МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

На правах рукописи

Короткова Юлия Леонидовна

ИССЛЕДОВАНИЕ ЗАДАЧ И РАЗРАБОТКА МЕТОДОВ СИНТЕЗА И РЕГУЛИРОВАНИЯ РАСПИСАНИЙ ДВИЖЕНИЯ ВОЗДУШНЫХ СУДОВ АВИАКОМПАНИИ

Специальность 2.3.1. – Системный анализ, управление и обработка информации, статистика

Диссертация на соискание ученой степени
кандидата технических наук

Научный руководитель: д.т.н., доцент Мезенцев Юрий Анатольевич

Новосибирск – 2022

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Актуальность темы исследования

Одной из традиционных областей применения методов оптимизации является авиация. В настоящее время роль воздушного транспорта значительно возросла. В транспортной системе России в последние годы наблюдается значительное развитие гражданской авиации, характеризующееся стабильным ростом пассажирских перевозок. Увеличение пассажирских перевозок приводит к повышению интенсивности выполнения рейсов, и поскольку расписание – это основа производственного процесса любой авиакомпании, задачи, связанные с планированием и управлением расписаниями, являются наиболее актуальными из всего набора задач управления авиапарком.

Процессы планирования и управления расписаниями могут быть разделены по уровню: стратегические, месячные, оперативные. Стратегическое планирование предполагает синтез расписания на сезон, исходя из коммерчески обоснованного перечня маршрутов, типов воздушных судов, планового времени обслуживания в аэропортах оперирования и планового полетного времени. На этапе месячного планирования выполняется процесс распределения воздушных судов по рейсам, а также решение отдельного блока задач, связанного с планированием экипажей и технического обслуживания. В силу того, что деятельность авиакомпании подвержена изменениям, вызываемым как внешней, так и внутренней средой, плановое расписание нуждается в постоянном управлении и контроле. Процесс регулирования назначений воздушных судов по рейсам в рамках суточного плана полетов авиакомпании называется оперативным управлением.

Задачи управления расписаниями авиакомпании относится к классу задач, изучаемых в рамках теории расписания (раздел оптимизации расписаний параллельно-последовательных систем), являются NP-трудными. Сложность решения задач управления расписаниями обусловлена не только большим количеством переменных, но и необходимостью учета всевозможных ограничений. Можно выделить такие ограничения как необходимость выполнения технического

обслуживания воздушных судов в определенные периоды времени, ограничения по емкостям при изменении типов ВС в случае переназначений, ограничения аэропортов оперирования по параметрам пропускной способности при корректировке времен отправления и прибытия и т.д.

В работе исследуются задачи оптимального оперативного регулирования назначений ВС по рейсам авиакомпании. Оперативное управление расписаниями авиакомпании предполагает внесение корректировок в план на глубину от нескольких часов до нескольких суток, исходя из совокупности ожидаемых или реализовавшихся событий, и заключается в принятии решения о переназначении воздушных судов для выполнения рейсов [1]. При оперативном управлении расписанием в сбойной ситуации, когда имеет место отклонение от запланированного графика, лицо, принимающее решение, (в данном случае – сотрудник центра управления полетами авиакомпании) в критически короткие сроки должен выполнить анализ огромного количества возможных сценариев с учетом ряда ограничений и условий. Время принятия решения о переназначении ВС не должно превышать критического значения для обеспечения своевременного реагирования на изменившиеся условия. Неэффективное управление приводит к финансовым потерям авиакомпаний, связанным с дополнительными расходами при задержках рейсов и необходимостью дополнительного обслуживания пассажиров (например, предоставление питания и гостиницы в случае длительной задержки). Противоречие между трудоемкостью и необходимостью решения задачи в реальном времени обосновывает актуальность темы исследования.

Степень разработанности темы

В ходе диссертационного исследования был изучен широкий круг научных работ отечественных и зарубежных авторов, в том числе классические труды в области дискретной оптимизации и смежных разделах, работы современных авторов, теоретиков и практиков. Развитием данного направления науки занимались отечественные и зарубежные ученые: В.С. Танаев, В.А. Струсевич, Э.Х. Гимади, В.С. Канев, С. В. Севастьянов, А.В. Кононов, Ю.А. Мезенцев и др.

Исследование прикладных задач управления графиками движения воздушных судов в случаях отклонения от заданного плана началось в 1980-х годах и продолжается в настоящее время. С 1984 года по июнь 2020 года опубликовано, в общей сложности, 110 статей. Это работы таких ученых, как J. Abara, D. Teodorovic, A. Jarrah, M.F. Arguello, J.M. Cao, S. Yan, K.T. Talluri, J.P. Clarke, C. Barnhart, Y. Hu, J.F. Zhu, Zh. Wang, M. Grönkvist, T. Andersson, N. Eggenberg и др. При этом, только за последние 10 лет опубликовано более 50 % работ. Это позволяет утверждать, что интерес к решению проблем управления расписаниями в сбойных ситуациях растет.

Анализ научных публикаций по теме исследования позволяет утверждать, что, несмотря на значительное количество работ, непосредственно посвященных задаче управления расписаниями в сбойных ситуациях (при отклонении от запланированных графиках), достигнутые к настоящему времени, как теоретические, так и практические результаты едва ли можно считать удовлетворительными. Это следует из того, что ни один из существующих подходов не гарантирует получения оптимального решения даже при обособленном рассмотрении задачи переназначения ВС, без учета требования по переработке графиков работы экипажей, маршрутов следования пассажиров и т.д., так как применяются либо весьма приближенные алгоритмы, либо эвристики, неконтролируемо сужающие пространство поиска. Также можно сделать вывод об отсутствии алгоритма нахождения цепочек оптимальных назначений для каждого ВС с доказанной эффективностью и должным быстродействием. Таким образом, для эффективного оперативного управления расписанием авиакомпании оказывается актуальной задача разработки инструментария синтеза и регулирования назначений и расписаний флота ВС в режиме реального времени. Разработка и применение эффективного метода в системах поддержки принятия решений позволит минимизировать потери авиакомпаний.

Объект исследования – данные о структуре расписания, факторы, влияющие на принятие решения о корректировке назначений ВС на рейсы.

Предмет исследования – модели и алгоритмы принятия решения о переназначении ВС в рамках оперативного управления расписаниями.

Целью диссертационной работы является разработка вычислительно эффективного метода для обоснованного принятия решения по оптимизации назначений ВС по рейсам в рамках суточного плана полетов, исходя из совокупности ожидаемых или реализовавшихся событий. Под вычислительной эффективностью понимается трудоемкость поиска решения рассматриваемой задачи дискретной оптимизации.

Решение задачи заключается в определении однозначного соответствия рейсов и конкретных воздушных судов, которые будут выполнять эти рейсы, при условии минимизации отклонения расписания от плановых параметров, а также выполнении производственных ограничений, таких как обеспечение минимального времени для наземного обслуживания в аэропорту между рейсами, ограничение по количеству выполняемых рейсов одним ВС и т.д.

Для достижения поставленной цели были выделены следующие **задачи** исследования:

1. Формализация задачи управления назначениями ВС, относящейся к классу задач теории расписаний, а именно – к задаче оптимизации расписаний параллельно-последовательных систем с задержками начала обслуживания и неопределенными маршрутами обслуживания.
2. Критический анализ существующих методов, моделей и программных средств для решения задачи управления назначениями воздушных судов.
3. Разработка вычислительно эффективных методов решения задачи оптимального управления расписанием авиакомпании в режиме реального времени.
4. Исследование и определение критерия эффективности оперативного управления парком воздушных судов авиакомпании.
5. Программная реализация разработанного алгоритма.

6. Апробация разработанного алгоритма на данных реальной размерности действующей авиакомпании. Оценка эффекта от применения разработанного инструментария, экспериментальное доказательство эффективности использования алгоритма.

Научная новизна работы определена следующими результатами:

1. Разработана постановка задачи регулирования расписаний параллельно-последовательных систем с задержками начала обслуживания, отличающая учетом фактора формирования связных технологических маршрутов, заранее не определенных, и позволяющая решать задачу любых размерностей. Таким образом одновременно решаются: динамическая задача маршрутизации и задача синтеза приближенного к оптимальному расписания.

2. Разработан алгоритм решения дискретной задачи оптимизации управления расписанием авиакомпании, отличающийся высокой эффективностью с точки зрения быстродействия и подходящий для решения задач любой размерности. Предложенный алгоритм основан на декомпозиции, линейной релаксации и быстром поэтапном решении, переводящим задачу в класс полиномиально разрешимых, что принципиально отличает его от всех известных ранее.

3. Предложен принципиально новый, ранее не применявшийся критерий для оценки эффективности полученного расписания, основанный на применении риск-ориентированного подхода. Отличие данного подхода от всех ранее использовавшихся заключается в нахождении баланса между количеством и величиной задержек, что позволяет получить устойчивое расписание.

Теоретическая значимость результатов исследования заключается в том, что получила развитие концепция оптимального управления сложными технологическими системами, реализуемая посредством инструментария моделей и методов дискретной оптимизации. В частности, сформулированы NP-трудные задачи управления технологической системой, решение которых стало возможным в режиме реального времени за счет развития и программной реализации методов неполной декомпозиции в сочетании с алгоритмами смешанного целочисленного

программирования, эвристическими процедурами релаксации и разбиения исходного множества альтернатив.

Практическая значимость результатов исследования заключается в том, что предложенный подход может применяться в авиакомпаниях любого размера для принятия решения о необходимости корректировки назначений ВС в случае отклонения от суточного плана полетов. Применение разработанного программного комплекса позволяет повысить эффективность системы принятия решений авиакомпании за счет применения методов оптимизации и эффективных критериев, а также сократить суммарное время потенциальных задержек на 50 %.

