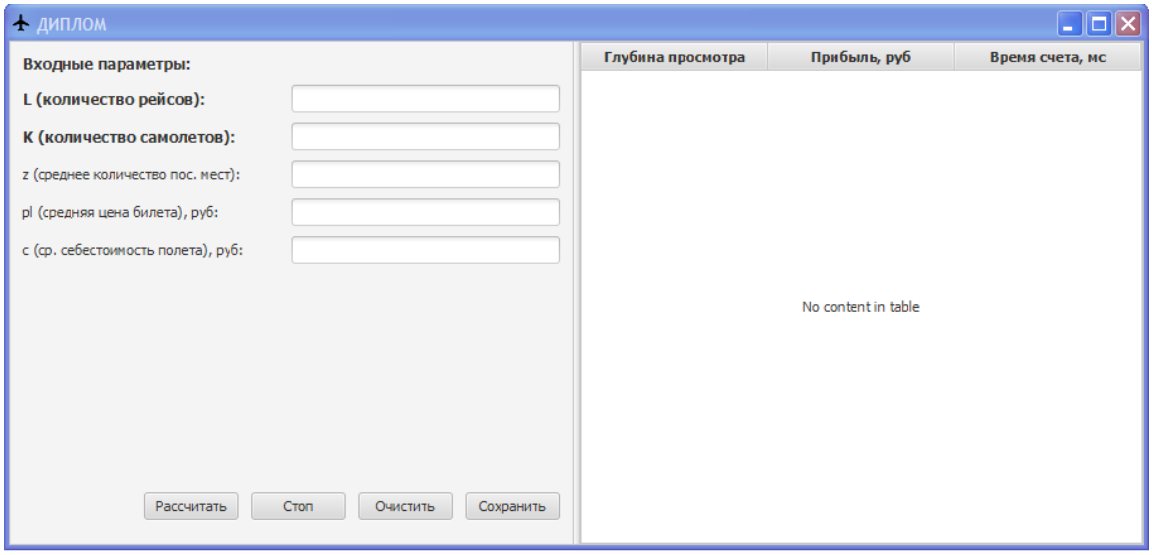

Рисунок 3.2 – Основной интерфейс программы

После ввода начальных данных получаем расчет прибыли (рис. 3.3).