Результаты работы внедрены на предприятии транспортной отрасли, в АО «Авиакомпания «Сибирь». Помимо разработанного программного комплекса, на предприятии также на систематической основе используется предложенная методика оценки расписания авиакомпании по векторному критерию минимизации отклонений от действующего расписания и нарушению общей пунктуальности рейсов авиакомпании используется, которая позволяет оценить риск недостижения цели по выполнению показателя регулярности полетов.

Кроме того, результаты работы внедрены в учебный процесс Новосибирского государственного технического университета на кафедре автоматизированных систем управления в курсах «Системный анализ и исследование операций», «Методы оптимизации». Оригинальное математическое обеспечение, реализованной среде MS Visual Studio 2015 и IBM CPLEX успешно применяется в научно-исследовательских работах студентов при выполнении квалификационных работ бакалавров и магистерских диссертаций.

Методология и методы исследования

В работе развиваются подходы к созданию вычислительно эффективных алгоритмов быстрого приближенного решения дискретной задачи оптимизации управления расписанием авиакомпании. Основные теоретические результаты работы базируются на точных и приближённых методах системного анализа и исследования операций, непрерывной и дискретной оптимизации, включая

барьерно-ньютоновские методы, методы декомпозиции, линеаризации, ветвлений и отсечений, динамическое программирование, а также ряд эвристических алгоритмов.

Научные положения, выносимые на защиту

1.Постановка задачи синтеза оптимальных расписаний многостадийной параллельно-последовательной обслуживающей системы (применительно к оптимальному оперативному управлению расписаниями авиакомпании).

2.Результаты исследования применимости критерия минимизации уровня риска нарушения пунктуальности для решения задачи оперативного управления расписаниями авиакомпании.

3.Декомпозиционный алгоритм решения задачи оптимизации расписаний параллельно-последовательных обслуживающей системы.

4.Прототип системы поддержки принятия управленческих решений, реализующий представленные методы решения задач управления назначениями воздушных судов.

Соответствие паспорту специальности

Полученные научные результаты соответствует п. 2 «Формализация и постановка задач системного анализа, оптимизации, управления, принятия решений и обработки информации», п. 3. «Разработка критериев и моделей описания и оценки эффективности решения задач системного анализа, оптимизации, управления, принятия решений и обработки информации», п. 4 «Разработка методов и алгоритмов решения задач системного анализа, оптимизации, управления, принятия решений и обработки информации» и п. 9 «Разработка проблемно-ориентированных систем управления, принятия решений и оптимизации технических, экономических, биологических, медицинских и социальных объектов» паспорта научной специальности 05.13.01 – «Системный анализ, управление и обработка информации» (в соответствии с новой номенклатурой научных специальностей - 2.3.1. − «Системный анализ, управление и обработка информации, статистика»).

Степень достоверности результатов исследования

Обоснованность и достоверность научных положений и выводов, содержащихся в диссертационной работе, подтверждается вычислительными экспериментами и полученными апостериорными оценками точности и быстродействия всех разработанных в рамках диссертационной работы алгоритмов оптимизации.

Представленное в диссертационной работе исследование выполнялось при поддержке гранта Российского Фонда Фундаментальных Исследований (проект 19-37-90012) «Информационные технологии и математические методы оптимального оперативного управления графиком движения воздушных судов авиакомпании» и стал одним из составляющих проекта FSUN-2020-0009 (мнемокод 0735-2020-0009) «Моделирование системной организации когнитивных функций с применением интеллектуального анализа массивов психометрических и нейрофизиологических данных».

Апробация результатов диссертации

Результаты диссертационной работы докладывались и обсуждались на международных и всероссийских конференциях: II Всероссийской научно-технической конференции с международным участием им. В.В. Губарева «Интеллектуальный анализ сигналов, данных и знаний: методы и средства» (Новосибирск, 2018 г.); XXV международной научно-технической конференции «Машиностроение и техносфера XXI века» (Севастополь, 2018 г.); Международной научной конференции, посвященной 100-летию со дня рождения академика Е. А. Барбашина (Минск, 2018 г.); V международной конференции и молодежной школы "Информационные. технологии и нанотехнологии (Самара, 2019 г.); 14-ом Международном форуме по стратегическим технологиям IFOST (Томск, 2019 г.); XX Всероссийской конференции молодых ученых по математическому моделированию и информационным технологиям (Новосибирск, 2019 г.); XV Международной научно-технической конференции «Актуальные проблемы электронного приборостроения» (Новосибирск, 2021 г.).

Публикации

По результатам исследований опубликовано 14 научных работ, в том числе 4 статьи в журналах, рекомендованных ВАК, 3 статьи в периодических научных журналах, индексируемых Web of Science и Scopus, а также 2 свидетельства о государственной регистрации программ для ЭВМ («Программа для расчета задержек и корректировки расписания движения воздушных судов авиакомпании», регистрационный номер 2020663452 от 28.10.2020 и «Программа для расчета задержек и корректировки расписания движения воздушных судов авиакомпании», регистрационный номер 2020663452 от 12.10.2021).

Личный вклад автора

Все представленные в диссертации результаты исследований получены лично автором или при его непосредственном участии. Доля личного вклада в работах, выполненных в соавторстве, составляет не менее 70%. Автор внёс определяющий вклад в постановку задач, анализ существующих подходов к решению, проведение вычислительных экспериментов на реализованных алгоритмах синтеза решений по переназначению воздушных судов, интерпретацию полученных результатов.

Структура и объем диссертации

Диссертация состоит из введения, пяти глав, заключения, перечня условных обозначений и сокращений на двух страницах, списка литературы 97 наименований и шести приложений. Общий объем работы составляет 191 страницу, в том числе основной текст на 125 страницах, 24 рисунка и 40 таблиц.

ГЛАВА 1 ПРОЦЕСС УПРАВЛЕНИЯ РАСПИСАНИЯМИ АВИАКОМПАНИИ КАК ОБЪЕКТ ИССЛЕДОВАНИЯ

В главе рассматривается комплексный процесс управления расписаниями авиакомпании. Выполнен аналитический обзор задач, решаемых в рамках общего процесса управления расписанием: построение маршрутной сети и распределение типов воздушных судов по направлениям оперирования при планировании сезонного расписания, планирование движения ВС с учетом необходимого ТО и планирование экипажей на ежемесячной основе, распределение ВС по рейсам в рамках оперативного управления расписанием.

Выполнена содержательная постановка задачи оперативного управления назначениями ВС на рейсы. Представлена историческая справка и аналитический обзор существующих подходов к решению задачи оптимального оперативного управления расписания. Выполнено развернутое обоснование актуальности исследования и необходимость разработки вычислительно эффективных методов решения дискретных задач оптимизации управления сложными производственными процессами, к которым относится оперативное управление назначениями воздушных судов на рейсы в рамках суточного плана полетов на глубину от нескольких часов до нескольких суток.

1.1. Задачи управления расписаниями флота авиакомпании

Результатом процесса формирования расписания авиакомпании является график рейсов, характеризующийся перечнем конкретных направлений полетов, частотами и конкретными временами вылета и прибытия каждого рейса. Структурированное описание процесса формирования расписаний представлено в работе [22], в которой приведено деление на оперативное, тактическое и стратегическое планирование. В работе описываются сроки, в которые производится планирование расписания на том или ином этапе, а также перечислен набор основной входной и выходной информации. Расписание рейсов является

ключевым объектом управления авиакомпании и обеспечивает основу для планирования ресурсов, необходимых для производства полетов.

Большинство авиакомпаний при организации производственных процессов руководствуются рекомендациями IATA. Это неправительственная организация авиаперевозчиков, занимающаяся вопросами организации и регулирования воздушных перевозок, была основана в апреле 1945 г. В библиографическом списке под номером [23] представлен один из важнейших документов IATA, согласно которому авиакомпании по всему миру выстраивают свои производственные процессы, руководствуясь представленными в нем требованиями. IATA.

В [23] указано, что авиакомпании формируют расписание и осуществляют производственную деятельность по сезонам. Выделяется два сезона: летний сезон (это период времени с последнего воскресения марта по последнее воскресение октября) и зимний сезона (период времени с конца летнего IATA-сезона с по последнюю субботу марта).

Таким образом, существенные изменения расписание претерпевает, как минимум, дважды в год, при планировании IATA-сезона, чтобы отразить маркетинговые цели и скорректировать его под разные модели путешествий между зимними и летними месяцами.

Далее, на ежемесячной основе вносятся незначительные изменения на основании информации об изменившихся потребностях рынка, изменении расписания и цен конкурентов, а также изменении в ключевых ресурсах, таких как количество воздушных судов, количество экипажей, ограничения по пропускной способности в аэропортах оперирования и т.д. Большинство изменений заключается в корректировке частоты выполнения рейсов на определенных направлениях полетов. Это, в свою очередь, может потребовать корректировки времени вылета рейсов таким образом, чтобы были сохранены запланированные стыковочные маршруты. При внесении изменений нужно также учитывать тот

факт, что спрос на воздушные перевозки очень чувствителен к времени отправления с хорошо известными пиками в утренние и дневные часы.

В конечном счете, расписание должно быть разработано таким образом, чтобы обеспечить максимальную рентабельность, снабжая рынок соответствующими емкостями с учетом ограничений доступных емкостей и кадровых ресурсов [1].

Укрупненно процесс управления расписаниями авиакомпании можно представить в виде схемы (рисунок 1.1.), на которой отражены ключевые задачи по периодам выполнения.

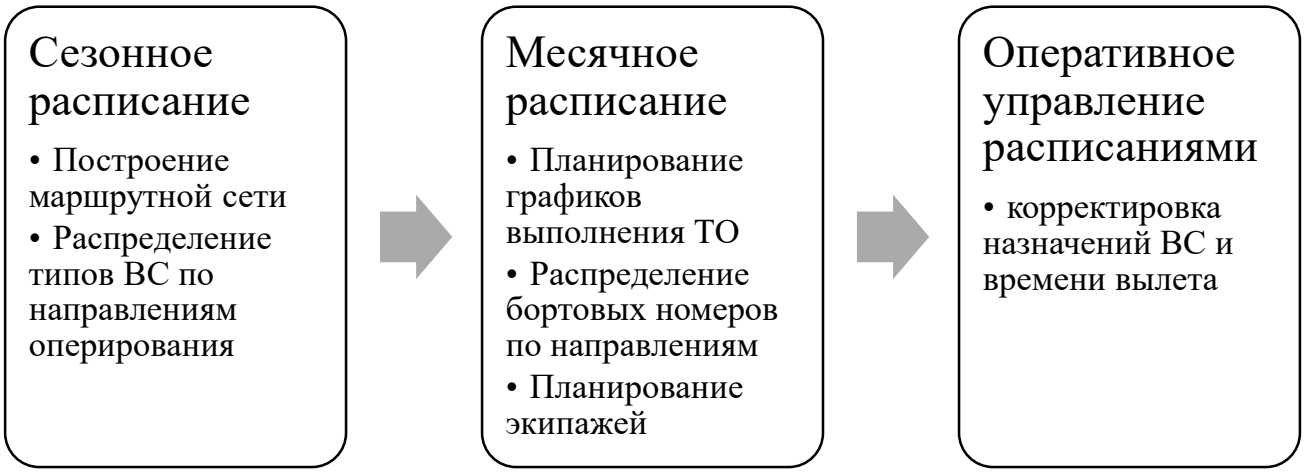


Рисунок 1.1 – Общая схема управления расписаниями.

Результатом процесса формирования расписания, согласно рекомендациям IATA, является SSIM-файл, содержащий сведения обо всех направлениях полетов на плановый период. SSIM – стандартизированный формат текстового файла с расписанием, позволяющий авиакомпаниям обмениваться данными с аэропортами и системами бронирования.

Для любого SSIM-файла с расписанием обязательно указание следующих характеристик в начале файла: обозначение IATA-сезона (например, если в файле предоставляется летнее расписание на 2018 год, то будет обозначение «s18»); указание временной зоны, в соответствии с которой отражается время вылета и прилета – LOC или UTC.

Минимальный состав полей представлен в таблице 1.1.

Таблица 1.1 – Минимальный параметрический состав полей SSIM-файла

Параметр	**Описание**
Рейс	Рейс обозначается в формате «XX0000», где xx - это IATA-код авиакомпании, а четырёхзначное число является номером рейса.
Период выполнения рейса	Даты начала и окончания периода обозначаются в формате ДДММММГГ.
Частота выполнения рейса	Указывается через указание номеров дней недели, в которые выполняется рейс. Если в какой-либо день недели рейс не выполняется, указывается «_».
Тип ВС	Это категория, объединяющая определенные классы ВС, обусловленных технико-экономическими характеристиками (дальность полета, количество пассажиров, схемы рассадки, период эксплуатации и др.). Авиакомпании оперируют международными кодировками типов ВС, предоставляемыми IATA.
Аэропорт отправления/ прибытия	Пункт отправления обозначается международными кодами аэропортов, предоставляемыми IATA. Например, OVB – Новосибирск, аэропорт Толмачево, а DME – Москва, аэропорт Домодедово. Аэропорт вылета и аэропорт прилета вместе составляют участок перелета – направление, маршрут, в соответствии с которым выполняется рейс.
Плановое время вылета и прилета	Указывается в формате «ччмм» в соответствии с указанной временной зоной (LOC или UTC)

Тип рейса	Выделяются следующие основные типы рейсов: регулярный «j», чартерный «c» рейсы.
Конфигурация ВС	Это количество кресел в бизнес-кабине ВС, указывается после «C» и количество кресел в кабине эконом-класса, которое указывается после «Y».

С целью обеспечения возможности аналитической обработки расписаний, авиакомпании выполняют обработку данных из SSIM-файла и загружают их в базы данных в табличном формате, где каждой строке соответствует один сегмент (таблица 1.2 и 1.3).

Важно отметить, что *сегмент* – это упорядоченная пара аэропортов, через которые проходит плановый перелет рейса. При этом *рейс* – это набор участков перелета (сегментов), для которых зафиксировано время вылета и которые объединены одним номером.

Таблица 1.2 – Пример расписания авиакомпании

AIRCRAFT	**STD**	**STA**	**BLK**	**FLT**	**DEP**	**ARR**	**CAPY**	**CAPC**
320	19.01.2018 0:35	19.01.2018 5:00	265	812	TOF	DME	150	8
320	19.01.2018 7:15	19.01.2018 10:35	200	169	DME	OMS	150	8
320	19.01.2018 11:40	19.01.2018 15:15	215	170	OMS	DME	150	8
320	19.01.2018 16:30	19.01.2018 20:00	210	693	DME	VRN	150	8

Таблица 1.3 – Расшифровка атрибутов SSIM файла.

Атрибут	**Содержательное описание**
AIRCRAFT	Трехсимвольный код типа ВС, присваиваемый IATA (например, 320 – Airbus 320).

	Обозначает категорию, объединяющую ВС по ряду технико-экономическим характеристикам (дальность полета, количество пассажиров, схемы рассадки, период эксплуатации и др.).
STD	Время отправления рейса по расписанию. Время, установленное для начала движения ВС со стоянки. Согласовывается в аэропорту отправления, как слот для отправления и указывается в системах бронирования и авиабилете.
STA	Время прибытия рейса по расписанию. Время, установленное для остановки ВС на месте стоянки. Согласовывается в аэропорту прибытия, как слот для прибытия.
BLK	Полётное время рейса по расписанию. Это время от отправления рейса по расписанию до прибытия рейса по расписанию. Полетное время включает в себя сумму летного времени и времени руления на прилет и на вылет: Летное время – это время от момента отрыва шасси ВС от ВПП до момента касания шасси ВПП Время руления ВС на вылет - это время движения ВС от места стоянки до момента отрыва ВС от ВПП Время руления ВС на прилет - это время движения ВС с момента касания ВПП до установки ВС на место стоянки.
FLT	Номер рейса авиакомпании
DEP	IATA-код аэропорта вылета. Уникальный идентификатор аэропорта Трёхбуквенный уникальный идентификатор, присваиваемый IATA. Например, KZN – Казань.
ARR	IATA-код аэропорта прилета. Уникальный идентификатор аэропорта Трёхбуквенный уникальный идентификатор, присваиваемый IATA. Например, KZN – Казань.

CAPY	Количество доступных для перевозки мест в эконом-классе
CAPC	Количество доступных для перевозки мест в бизнес-классе

В строке 1 таблицы 1.2 наблюдаем рейс, который выполняется под номером 812 по маршруту Томск, аэропорт Богашёво – Москва, аэропорт Домодедово на ВС типа Airbus 320, которое имеет 8 кресел бизнес-класса и 150 кресел эконом-класса. Плановое время вылета рейса – 00:35, полетное время составляет 4 часа 35 минут.

1.1.1. Распределение типов ВС по направлениям оперирования

Как отмечено ранее, расписание авиакомпании должно быть составлено таким образом, чтобы максимизировать совокупный возможный доход от выполнения рейсов при наличии тех или иных ограничений. С целью обеспечения максимальной эффективности производственного процесса, все ресурсы авиакомпании должны использоваться в полном объеме, с минимальным временем простоя. Так, например, в пиковые месяцы занятость воздушных судов может достигать 93% (рисунок 1.2), это означает, что лишь 2 часа в сутки воздушное судно находится в состоянии покоя.

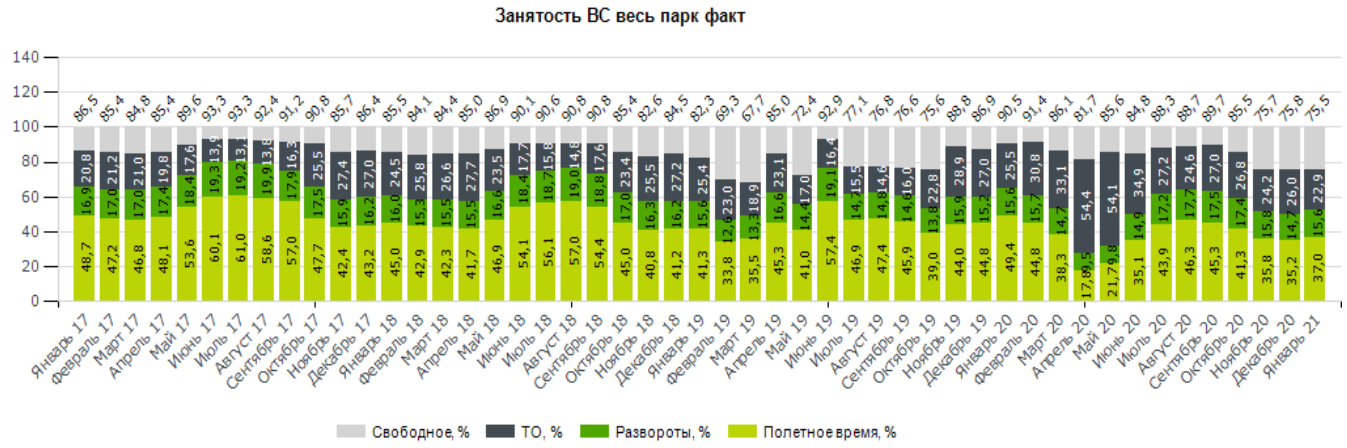


Рисунок 1.2 – Занятость ВС на примере одной из авиакомпаний РФ

Именно с целью повышения эффективности использования ресурсов одной из первых и важнейших задач при построении расписания является распределение типов ВС по направлениям оперирования. Решение заключается в присвоении

конкретного типа ВС по каждому направлению оперирования, максимально сокращая разницу между предоставляемым количеством емкостей (кресел) и прогнозируемым спросом (пассажиропотоком).