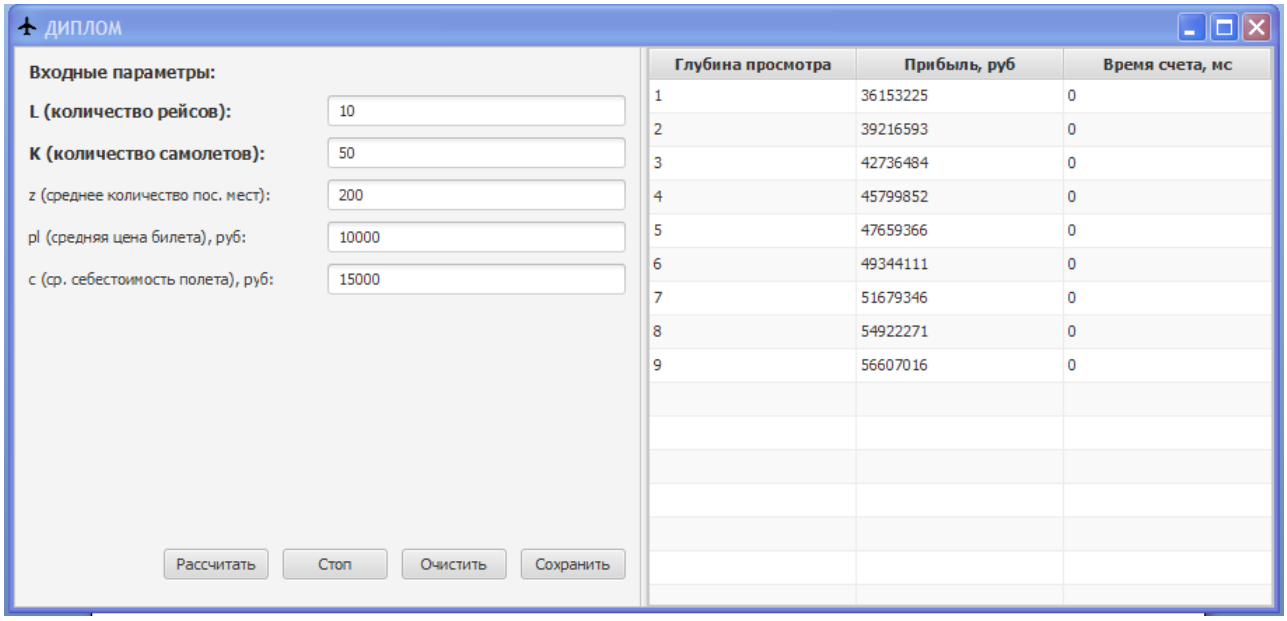

Рисунок 3.3 - Расчет прибыли авиакомпании

Данные показатели можно использовать практически в любой авиакомпании. Также был усовершенствован алгоритм, что позволило гораздо быстрее обрабатывать расчетную часть.

ЗАКЛЮЧЕНИЕ

Существует большое количество работ, в которых при решении различного рода экономических задач используются методы стохастического программирования. Исторически сложилось два направления в стохастическом программировании при моделировании экономических систем. К первому относятся модели, в которых оптимизируется какой-либо средний показатель эффективности работы экономической системы, например, средняя прибыль или средние издержки компании. Как правило, это двухэтапные задачи с критерием в форме математического ожидания. Второе направление содержит модели, позволяющие получать гарантированный с заданной вероятностью результат. К ним относятся задачи с вероятностными ограничениями. Достаточно редко встречаются модели, где критерий в виде математического ожидания и вероятностные ограничения используется совместно.

В выпускной квалификационной работе с помощью языка программирования Java была реализована задача по максимизации средней прибыли авиационной компании при существовании некоторых вероятностных ограничений при фиксированном временном расписании полетов.

Рассмотренная модель не претендует на оптимизацию всего процесса функционирования авиационной компании.

Разработанный алгоритм может быть использован для максимизации средней прибыли авиакомпании.

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

Научная и методическая литература

1. Азанов В.М. Оптимизация коррекции околокруговой орбиты искусственного спутника Земли по вероятностному критерию / Кан Ю.С.: Труды ИСА РАН. 2015. Т. 65. №2. С. 18–26.
2. Акулич И. Л. Математическое программирование в примерах и задачах: Учеб. пособие для студентов эконом. спец. вузов. - М.: Высш. шк., 1986. - 319 с.
3. Аттетков А.В. Методы оптимизации / Галкин С.В., Зарубин В.С.: Учебник для вузов. - М.: Изд-во МГТУ им. Баумана, 2001. - 440 с.
4. Базара М.Д. Нелинейное программирование. Теория и алгоритмы / Шетти К.: – М.: Мир, 1982.-583с.
5. Богуславский И.А. Методы навигации и управления по неполной статистической информации - М.: Машиностроение, 2014.
6. Вагнер Г.С. Основы исследования операций - М.: Мир, 1972-1973. - 336 с. + 488 с. + 503 с..
7. Вишняков Б.В. Оптимизация двухшаговой модели изменения капитала по различным статистическим критериям / Кибзун А.И. - АиТ. 2005. №7. С. 126–143.
8. Кан Ю.С. Задачи стохастического программирования с вероятностными критериями / Кибзун А.И. - М.: Физматлит, 2013.
9. Кан Ю.С. Минимизация квантили нормального распределения билинейной функции потерь / Тузов Н.В. - АиТ. 1998. № 11. С. 82–92.
10. Карманов В. Г. Математическое программирование: Учеб. пособие. — 5-е изд., стереотип. — М.: ФИЗМАТЛИТ, 2004. — 264 с.
11. Кибзун А.И. Сведение двухшаговой задачи стохастического оптимального управления с билинейной функцией дохода к задаче смешанного целочисленного линейного программирования / Игнатов А.Н. - АиТ. 2016. №12. С. 80–101.