Воздушные суда одного типа – это самолеты от одного и того же производителя, которые имеют одинаковую емкость и могут управляться одним и тем же кабинным экипажем. Таким образом, Boeing 738 будет одним типом, а Airbus 320 - другим. При этом, даже Boeing 737-300 и 737-400 могут рассматриваться как различные типы из-за разницы в количестве пассажирских мест.

Задача распределения типов ВС по направлениям усложняется тем, что, как правило, авиакомпании имеют несколько типов воздушных судов, которые могут быть приняты не всеми аэропортами и могут обслуживаться экипажами, имеющими соответствующие допуски. Перечень типов ВС, которые могут быть приняты конкретным аэропортом оперирования определяется реестром допусков рассматриваемого аэропорта. Оценка пригодности аэропортов для приема и выпуска зарубежных типов воздушных судов производится комиссией Регионального управления ФАС России. Комиссия рассматривает состояние пригодности аэропорта, аэродрома, оборудования и служб для приема, выпуска и обслуживания конкретного типа воздушного судна.

Решение задачи распределения типов выполняется с помощью FAM-моделей и позволяет повысить экономическую эффективность за счет сокращения расходов, вызванных наличием избыточных емкостей и повышением доходности ввиду возможности использования более гибкой тарифной политики. Данная задача актуальна исключительно при планировании сезонного расписания.

Справку по ретроспективе исследований в данном направлении также можно получить в работах [2,3]. Приводим краткое хронологическое перечисление работ по тематике.

Впервые попытка решить задачу оптимального управления расписанием была предпринята в 1989 году J. Abara для авиакомпании American [24], в том числе поднят вопрос о распределении типов ВС по направлениям оперирования.

В [25] Subramanian и соавт. (сотрудники) Delta Airlines также описывается аналогичная задача оптимального распределения парка воздушных судов по направлениям оперирования, представлена математическая модель. Авторы делают оценку ожидаемого эффекта от реализации оптимального алгоритма – более 300 000 долларов в день. Спустя год, в работе [1] упоминаются результаты апробации модели моделей оптимального распределения флота в авиакомпании Delta (1991). Согласно представленному отчету экономия затрат в летние месяцы достигала 220 000 долларов в день только за счет моделирования назначений. В этой же работе говорится о том, что аналогичная модель была использована в авиакомпании USAir (1993), которая отчиталась об экономии порядка 20 долларов с каждой минуты задержки [4].

В [26] представлена классическая постановка задачи распределения парка воздушных судов по направлениям с целью сокращения расходов исходя из ожидаемого спроса на направлении и расходов на кресло-километр.

В [27] задача распределения воздушных судов по направления расширена, в результате решения представленной модели формируются цепочки рейсов для каждого типа воздушного судна. Авторы уделяют особо внимание анализу вариантом сокращения размерности задачи за счет разбиения множества рейсов на группы.

В [28] задача распределения воздушных судов по направлениям рассматривается как подзадача более общей постановки. В качестве целевой функции автор рассматривает сокращение налета, обосновывая это возможностью сокращения расходов через оптимизацию использования парка.

Разработка оптимизационных алгоритмов решения FAM-моделей продолжается до сих пор, наиболее активными в этой области являются исследователи Massachusetts Institute of Technology и Georgia Institute of

Technology. В России активные попытки исследования задачи распределения парка предпринимались в 2008 – 2016 годах и продолжаются до сих пор.

1.1.2. Планирование графика движения ВС в рамках месячного расписания

Как отмечено ранее, до момента планирования графиков движения воздушных судов, в расписании определены только типы воздушных судов по направлениям движения. На этапе построения графиков движения воздушных судов на каждый конкретный вылет рейса назначается конкретный бортовой номер воздушного судна таким образом, чтобы обеспечить выполнение необходимо технического обслуживания [29, 30]. Построение графиков движения ВС выполняется на горизонт от одного до двух месяцев. Таким образом, для каждого конкретного воздушного судна в парке формируется цепочка рейсов - последовательность рейсов, выполняемых самолетом в течение дня, построенная с учетом необходимости выполнения требований по техническому обслуживанию.

Техническое обслуживание (ТО) авиационной техники – это комплекс работ, выполняемый для поддержания летной годности ВС при его подготовке к полетам, а также при обслуживании воздушного судна и его компонентов после полетов, при хранении и транспортировке. Частота ТО зависит от сочетания летных часов и количества циклов взлета и посадки и может быть выполнена на любой площадке, оборудованной надлежащим образом. Во многих авиакомпаниях используется классификация ТО, формируемая на основании таких характеристик, как объем, продолжительность и частота ТО [31, 32]. В таблице 1.4 представлена типовая классификация видов работ по техническому обслуживанию ВС.

Таблица 1.4 – Типовая классификация видов работ по ТО ВС.

Вид работ	**Описание**
Transit check	Самая простая форма сервисного обслуживания самолёта, выполняется перед каждым вылетом воздушного судна.

Daily Check	Ежесуточная проверка технического состояния воздушного судна, должна выполняться каждые 24 часа.
Weekly Check	Выполняется приблизительно раз в неделю, не требует обязательного наличия помещения, т.е. ангара. Как правило, длительность данного вида ТО составляет 3-4 часа.
A-check	Производится примерно раз в месяц или каждые 500 часов налёта. Как правило, выполняется в ночное время суток в ангаре базового аэропорта. Содержание данного вида работ зависит от типа самолёта, количества циклов (т.е. взлётов и посадок) или количества часов налёта с момента последней проверки.
B-check	Комплексное ТО, осуществляется примерно каждые 3 месяца
C-check	Является более сложной формой ТО и выполняется каждые 15-20 месяцев или 4 000 часов налёта. Для выполнения данного вида работ требуется вывести самолёт из эксплуатации на некоторое время (порядка 2-х недель), а также обязательным условием выполнения работ является наличие ангара.
D-check	Одна из самых тяжёлых форм обслуживания воздушного судна. Проверка выполняется приблизительно раз в 12 лет и длится 30-40 дней. В это время выполняется комплексная проверка всего воздушного судна, его узлов и деталей. Узлы, выработавшие ресурс или не прошедшие проверку, подлежат замене.

Что касается исследований области оптимизации графиков движения воздушных судов с учетом требований о выполнении необходимого ТО, то можно сказать, что это задача – всего лишь подвид задач о назначении воздушных судов.

В [33] приводится перечень основных источников по данной теме. Это работы T.A. Feo и J. F. Bard (1989), N. M. Kabbani и B.W. Patty (1992), G. Desaulniers и соавт. (1997), J. P. Clarke и соавт. (1998), C. Barnhart (1998), K. Talluri (1998).

Авторы статьи отмечают, что модели планирования графиков движения воздушных судов с учетом ТО предполагают, что график работы будет повторяться каждый день и самолеты, которые находятся в базовом аэропорту, имеют возможность пройти техническое обслуживание.

При этом эффект от использования таких моделей посчитать крайне сложно. Также важно, что модели, акцентирующие внимание на своевременности выполнения ТО, не пригодны для использования в рамках оперативного управления, так как не учитывают последствия задержек и отмен рейсов.

1.1.3. Планирование экипажей

В соответствии с перечнем, утвержденным Минтрансом России, экипажи относятся авиационному персоналу гражданской авиации. Согласно [34], «авиационный персонал — лица, имеющие специальную подготовку и осуществляющие деятельность по обеспечению безопасности полетов воздушных судов или авиационной безопасности, деятельность по организации, выполнению, обеспечению и обслуживанию воздушных перевозок и полётов воздушных судов, авиационных работ, организации использования воздушного пространства, организации воздушного движения и включены в перечни по видам авиации».

Экипажи могут быть классифицированы следующим образом: летный экипаж (пилоты, штурманы) и кабинный экипаж (бортпроводники).

При этом кабинный и летный экипажи могут быть сертифицированы для полетов одного или нескольких типов ВС. Например, одному и тому же экипажу может быть разрешено летать как на самолетах серии Boeing 737-300, так и на самолетах серии Boeing 737-400, поскольку они имеют одинаковую конфигурацию кабины [1], при этом запрещено летать на самолетах Airbus. Данное ограничение обязательно к учету при решении задач планирования экипажей.

Задачи, связанные с планированием экипажей, были подняты Abara J. в его первой работе [24]. Цель процесса планирования экипажей состоит в том, чтобы назначить имеющиеся экипажи на рейсы с соответствующими типами воздушных

судов таким образом, чтобы общие расход, связанные с экипажами для всего расписания были сведены к минимуму при удовлетворении условий доступности и норм режима труда и отдыха экипажей [35]. К расходам на экипажи относится не только оплата труда, но также расходы, связанные с размещением экипажей в случае длительного ожидания следующего рейса после завершения предыдущего вне базового аэропорта.