12. Кибзун А.И. Оптимальное управление по квантильному критерию портфелем ценных бумаг / Кузнецов Е.А. - АиТ. 2011. № 9. С. 101–113.

13. Котов, В.П. Математическое программирование: Учебное пособие / В.П. Котов, Н.А., Адрицкая и др. - СПб.: Лань, 2014. - 432 с.

14. Мину М.Н. Математическое программирование. Теория и алгоритмы: Пер с франц. - М.: Наука, 1990. - 487 с.

15. Поляк Б.Т. Введение в оптимизацию. М.: Наука, 2012.

16. Соколов А.В. Методы оптимальных решений. В 2 т.Т. 1.Общие положения. Математическое программирование / А.В. Соколов, В.В. Токарев. - М.: Физматлит, 2012. - 564 с.

17. Таха Х. А. Введение в исследование операций, 7-е издание / Хемди А.: Пер. с англ. — М.: Издательский дом "Вильямс", 2005. — 912 с: ил.

18. Хедли Дж. Нелинейное и динамическое программирование: Пер. с англ. - М.: Мир, 1967.-506с

19. Юдин Д.Б., Линейное программирование. Теория, методы, приложения / Гольштейн Е.Г. - М.: Наука, 1969. - 424 с

20. Юрьева, А.А. Математическое программирование: Учебное пособие / А.А. Юрьева. - СПб.: Лань, 2014. - 432 с.

Литература на иностранном языке

21. Artzner P.T. Coherent measures of risk. Mathematical Finance. 2014. V. 9. No. 3. P. 203228.

22. Barmish B.R., The uniform distribution: a rigorous justification for its use in robustness analysis / Lagoa C.M. - Math. Control, Signals Systems. 1997. V. 10. P. 203222.

23. Beale E.M.L. On Minimizing a Convex Function Subject to Linear Inequalities / Journal of Royal Statistical Society. 1955. V. 17. Series B. P. 173–184.

24. Benati S. A. Mixed integer linear programming formulation of the optimal of the optimal mean/ValueatRisk portfolio problem / Rizzi R. - European Journal of Operational Research. 2007. V. 176. No 1. P. 423–434.

25. Benson H.Y. Interiorpoint methods for nonconvex nonlinear programming: Filter methods and merit functions./ Shanno D.F., Vanderbei R.J.: - Computational Optimization and Applications, 2002, V. 23, No. 2, P. 257272.

26. Jorion P.F. Value at Risk: The New Benchmark For Managing Financial Risk. Irwin Professional Publishing, 1997.

27. Kelly J.L. A new interpretation of information rate / Bell System Technical Journal, 1956. No. 35. P. 917–926.

28. Kovacevic R.M. Electricity swing option pricing by stochastic bilevel optimization: A survey and new approaches / Pflug G.Ch. European Journal of Operational Research, 2014. V. 237. No. 2. P. 389-403.