Оптимальное расписание экипажа должно быть максимально плотным, так, чтобы экипажи проводили большую часть своих рабочих часов в полете. Заработная плата летного состава является одной из существенных статей расходов авиакомпании, которая зависит не только от налета, но и от количества посадок, ночных вылетов, общего рабочего времени вне базы, минимальной гарантированной оплаты труда [27]. Подобная оптимизация позволяет экономить миллионы долларов в год [1].

В работах [36, 37] задача назначения экипажей разбивается на два последовательных этапа: Airline Crew Pairing Problem (ACPP) и Airline Crew Rostering Problem (ACRP).

На первом определяется минимальный набор летных заданий, требуемый для того, чтобы обслужить все рейсы полетного расписания на горизонт планирования с учетом всех требований.

На втором - каждому члену экипажа, в соответствии с его параметрами (допуски, время отдыха, класс обслуживания и др.), назначается индивидуальное летное задание. Решение этих оптимизационных задач чрезвычайно трудоемко и требует значительных вычислительных ресурсов.

1.1.4. Оперативное управление расписанием

Потребность в регулировании расписаний возникает из-за появления непредвиденных задержек вылета рейсов. Плохие погодные условия, загруженность аэропортов и технические проблемы - вот лишь некоторые из причин, которые мешают авиакомпаниям четко выполнять расписание рейсов в

соответствии с планом. Возможны отмены рейсов, задержки вылетов и прибытия. Эти нарушения в работе называются сбойными ситуациями.

Сбойные ситуации в значительной степени влияют на экономические показатели авиакомпаний. Так, например, нарушения выполнения плана полетов могут привести к значительному увеличению эксплуатационных расходов путем возникновения следующих видов расходов: дополнительных расходов на оплату сверхурочной работой экипажей, представителей авиакомпании; дополнительных расходов на топливо; расходов, связанных с дополнительным обслуживанием пассажиров (например, размещение в гостинице, предоставление питания, доставка пассажиров и т.д.).

Стоимость одной минуты задержки рейса оценивается ~2000 рублей (рисунок 1.3). Оценка выполнена экспертами одной из российских авиакомпаний.

Для того, чтобы оценить масштабы проблематики, связанной с необходимостью оперативного регулирования расписания обратимся к статистике нарушения графиков вылета рейсов – рейтингу OAG. OAG - глобальный поставщик данных о перевозках со штаб-квартирой в Великобритании.

Рисунок 1.3 – Оценка стоимости 1 минуты задержки рейса

Согласно статистике, за 2017 – 2020 года в среднем 15% рейсов прибывали в аэропорт назначения с задержкой более 15 минут (таблица 1.5). Уточним, что с марта 2020 по январь 2021 данные не публиковались в связи с малым количеством рейсов.

Таблица 1.5 – Статистика регулярности по данным OAG

Авиакомпания / Месяц	S7	SU	U6	UT	DP	BA	JL	LH
янв.17	73,8	64,3	67,8	80	73,9	78,5	84	67,6
фев.17	84,7	71,6	77,2	81,1	80,6	83,6	79	86,4
мар.17	90,1	82,7	84,6	87,7	89,1	82,3	86,9	86,8
апр.17	86,1	81,6	75,8	88,7	83,3	86	85,9	80,9
май.17	85,6	86,1	71	91,9	89,4	80,4	89,9	78,8
июн.17	71,8	78	63,6	84,4	84,2	77,6	89,3	74,3
июл.17	68,8	74,8	57,4	90,4	82,6	75,9	88,5	73,1
авг.17	75,4	83,2	59,6	91,6	79,4	77	81,9	75,9
сен.17	77,8	84,8	52,9	90,6	79,1	74,5	86,3	70,2
окт.17	84,8	81,5	68,4	90,7	82,8	78,6	83,1	76,4
ноя.17	85	80,8	65,9	88,1	86,1	82,8	82	84,2
дек.17	69,7	66,2	50,7	76,3	74,6	69,2	82,5	68,7
янв.18	76,2	70,9	66,8	82	81,5	80,9	77,3	81,9
фев.18	78,7	70,2	70,2	84,9	84,7	78,5	82,7	72,8
мар.18	84,2	81,2	66,4	87,5	87,9	74,8	84,5	70,7
апр.18	88,5	87,3	77,2	87,7	78	79	87,5	73,9
май.18	90	89,4	81	93,8	88,7	76,4	88,2	65,2
июн.18	81,8	88,9	73,7	87,6	81,8	74,9	87,1	58,8
июл.18	81,9	86,3	69,9	87,9	82,4	68,8	80,3	63,2
авг.18	80,6	89,7	63,1	90,7	84,7	74,3	78,9	68,3
сен.18	75,9	88,9	61,7	86,5	78,4	79,4	80	65,7
окт.18	86,8	90,1	68	86,6	78,3	74,9	88,6	70,1
ноя.18	85,7	85,4	77,3	89,8	88,6	75,6	90,1	77,6
дек.18	73,4	72,3	71	83,1	84,4	72	83,2	68,4
янв.19	85,2	71,6	75	82,4	79	84	86,9	68,7
фев.19	87,7	81,1	71	82,7	86,1	79,6	85,4	80,4
мар.19	90,3	84,5	76,6	87,2	88,8	73,4	81	75,1
апр.19	89,8	91,2	78,1	90,4	91,5	82,8	85	77,1
май.19	90,7	85,9	77	89,4	90,3	82,5	84,5	70,2
июн.19	84,6	85,4	65	85,5	86,5	71	84,9	65,3
июл.19	80,8	86,7	64,2	85,9	86,5	70,6	84,1	68,9
авг.19	77,6	89	45,7	87	86,6	71	75,3	73,6
сен.19	80	89,2	46,8	82,3	80,7	69,7	83,4	74,1
окт.19	81,6	89,8	44,1	80,8	83,4	76,8	80,2	75,7
ноя.19	86,6	92,1	78	83,3	85,6	81,1	87,9	82
дек.19	75,8	86	71	79,6	74,5	73,8	83,4	78,7
янв.20	82,6	88,1	76,2	80	78	83,6	87,8	86,5
фев.20	81,8	89	79,3	84,7	86,3	66,7	90,3	81
фев.21	73,8	82,7	45,9	65,2	62,7	82,2	91	78,6
мар.21	88,3	92,9	64,6	78,5	85,7	90,5	94,1	85,6
апр.21	87,1	92,5	58,2	77,7	87,5	85,5	95,4	88,9

май.21	90,3	92	49,9	89,1	90,9	92,1	92,6	86,7
Среднее	**82**	**83**	**67**	**85**	**83**	**78**	**85**	**75**

Согласно данным одного из перевозчиков РФ абсолютная регулярность прибытий (доля рейсов, прибывших в аэропорт назначения по расписанию, без задержки) за последние 4 года не превысила 70% (рисунок 1.4).

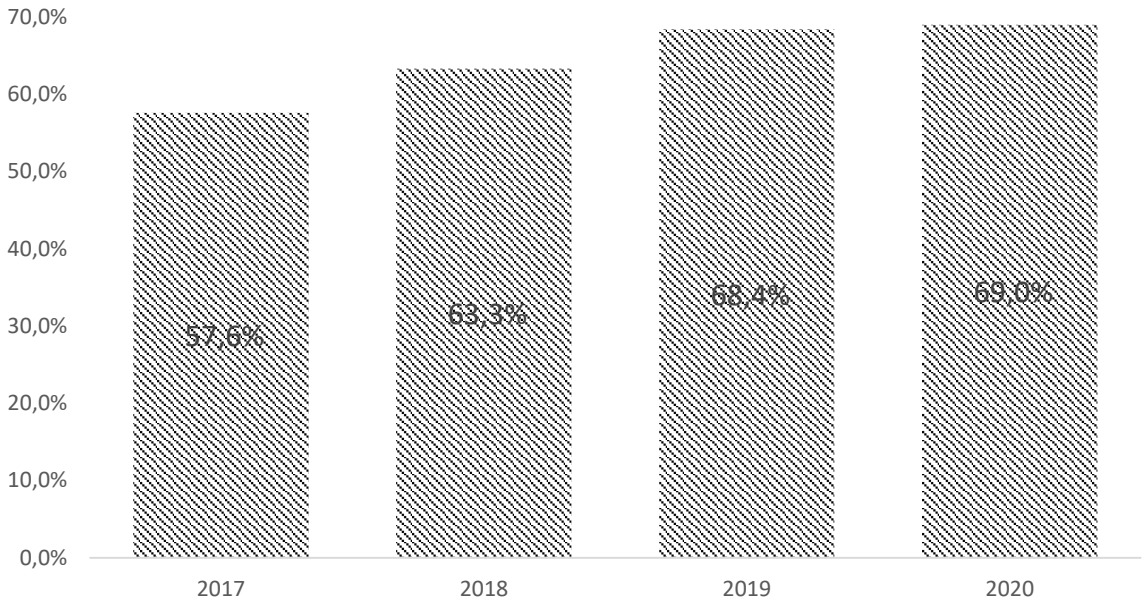


Рисунок 1.4 – График регулярности прибытий за 2017 – 2020 гг. на примере одного из перевозчиков РФ

При этом более половины задержек в интервале от 1 минуты до 15 минут. Длительные задержки более 2 часов составляют 5% от общего количества задержанных рейсов (рисунок 1.5).