ПРИЛОЖЕНИЕ А

Листинг кода программы

MainApp.java

```
package ru.mathnet.airplanecompany;
import java.io.IOException;
import javafx.application.Application;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.fxml.FXMLLoader;
import javafx.scene.Scene;
import javafx.scene.image.Image;
import javafx.scene.layout.AnchorPane;
import javafx.stage.Stage;
import ru.mathnet.airplanecompany.model.AlgoritmResult;
import ru.mathnet.airplanecompany.model.ParkOfAirline;
import ru.mathnet.airplanecompany.view.MainWindowController;
/**
 * Главный класс-контроллер.
 * В нем запускаем главное окно, также добавляем тайтл, иконку, инициализируем модель функционирования аэропорта.
 *
 */
public class MainApp extends Application {

    //Главная "сцена" приложения
    private Stage primaryStage;
    //Список для представления в виде таблице
    private ObservableList<AlgoritmResult> resultData = FXCollections.observableArrayList();
```

```
//Модель аэропорта
private ParkOfAirline parkOfAirline;
    private MainWindowController controller;
//Метод гет для получения данных что используются в таблице. Используется в классе-контроллере
public ObservableList<AlgoritmResult> getResultData() {
    return resultData;
}
//Метод для вычисления результатов
public void startCalculation(int numberOfAirplanes, int numberOfFlights, int avarageNumberOfSeats,
            float avarageTicketPrice, float avarageCostOfFlight, int depthOfView) {
    if (parkOfAirline == null) {
        parkOfAirline = new ParkOfAirline(numberOfAirplanes, numberOfFlights, avarageNumberOfSeats,
                avarageTicketPrice, avarageCostOfFlight);
    }
    AlgoritmResult result = parkOfAirline.calculateIncomeForDepth(depthOfView);

    resultData.add(result);
    controller.refreshTable();
}
    //Метод гет для получения "сцены" приложения. Используется в классе-контроллере.
public Stage getPrimaryStage() {
    return primaryStage;
}
```

```
        //Переопределенный метод для запуска программы
        @Override
        public void start(Stage primaryStage) {
                this.primaryStage = primaryStage;
                this.primaryStage.setTitle("ДИПЛОМ");
                this.primaryStage.getIcons().add(new
Image("file:resources/images/airport_48px.png"));

                initRootPane();
        }
                //Инициализируем главное окно
        private void initRootPane() {
                try {
                        FXMLLoader loader = new FXMLLoader();

        loader.setLocation(MainApp.class.getResource("view/mainWindow.fxml"));
                        AnchorPane rootPane = (AnchorPane) loader.load();

                        Scene scene = new Scene(rootPane);
                        primaryStage.setScene(scene);

                        controller = loader.getController();
                        controller.setMainApp(this);

                        primaryStage.show();
                }
                catch (IOException e) {
                        e.printStackTrace();
                }
```

```
    }
    public static void main(String[] args) {
        launch(args);
    }
}
```

AlgoritmResult.java

```
package ru.mathnet.airplanecompany.model;

import javafx.beans.property.IntegerProperty;
import javafx.beans.property.LongProperty;
import javafx.beans.property.SimpleIntegerProperty;
import javafx.beans.property.SimpleLongProperty;

/**
 * Класс-модель для представления данных в таблице
 *
 */
public class AlgoritmResult {

    private final IntegerProperty depth;
    private final LongProperty income;
    private final IntegerProperty time;

    public AlgoritmResult(int depth, long income, int time) {
        this.depth = new SimpleIntegerProperty(depth);
        this.income = new SimpleLongProperty(income);
        this.time = new SimpleIntegerProperty(time);
    }
```

```
    public int getDepth() {
        return depth.get();
    }
    public void setDepth(int depth) {
        this.depth.set(depth);
    }
    public IntegerProperty depthProperty() {
        return depth;
    }
    public long getIncome() {
        return income.get();
    }
    public void setIncome(long income) {
        this.income.set(income);
    }
    public LongProperty incomeProperty() {
        return income;
    }
    public int getTime() {
        return time.get();
    }
    public void setTime(int time) {
        this.time.set(time);
    }
    public IntegerProperty timeProperty() {
        return time;
    }
}
```