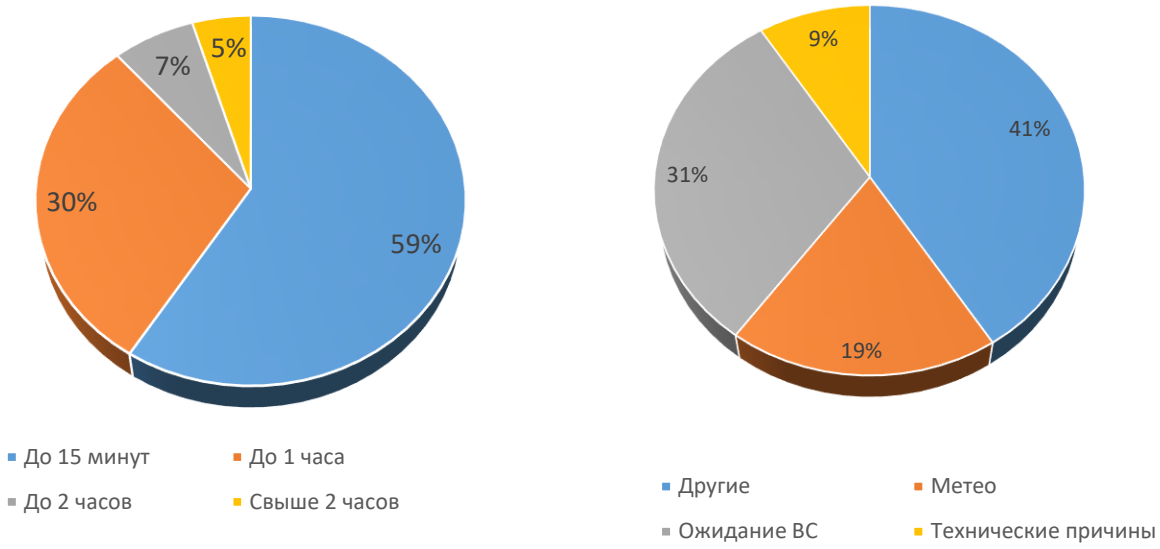


Рисунок 1.5 – Основные причины задержек рейсов

Для того чтобы нивелировать влияние задержек, необходимо выстраивать процессы управления расписаниями воздушных судов и экипажей, в случае сбойных ситуаций.

Содержательно задачу управления переназначениями воздушных судов можно сформулировать следующим образом: учитывая расписание рейсов и информацию о текущих задержках и сбойных ситуациях (например, поломка ВС) определить какие рейсы следует задержать или отменить, а также повторно выполнить переназначение доступных ВС на рейсы таким образом, чтобы минимизировать расходы и имиджевые потери, связанные со сбойными ситуациями.

Как было отмечено ранее, к расходам на сбойные ситуации относятся расходы, связанные с дополнительными эксплуатационными расходами на наземное обслуживание рейсов в аэропортах оперирования (предоставление дополнительных услуг, увеличение времени использования ресурсов и пр.), расходы на обслуживание пассажиров в сбойных ситуациях (предоставление питания, размещение в гостиницах в случае длительных задержек), расходы на компенсации пассажирам в случае обращения в суды и пр.

1.2. Содержательная постановка задачи оперативного управления расписанием

Как отмечено ранее, прикладная задача оптимального оперативного регулирования текущих графиков вылета ВС повседневно актуальна для любой авиакомпании.

Оперативное управление предполагает корректировку графика движения воздушных судов с учетом заданного критерия оптимальности на основании оперативной информации о текущем статусе выполнения рейсов и требует многократного решения в течение суток в реальном масштабе времени.

Именно от эффективности управленческих решений на этапе оперативного планирования зависят такие параметры как прямые эксплуатационные расходы, регулярность рейсов. В отличие от общей задачи составления расписаний вылетов на относительно длительный период, задача оптимального регулирования действующего расписания с одной стороны имеет меньшую размерность, но с другой стороны для регулирования расписания устанавливаются значительно более жесткие временные рамки.

Содержательно задача оперативного управления расписанием заключается в построении такого графика движения для каждого ВС (определение последовательности выполнения рейсов по маршрутам оперирования авиакомпании для каждого воздушного судна) на планируемый период, которое минимизирует суммарное отклонение времени выполнения всех рейсов от планового расписания (либо минимизирует суммарное отклонение по всем рейсам от исходного планового расписания) при условии, что: не будет ни одного рейса с отсутствующим назначением ВС; для каждого рейса будет назначен только один борт; количество распределенных на рейсы воздушных судов не будет превышать входящие ограничения; ВС, назначенные на рейсы, будут логически связаны по времени и местам стыковок.

Для расчетов потребуются нижеследующие входные данные.

1.Ограничения по количеству ВС, доступных для назначения. Количество ВС, которые будут в дальнейшем распределены по рейсам, может быть меньше номенклатурной численности ВС в парке ввиду необходимости резервирования ВС на случай сбойных ситуаций.

2.Плановое расписание ВС, включающее в себя перечень рейсов с уникальными идентификаторами и маршрутами следования, плановые времена отправления и прибытия по каждому рейсу, информация о текущих назначениях ВС на рейсы.

3.Оперативная информация о текущих задержках рейсов на заданный момент времени во всех аэропортах.

4.Справочная информация о минимальных временах наземного обслуживания ВС (МТТ).

Минимальное время наземного обслуживания определяется для каждого аэропорта (исходя из его ресурсов и возможностей), а также для каждого типа ВС (исходя из его технических характеристик) и необходимо для того, чтобы обеспечить требуемый резерв времени для подготовки ВС к следующему рейсу, а также для того, чтобы минимизировать влияние задержек на последующие рейсы, последовательно выполняемые воздушным судном [33].

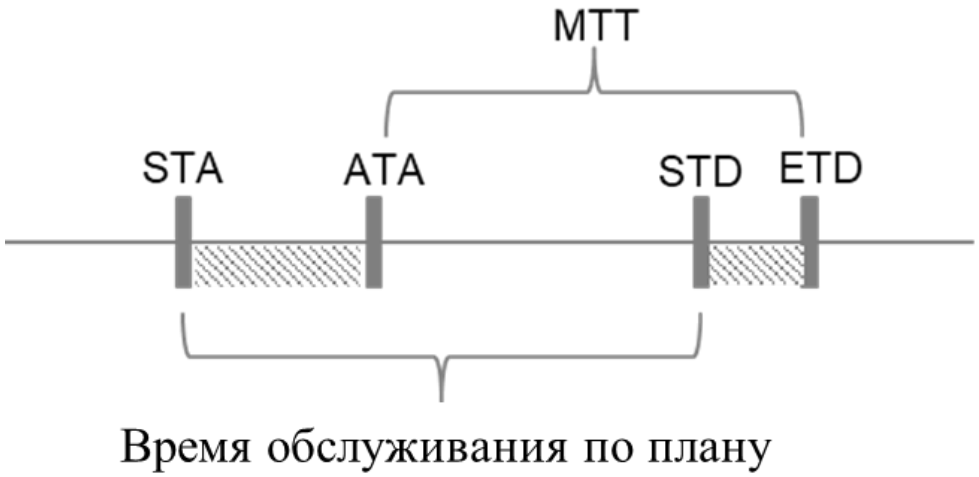


Рисунок 1.6 – Минимизация задержки вылета в случае ППС

Предположим, борт *j* прибыл в аэропорт с задержкой (рисунок 1.6), и начинает подготовку к выполнению рейса *i*, на котором ожидается задержка вылета ввиду позднего прибытия борта *j*. В этом случае аэропортовые службы стремятся выполнить подготовку борта *j* к рейсу *i* в ускоренном режиме. Таким образом, задержка вылета рейса *i* должна быть рассчитана с учетом минимального времени подготовки борта *j*, поскольку благодаря ускоренному обслуживанию, задержка рейса *i* может выть минимизирована или полностью купирована.

В данном случае расчетное время вылета рейса определяется по следующему алгоритму. Если ETA + MTT <= STD, то ETD = STD. В противном случае ETD = ETA + MTT.

1.3. Ретроспектива исследований, направленных на решение задачи оперативного управления расписаниями

Исследование задач восстановления расписаний авиарейсов в случаях отклонения от заданного плана началось в 1980-х годах и продолжается в настоящее время. В работе [42] приводится статистика по количеству работ, освещающих проблемы управления расписаниями. Согласно [42], с 1984 года по июнь 2020 года опубликовано, в общей сложности, 110 статей (рисунок 1.7).

Приведем краткий обзор и классификацию литературы по управлению расписаниями авиакомпаний в сбойных ситуациях, а также оценку тенденций развития методологий решения с акцентом на существующем разрыве между требованиями к инструментам для принятия решения о корректировке расписания и возможностями.

Итак, D. Teodorović и S. Guberinic были одними из первых, кто в 1984 году исследовал задачу восстановления расписания в случае, когда один или несколько самолетов выходят из строя и становятся недоступны для эксплуатации. В данной работе цель заключалась в минимизации суммарного времени задержек. Задача была формализована в виде сетевой модели, в которой узлами являются рейсы, а дугами – задержки. Решение было основано на применении метода ветвей и границ

[43] при ограничении, что авиакомпания эксплуатирует только один тип воздушного судна (т.е. возможна замена любого самолета) и без учета технических ограничений. Модель была протестирована на сети из восьми рейсов, выполняемых тремя самолетами.

Рисунок 1.7 – Публикационная активность по тематике управления расписаниями в сбойных ситуациях

В 1990 году D. Teodorović и G. Stojković задача была модернизирована. Авторы учли в качестве ограничений режим работы аэропорта (когда аэропорт закрывается и не может принимать воздушные суда) и отмену рейсов в качестве возможной меры восстановления расписания. Для решения задачи в данной постановке применен жадный эвристический алгоритм формирования нового суточного плана полетов с учетом проблемы нехватки ВС по критерию минимизации количества отмененных рейсов и задержек [44]. Они разработали эвристический алгоритм, который протестировали на маленькой выборке из 14 ВС и 80 рейсов. Спустя 5 лет (в 1995 году, [45]) авторы вновь расширили задачу, включив в рассмотрение ограничения по экипажам. Они разработали

эвристический алгоритм, основанный на принципе FIFO при планировании рейсов и динамическом программировании, минимизирующий общее количество отмененных рейсов. Модель была протестирована на 240 сгенерированных выборках.