ParkOfAirline.java

```
package ru.mathnet.airplanecompany.model;

import java.util.ArrayList;
import java.util.List;
/**
 * Класс модели функционирования парка авиакомпании
 * В нем проводятся основные расчеты алгоритма
 *
 */
public class ParkOfAirline {

	//Входные параметры
	private int numberOfAirplanes;
	private int numberOfFlights;
	private float avarageNumberOfSeats;
	private float avarageTicketPrice;
	private float avarageCostOfFlight;
	//Дополнительные параметры для упрощения расчетов
	private double[][] ticketPrice;
	private double[][] costOfFlights;
	private int[] basePassangerTraffic;
	private int[] destPassangerTraffic;
	private int[] numberOfSeats;

	//Конструктор класса
	public ParkOfAirline(int numberOfAirplanes, int numberOfFlights, float avarageNumberOfSeats,
```

```
                float avarageTicketPrice, float avarageCostOfFlight) {
        this.numberOfAirplanes = numberOfAirplanes;
        this.numberOfFlights = numberOfFlights;
        this.avarageNumberOfSeats = avarageNumberOfSeats;
        this.avarageTicketPrice = avarageTicketPrice;
        this.avarageCostOfFlight = avarageCostOfFlight;

        init();
    }
    //Методы гет для некоторых полей
    public float getAvarageNumberOfSeats() {
        return avarageNumberOfSeats;
    }
    public float getAvarageTicketPrice() {
        return avarageTicketPrice;
    }
    public float getAvarageCostOfFlight() {
        return avarageCostOfFlight;
    }
    //Инициализируем дополнительные переменные случайным образом
    private void init() {
        ticketPrice = new double[numberOfAirplanes][numberOfFlights];
        costOfFlights = new double[numberOfAirplanes][numberOfFlights];
        for (int i = 0; i < numberOfAirplanes; i++) {
            for (int j = 0; j < numberOfFlights; j++) {
                ticketPrice[i][j] = avarageTicketPrice * (1.5 -
Math.random());
                costOfFlights[i][j] = avarageCostOfFlight * (1.5 -
Math.random());
```

```
                }
            }
            basePassangerTraffic = new int[numberOfFlights];
            destPassangerTraffic = new int[numberOfFlights];
            for (int i = 0; i < numberOfFlights; i++) {
                basePassangerTraffic[i] = (int) (avarageNumberOfSeats * (1.5 - Math.random()));
                destPassangerTraffic[i] = (int) (avarageNumberOfSeats * (1.5 - Math.random()));
            }
            numberOfSeats = new int[numberOfAirplanes];
            for (int i = 0; i < numberOfAirplanes; i++) {
                numberOfSeats[i] = (int) (avarageNumberOfSeats * (1.5 - Math.random()));
            }
        }
        //Метод для рассчета дохода по формулах [1][2][3]
        //где sheduleMatrix - матрица расписаний
        private long calculateIncome(boolean[][] sheduleMatrix) {
            double income = 0;
            for (int i = 0; i < numberOfAirplanes; i++) {
                for (int j = 0; j < numberOfFlights; j++) {
                    if (sheduleMatrix[i][j]) {
                        income += ticketPrice[i][j] * (Math.min(numberOfSeats[i], basePassangerTraffic[j]) +
                                Math.min(numberOfSeats[i], destPassangerTraffic[j])) - costOfFlights[i][j];
                    }
                }
            }
```

```
            return (long) income;
        }
        //Метод для расчета дохода от определенного полета
        private long calculateIncomeFromFlight(int airplaneIndex, int flightIndex) {
            return (long) (ticketPrice[airplaneIndex][flightIndex] *
                    (Math.min(numberOfSeats[airplaneIndex],
basePassangerTraffic[flightIndex]) +
                    Math.min(numberOfSeats[airplaneIndex],
basePassangerTraffic[flightIndex])) -
                    costOfFlights[airplaneIndex][flightIndex]);
        }
        //В данном методе обьеденяются два алгоритма по формулам [10]-[14],
[18], [19]-[25] для одной заданной глубины
        public AlgoritmResult calculateIncomeForDepth(int lm) {
            long time = System.currentTimeMillis();
            List<SheduleMatrix> sheduleList = new ArrayList<SheduleMatrix>();