В 1993 Z. Jarrah и др. разработали систему поддержки принятия решений для авиадиспетчеров ЦУП авиакомпании United Airlines [4]. В основе предложенного подхода к решению лежат модели сетевых потоков и алгоритмы теории графов. Было разработано две модели: одна модель использовалась для определения необходимости отмены рейса, другая – для определения изменения времени вылета и прилета рейсов. Принятие решение основывалось на минимизации расходов, связанных с задержками и отменами рейсов. При заданном подходе компромисс между двумя моделями не допускается, что является ключевым недостатком предлагаемого подхода. Однако, эксперты утверждают, что реализованный в системе поддержки принятия решений United Airlines подход позволил сэкономить более 27 000 минут потенциальных задержек, что означает экономию затрат в размере 540 000 долларов США. При этом количество задержанных рейсов с момента внедрения системы поддержки принятия решений сократилось на 50% [46].

Идеи Z. Jarrah находят продолжение в работах J.M. Cao и A. Kanafi, которые занимались разработкой системы принятия решений, способной работать в режиме реального времени [5]. Подход к решению заключался в поиске компромисса между отменой и задержкой рейса с учетом функции максимизации выручки за вычетом расходов, связанных с задержками и отменами рейсов путем применения квадратичного программирования. Модель также предусматривает возможность перегона ВС к месту назначения из другого аэропорта, а также возможность замены типа ВС на рейсе. При этом не учитываются ограничения, связанные с планированием экипажей, техническим обслуживание и т.д.

Стоит отметить исследования K. Talluri [6] в 1996 (разработка эвристического алгоритма принятия решений о замене типа ВС на рейсе по критерию минимизации

расходов), S. Yan и D-H. Yang (формирование графика рейсов после нарушения расписания при использовании симплекс-метода и лагранжевой релаксации [7]), а также S. Yan и Y.P. Tu (отработке нарушений планового расписания при выполнении многоплечевых рейсов для парка с несколькими типами ВС [8]). Вышеперечисленные работы также основаны на применении методов и потоковых алгоритмов в сетях.

Одна из интересных работ с точки зрения выбранной целевой функции была представлена в 1997 году S. Lou и G. Yu [9]. Они занимались проблемой регулярности и сформулировали ее как задачу целочисленного программирования по критерию минимизации процента рейсов с задержкой более 15 минут. Решение было основано на использовании ЛП-релаксации.

В 1997 M.F. Argüello и соавт. разработали эвристический алгоритм жадного случайного поиска новых маршрутов ВС в случае задержки с целью минимизации расходов, связанных с переназначениями и отменами рейсов [10]. При этом в основе была заложена сеть, содержащая два измерения – временные интервалы и аэропорты (Time-Band Network). В 2000 году авторы использовали метод ветвей и границ и ЛП-релаксацию для минимизации расходов, связанных с задержками и отменами [11]. В этом же году часть команды исследователей (J.F. Bard, G. Yu) в соавторстве с B.G. Thengvall представили еще один подход к решению задачи восстановления расписания, в котором задержки и отмены используются для решения проблемы нехватки воздушных судов таким образом, чтобы значительная часть первоначальных маршрутов воздушных судов оставалась нетронутой. Модель была уникальной в той части, что позволяла пользователям самостоятельно задавать цель. Для решения применялся метод ЛП-релаксации, в случае, когда оптимальное решение не находилось, использовалась эвристика. Тестирование показало, что в большинстве случаев решения были близкими к оптимальным [12].

В 2003 году J.M. Rosenberger, E.L. Johnson и G.L. Nemhauser др. [13] предложили очередной эвристический алгоритм для синтеза графиков движения

ВС в рамках каждого типа ВС в отдельности. Алгоритм получил название ASR (Aircraft Selection Heuristic). В качестве целевой функции была минимизация расходов, связанных с переназначениями, задержками и отменами рейсов.

В 2004 году T. Andersson и P. Varbrand [14] презентовали решение задачи управления задержками рейсов с целью максимизации доходов от продажи билетов. Для решения использовался метод Генерации столбцов. Тестирование осуществлялось на данных действующей авиакомпании Швеции и результаты показали, что система может быть использована для принятия решений в режиме реального времени.

В 2006 году C. Barnhart, S. Lan и J.P. Clarke [33] рассмотрели два новых подхода к формированию расписаний с целью минимизации срыва стыковок трансферных пассажиров. Один из подходов включал в себя решение задачи маршрутизации воздушных судов, другой – изменение времени вылета рейса. Рассматриваемая задача маршрутизации состояла в том, чтобы определить последовательность рейсов, называемую маршрутом воздушного судна, которая будет выполняться отдельным воздушным судном таким образом, чтобы каждый рейс включался в один маршрут одного воздушного судна, соблюдая при этом требования к графику выполнения ТО. Поскольку каждый самолет, как правило, выполняет последовательность рейсов, задержка одного рейса может распространиться по маршруту движения ВС к следующим рейсам и вызвать дальнейшие сбои. Авторы классифицировали задержки рейсов разделением на две категории: распространяемые задержки (задержки, возникающие, когда воздушное судно задерживается на предыдущем рейсе, зависят от маршрута самолета) и не распространяемые задержки (задержки, которые происходят по причинам, не зависящим от маршрута).

Основная идея, которая закладывалась в решении задачи маршрутизации заключается в том, чтобы минимизировать распространяемые задержки рейсов. Сформулированная задача была отнесена к стохастической дискретной

оптимизации. Для решения задачи применялись методы ветвей и границ, Генерации столбцов, ЛП-релаксации.

Как отмечено выше, вторая часть исследования была связана с анализом пассажирских стыковок, которые срываются из-за недостаточного времени стыковки и разработкой подходов к построению расписания таким образом, чтобы минимизировать количество сорванных стыковок пассажиров через перепланирование времени вылета в пределах небольшого временного окна. Также авторы затронули возможность развития исследования в направлении разработки интегрированных моделей, учитывающих потребность в одновременном решении нескольких задач – например, задачи распределения воздушных судов и планирования графика обслуживания ВС.

В 2009 Q. Gao, X-W. Tang, J.F. Zhu [47] рассмотрели задачу восстановления расписания и нивелирования задержек рейсов. В своей работе авторы акцентируют внимание на том, что когда размерность задачи достигает 20 самолетов и 80 рейсов, вычислительное время, необходимое для решения превышает 30 минут, что недопустимо для внедрения в практическое использование. Основная идея, которая закладывается авторами в решение заключается в использовании алгоритма жадного рандомизированного поиска (GRASP) и одного из алгоритмов случайного поиска - алгоритма имитации отжига, чтобы минимизировать вероятность попадания в локальное оптимальное решение. Основные недостатки предложенного алгоритма и, как следствие, перспектива развития, как отмечают сами авторы, в том, что целевая функция не содержит всех затрат, необходимых для восстановления расписания нерегулярных рейсов, так как включает только затраты на перевозку и замену парка, выраженные через время задержки.

В 2010 N. Eggenberg и соавт. [15] применили динамическое программирование и метод генерации столбцов для решения задачи синтеза графиков движения ВС с учетом планов по техническому обслуживанию и планируемых стыковок пассажиров по критерию минимизации расходов (операционных, расходов на сбойные ситуации).

В 2010 N. Jafari и соавт. [16] представили задачу смешанного целочисленного программирования, цель решения которой заключалась в том, чтобы одновременно разрешить вопросы восстановления расписания и пассажирских стыковок после сбойной ситуации. Решить задачу удалось лишь на небольшом тестовом наборе данных, который содержал данные по 13 самолетам 2 типов.

В этом же году (2010) T-K. Liu, J-H. Chou, C-H Chen [48] представители гибридный многозадачный генетический алгоритм, позволяющий найти эффективное решение для ежедневных проблем восстановления расписания воздушных судов в случае задержек на коротких рейсах путем оптимизации пяти целевых функций, включающих время разворота, стыковки рейсов, общее время задержки, замена рейсов и максимальное время задержки в 30 минут. Предлагаемый алгоритм использует адаптивный оцененный вектор (AEV) для руководства поиском решения и использует метод многозадачного генетического алгоритма на основе неравенства для обеспечения многозадачного решения. Проведен эксперимент с имитацией помех, временное закрытие аэропорта, и показано, что гибридный метод может обеспечить очень эффективное решение для восстановления расписания в случае возникновения небольших задержек. Неоптимальные решения (с разрывом в оптимальности 4%) были найдены в среднем за 3,6 минуты; для оптимальных решений требовалось в среднем 7,5 минут.

В 2011 году Y. Hu и соавт. [20] сформулировали задачу целочисленного программирования, направленную на минимизацию задержек и расходов, связанных с отменами рейсов и обслуживанием пассажиров в сбойных ситуациях. Решение выполнялось путем ЛП-релаксации с последующим применением эвристического алгоритма. Эта модель была протестирована на данных, которые включали 16 самолетов и 70 рейсов.

В 2012 S. Bisaillon и соавт. [17] разработали эвристический алгоритм решения задачи управления расписаниями в сбойной ситуации, сочетающий перераспределение и ВС, и пассажиров с одной целью - минимизировать

эксплуатационные расходы и влияние на пассажиров. Подход к решению был улучшен в более поздних работах K. Sinclair [18,19].