            //Перебор всех рейсов
            for (int i = 0; i < numberOfFlights; i++) {
                SheduleMatrix sheduleMatrix = new
SheduleMatrix(numberOfAirplanes, numberOfFlights);
                //Переменная для запоминания самолета
                boolean[] availableAirplane = new boolean[numberOfAirplanes];
                for (int z = 0; z < numberOfAirplanes; z++) {
                    availableAirplane[z] = true;
                }
                int[] betterFlights = new int[lm];
                //Перебор m лучших полетов
                for (int m = 0; m < lm; m++) {
                    //Выбор найлучшего самолета для полета
                    long maxS = 0;
```

```
                int z = 0; //индекс выбраного самолета
                for (int k = 0; k < numberOfAirplanes; k++) {
                    if (m + i < numberOfFlights && availableAirplane[k]) {
                        long s = calculateIncomeFromFlight(k, m + i);
                        if (s > maxS) {
                            maxS = s;
                            z = k;
                        }
                    }
                }
                betterFlights[m] = z;

                availableAirplane[z] = false;
                //сброс отмеченных самолетов если их больше не осталось
                int summ = 0;
                for (int j = 0; j < numberOfAirplanes; j++) {
                    summ += availableAirplane[j] ? 1 : 0;
                }
                if (summ == 0) {
                    for (int j = 0; j < numberOfAirplanes; j++) {
                        availableAirplane[j] = true;
                    }
                }
            }

            //меняем соответсвующее значение в матрице
            for (int j = 0; j < lm; j++) {
```

```
                    for (int k = 0; k < numberOfAirplanes; k++) {
                        if (i + j < numberOfFlights) {
                            sheduleMatrix.getData()[k][i + j] = betterFlights[j] == k;
                        }
                    }
                }

                //сохраняем матрицу расписаний
                sheduleList.add(sheduleMatrix);
            }
            //находим из списка найболее прибыльное расписание
            int size = sheduleList.size();
            SheduleMatrix bestSheduleMatrix = new SheduleMatrix(numberOfAirplanes, numberOfFlights);
            long bestIncome = calculateIncome(bestSheduleMatrix.getData());
            for (int i = 0; i < size; i++) {
                SheduleMatrix sheduleMatrix = sheduleList.get(i);
                long income = calculateIncome(sheduleMatrix.getData());
                if (income > bestIncome) {
                    bestSheduleMatrix = sheduleMatrix;
                    bestIncome = income;
                }
            }
            time = System.currentTimeMillis() - time;
            /*
            System.out.println("Shedule matrix");
            for (int i = 0; i < numberOfAirplanes; i++) {
                for (int j = 0; j < numberOfFlights; j++) {
```

```
                    System.out.print(bestSheduleMatrix.getData()[i][j] ? "1" :
"0");
                }
                System.out.println();
            }
            */
            return new AlgoritmResult(lm, bestIncome, (int) time);
        }
        //Класс для упрощения создания и инициализации матрицы расписаний
        private class SheduleMatrix {

            private boolean[][] data;

            public SheduleMatrix(int numberOfAirplanes, int numberOfFlights) {
                data = new boolean[numberOfAirplanes][numberOfFlights];
                int airplaneIndex = 0;
                for (int i = 0; i < numberOfFlights; i++) {
                    for (int j = 0; j < numberOfAirplanes; j++) {
                        data[j][i] = airplaneIndex == j;
                    }
                    if (++airplaneIndex >= numberOfAirplanes) {
                        airplaneIndex = 0;
                    }
                }
            }

            public boolean[][] getData() {
                return data;
            }
```

```
        }
}
```

MainWindowController.java

```
package ru.mathnet.airplanecompany.view;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import javafx.collections.ObservableList;
import javafx.fxml.FXML;
import javafx.scene.control.TableColumn;
import javafx.scene.control.TableView;
import javafx.scene.control.TextField;
import javafx.stage.FileChooser;
import javafx.scene.control.Alert;
import javafx.scene.control.Alert.AlertType;
import ru.mathnet.airplanecompany.MainApp;
import ru.mathnet.airplanecompany.model.AlgoritmResult;
/**
 * Класс-контроллер для главного окна.
 * В нем перечислены все поля, использующие в программе, а также методы - обработчики для кнопок
 *
 */
public class MainWindowController {

        //Поля которые используются в приложении
        @FXML
```

```
private TextField numberOfAirplanesField;
@FXML
private TextField numberOfFlightsField;
@FXML
private TextField baseTimePeriodField;
@FXML
private TextField avarageFlightsTimeField;
@FXML
private TextField avarageNumberOfSeatsField;
@FXML
private TextField avarageTicketPriceField;
@FXML
private TextField avarageCostOfFlightField;
//@FXML
//private TextField depthOfViewField;

//Таблица вывода результатов
@FXML
private TableView<AlgoritmResult> resultTable;
@FXML
private TableColumn<AlgoritmResult, Integer> depthColumn;
@FXML
private TableColumn<AlgoritmResult, Long> incomeColumn;
@FXML
private TableColumn<AlgoritmResult, Integer> countingTimeColumn;

//Ссылка на главный класс-контроллер
private MainApp mainApp;
private Thread calcThread;
```

```
    //Для задания ссылки на главный класс-контроллер
    public void setMainApp(MainApp mainApp) {
        this.mainApp = mainApp;
    }
    public void refreshTable() {
        resultTable.setItems(mainApp.getResultData());
    }
    //Специальный метод инициализации
    @FXML
    private void initialize() {
        depthColumn.setCellValueFactory(cellData -> cellData.getValue().depthProperty().asObject());
        incomeColumn.setCellValueFactory(cellData -> cellData.getValue().incomeProperty().asObject());
        countingTimeColumn.setCellValueFactory(cellData -> cellData.getValue().timeProperty().asObject());
    }
    //Обработчик кнопки
    @FXML
    private void handleCalculate() {
        if (isInputValid()) {
            int numberOfAirplanes = Integer.parseInt(numberOfAirplanesField.getText());
            int numberOfFlights = Integer.parseInt(numberOfFlightsField.getText());
            int avarageNumberOfSeats = Integer.parseInt(avarageNumberOfSeatsField.getText());
            float avarageTicketPrice = Float.parseFloat(avarageTicketPriceField.getText());
            float avarageCostOfFlight = Float.parseFloat(avarageCostOfFlightField.getText());
```

```
                //int depthOfView = Integer.parseInt(depthOfViewField.getText());

                mainApp.getResultData().clear();

                calcThread = new Thread() {

                    @Override
                    public void run() {
                        for (int i = 1; i < numberOfFlights; i++) {
                            try {

    mainApp.startCalculation(numberOfAirplanes, numberOfFlights,
                                    avarageNumberOfSeats, avarageTicketPrice, avarageCostOfFlight, i);
                            }
                            catch (OutOfMemoryError e) {
                                Alert alert = new Alert(AlertType.ERROR);

    alert.initOwner(mainApp.getPrimaryStage());
                                alert.setTitle("Критическая ошибка");
                                alert.setHeaderText("Не хватает оперативной памяти");
                                alert.setContentText("Не хватает оперативной памяти для необходимых расчетов");

                                alert.showAndWait();

                                return;
                            }
```

```
                                refreshTable();
                            }
                        }
                    };
                    calcThread.start();
                }
            }

            @FXML
            private void handleStop() {
                if (calcThread != null && calcThread.isAlive()) {
                    //Хоть и запрещенный метод, зато действенный
                    calcThread.stop();
                }
            }
            @FXML
            private void handleClear() {
                ObservableList<AlgoritmResult> data = mainApp.getResultData();
                data.clear();
                resultTable.setItems(data);
            }
            @FXML
            private void handleSave() {
                FileChooser fileChooser = new FileChooser();
                FileChooser.ExtensionFilter extFilter = new
FileChooser.ExtensionFilter("txt файлы", "*.txt");
                fileChooser.getExtensionFilters().add(extFilter);
```

```
        File file = fileChooser.showSaveDialog(mainApp.getPrimaryStage());
        if (file != null && !file.getPath().endsWith(".txt")) {
                file = new File(file.getPath() + ".txt");
        }
        saveToFile(file);
}

private void saveToFile(File file) {
        FileWriter writer = null;
        try {
                writer = new FileWriter(file);

                int size = mainApp.getResultData().size();
                for (int i = 0; i < size; i++) {
                        AlgoritmResult result = mainApp.getResultData().get(i);
                        writer.write(String.valueOf(result.getDepth()));
                        writer.write(" ");
                        writer.write(String.valueOf(result.getIncome()));
                        writer.write(" ");
                        writer.write(String.valueOf(result.getTime()));
                        writer.write("\n");
                }
                writer.flush();
        }
        catch (IOException e) {
                Alert alert = new Alert(AlertType.ERROR);
                alert.initOwner(mainApp.getPrimaryStage());
                alert.setTitle("Ошибка");
                alert.setHeaderText("Ошибка записи");
```

```
                alert.setContentText("Ошибка записи в файл - " + file.getPath());
                alert.showAndWait();
            }
            finally {
                try {
                    if (writer != null) writer.close();
                }
                catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }
        //Проверка на правильность введения входных данных
        private boolean isInputValid() {
            String errorMessage = "";
            if (numberOfAirplanesField.getText() == null || numberOfAirplanesField.getText().isEmpty()) {
                errorMessage += "Не верно указано количество самолетов!\n";
            }
            else {
                try {
                    Integer.parseInt(numberOfAirplanesField.getText());
                }
                catch (NumberFormatException e) {
                    errorMessage += "Не верно указано количество самолетов (должно быть число)!\n";
                }
            }
            if (numberOfFlightsField.getText() == null || numberOfFlightsField.getText().isEmpty()) {
```

```
                errorMessage += "Не верно указано количество авиаполетов!\n";
        }
        else {
                try {
                        Integer.parseInt(numberOfFlightsField.getText());
                }
                catch (NumberFormatException e) {
                        errorMessage += "Не верно указано количество авиаполетов (должно быть число)!\n";
                }
        }
        if (avarageNumberOfSeatsField.getText() == null || avarageNumberOfSeatsField.getText().isEmpty()) {
                errorMessage += "Не верно указано среднее количество мест!\n";
        }
        else {
                try {
                        Integer.parseInt(avarageNumberOfSeatsField.getText());
                }
                catch (NumberFormatException e) {
                        errorMessage += "Не верно указано среднее количество мест (должно быть число)!\n";
                }
        }
        if (avarageTicketPriceField.getText() == null || avarageTicketPriceField.getText().isEmpty()) {
                errorMessage += "Не верно указана средняя цена билета!\n";
        }
```

```
            else {
                try {
                    Integer.parseInt(avarageTicketPriceField.getText());
                }
                catch (NumberFormatException e) {
                    errorMessage += "Не верно указана средняя цена билета (должно быть число)!\n";
                }
            }
            if (avarageCostOfFlightField.getText() == null || avarageCostOfFlightField.getText().isEmpty()) {
                errorMessage += "Не верно указана средняя себестоимость полета!\n";
            }
            else {
                try {
                    Integer.parseInt(avarageCostOfFlightField.getText());
                }
                catch (NumberFormatException e) {
                    errorMessage += "Не верно указана средняя себестоимость полета (должно быть число)!\n";
                }
            }

            /*
            if (depthOfViewField.getText() == null || depthOfViewField.getText().isEmpty()) {
                errorMessage += "Не верно указана глубина просмотра!\n";
            }
            else {
```

```
                try {
                        Integer.parseInt(depthOfViewField.getText());
                }
                catch (NumberFormatException e) {
                        errorMessage += "Не верно указана глубина просмотра (должно быть число)!\n";
                }
        }
        */
        if (errorMessage.isEmpty()) {
                return true;
        }
        else {
                Alert alert = new Alert(AlertType.WARNING);
                alert.initOwner(mainApp.getPrimaryStage());
                alert.setTitle("Ошибка");
                alert.setHeaderText("Неправильно введенные данные");
                alert.setContentText(errorMessage);

                alert.showAndWait();

                return false;
        }
    }
}
```