В 2012 C. Wu и M. Le [49] также занимались задачей восстановления расписания в случае задержек по критериям минимизации расходов при отменах и задержках рейсов. Авторы разработали модель, которая учитывала ряд ограничений и правил, в том числе техническое обслуживание. Решение представлено в виде итеративного дерева, растущего методом комбинации узлов. За счет объединения узлов возможность маршрутизации значительно упрощается. Таким образом, решение формировалось в более разумные сроки (около 4 минут).

В этом же году X. Zhao и Y. Guo [50] разработали новый подход к решению задачи с использованием гибридного эвристического алгоритма, основанного на GRASP и муравьиным алгоритме. По сравнению с оригинальным методом GRASP, предложенный алгоритм демонстрирует достаточно высокую способность к глобальной оптимизации. Вычислительные эксперименты по крупномасштабным задачам показали, что предлагаемая процедура способна генерировать выполнимые пересмотренные расписания менее чем за 5 секунд для расписания, включающего 50 воздушных судов.

В 2013 M. Le, J. Gao, C. Zhan [51] преобразовали задачу управления расписанием движения воздушных судов в комплексную задачу, учитывающую необходимость восстановления маршрутов движения пассажиров и обеспечения потерянных стыковок. Решение основывается на применении генетического алгоритма и моделировании временного окна. В этом же году S. Aktürk, A. Atamtürk, S. Gürel [52] впервые предложили рассмотреть решение задачи управления расписанием с точки зрения поиска компромисса между задержками рейсов и потреблением топлива (и его негативным воздействием на качество воздуха и выбросы парниковых газов), так как зачастую задержки рейса приводят к тому, что на последующих участках производится увеличение скорости полета, когда это возможно, в попытке сдержать задержки, «нагнать время». Для решения было применено коническое квадратичное смешанное целочисленное

программирование, время решения для расписания из 60 воздушных судов составило около 4 минут.

В 2014 Jens O. Brunner [53] представил новую линейную целочисленную модель, которая включает в себя следующие целевые функции: минимизация задержек, затрат на экипажи в случае неоптимального назначения, расходов на обслуживание пассажиров при сбойной ситуации, а также стоимости отмены рейсов с учетом ряда ограничений. Решение с помощью методов целочисленного линейного программирования было получено за несколько секунд.

В 2015 P. Arias и соавт. [54] разработали новую методологию, сочетающую методы оптимизации с методами имитационного моделирования для решения, так называемых, стохастических задач управления графиками движения воздушных судов. Задача решается за счет изменения плана полета с учетом задержек отмен рейсов и переназначения воздушных судов. Основная цель – восстановление исходного плана полетов, минимизируя общую задержку и количество отмененных рейсов.

В этом же году были опубликованы работы H-W. Vos, B. Santos, T. Omondi [55], H. Sousa, R. Teixeira [56], B. Zhu, J.F. Zhu, Q. Gao [57]. В [56] представлен инновационный подход к решению задачи восстановления графика движения воздушных судов, который основан на использовании эффективного алгоритма выбора воздушного судна и модели линейного программирования, представленной в виде параллельной пространственно-временной сети (parallel time-space networks) с целевой функцией минимизации расходов, связанных с задержкой рейсов и пассажиров. Представленная структура моделирования получила название DSS (Disruption Set Solver), идея заключается в том, что решение задачи производится не только с учетом уже реализовавшихся нарушений планового графика, но также учитывает нарушения, которые могут возникнуть вследствие уже реализовавшихся сбоев в расписании. Тестирование предложенного подхода позволило сделать вывод о том, что динамический подход позволяет получить более реалистичные решения, что было подтверждено экспертами предметной области.

В [56] представлен алгоритм, основанный на идее алгоритмов муравьиной колонии. А в [57] рассмотрена комплексная задача восстановления не только графиков движения воздушных судов, но и графиков работы экипажей, движения пассажиров. Решение задачи предлагается производить в два этапа. На первом – восстановление графиков движения воздушных судов путем применения многостадийной задачи целочисленного программирования (multi-stage IP-model). На втором – решение проблем, связанных с нарушением графика работы экипажей и стыковок трансферных пассажиров путем применения эвристического алгоритма и ослабления ограничений.

В 2015-2016 гг. в работах H. Xu, S. Han и соавт. [58, 59] была рассмотрена задача управления расписанием по критерию минимизации расходов, связанных с задержками рейсов и их отменой. Модель аппроксимации временных диапазонов для восстановления расписания в сбойной ситуации заключается в разделении рассматриваемого периода на временные диапазоны и аппроксимации затрат, связанных с задержками рейсов. Приведенная аппроксимационная модель, которая использует оценку фактического полетного времени вместо планового. Основная идея заключается в том, что фактическое полетное время – это время, которое с некоторой вероятностью отличается от планового. Эксперименты показали, что подобная модель предоставляет довольно реалистичные решения и может быть достаточно эффективной.

В 2017 в работе [60] Z. Wu был использован итеративный метод целочисленного программирования с фиксированной точкой для создания возможных маршрутов движения воздушных судов (Distributed fixed-point integer programming). Предложено несколько методов разделения, с помощью которых пространство решений может быть разделено на несколько независимых сегментов.

В этом же году, в работе Y. Hu и соавт. [61] задача управления расписаниями сформулирована как многокритериальная задача целочисленного программирования, в основе которой сетевая модель. Критерии оптимизации

конфликтуют между собой. Первый критерий заключается в минимизации суммарного времени задержки по всем рейсам в расписании, второй критерии минимизирует максимальную задержку рейса, третий – минимизирует количество замен воздушных судов.

Для решения использован эвристический алгоритм, который показывает неплохие практические результаты.

В 2017 году также была опубликовано исследование C. Zhang [62], посвященное проблеме восстановления расписания движения воздушных судов по критерию минимизации расходов, связанных с задержками. Основная идея решения заключается в формировании допустимой цепочки рейсов для каждого воздушного судна. Так как при увеличении количества рейсов в расписании количество цепочек растет экспоненциально, авторы предлагаю двухэтапную эвристику. На первом этапе предлагается определить набор возможных цепочек рейсов. На втором – решить задачи путем ЛП релаксации и минимизировать расходы, связанные с задержками для каждой цепочки. На завершающем шаге выбрать цепочки с минимальными расходами.

В 2018 опубликовано ряд работ. В работе [63] рассмотрена многокритериальная задача управления расписаниями в сбойной ситуации, сформулированная как задача целочисленного линейного программирования. В работе [64] применяется эвристический алгоритм. В [65] рассматривается метод генерации столбцов, а также производится попытка учета дополнительных ограничений, связанных с ограничениями пропускной способности аэропортов, а также необходимостью выполнения технического обслуживания воздушных судов. Тестирование предложенного подхода показало, что решение может быть получено в течение 6 минут для задачи большой размерности (более 600 рейсов и 44 воздушных судов). На основании вычислительных экспериментов показано, что полученные решения позволяют сократить расходы, связанные с задержками рейсов на 20-60%.

В работе [66] также предложен эвристический алгоритм, решающий задачу восстановления расписания в случае возникновения сбойной ситуации, при которой один из аэропортов оперирования становится недоступен для выполнения полетов. Решение основано на применении таблицы весов и пространственно-временной сети.

В работе [67] также рассматривается случай, при котором аэропорт имеет ограничения пропускной способности. Для решения задачи применяется алгоритм последовательного принятия решения, при котором в каждый момент времени выполняется расчёт затрат, связанных с задержкой рейсов, рейсы с наибольшей стоимостью имеют приоритет для постановки в очередь на отправку или прибытия.

В работе [68] авторы отмечают, что параметры модели (время полета, время разворота) являются стохастическими. Как было отмечено ранее, в 2015 году B. Zhu и соавт. Включили некоторые стохастические элементы в модель для работы с различным полетным временем. Однако другие элементы не были учтены ввиду наличия ограничения традиционных подходов к оптимизации. Авторы утверждают, что ввиду высокой сложности работы с подобными моделями одним из лучших вариантов является процесс моделирования, при котором выполняется анализ результатов с высокой и низкой точностью. Моделирование с низкой с точностью дает хорошие результаты с точки зрения времени получения решения. Однако требует проработки вопрос ранжирования решений с целью получения варианта, наиболее приближенного к оптимальному. Тем не менее, авторы отмечают, что возможности симбиотического моделирования имеют большой потенциал в деятельности авиакомпаний, особенно в качестве системы поддержки принятия решений.

В работе [69] D. Wang и соавт. выполнили исследование на примере решения задачи восстановления расписания в случае сбойной ситуации для China Eastern Airlines. Поскольку традиционные способы решения, основанные на использовании методов целочисленного или смешанного целочисленного линейного программирования, не позволяют учесть большое количество

производственных ограничений, авторы предлагают подход к решению, основанный на имитационном моделировании.

В 2019 была опубликована статья с подходом к решению комплексной задачи восстановления расписания и пассажирских маршрутов. Так, в случае сбоя (например, длительной задержки) авиакомпании необходимо отправить воздушное судно и обслужить пассажиров, потерявших стыковки в связи со сбойной ситуацией. При задержке или отмене рейса у пассажиров обычно есть два варианта: переоформиться на другой рейс или получить возврат денег за билет. В своей работе T. Yang и Y. Hu особое внимание уделяет комплексному процессу как восстановления расписания, так и пассажиров с учетом их предпочтений относительно обозначенных выше двух вариантов. В качестве целевой функции предложена минимизация общих расходов, связанных с переназначением ВС по рейсам, и затрат, связанных с изменением маршрутов по пассажирам [21]. Для решения задачи оптимизации авторы представили новый генетический алгоритм.

Один из активных исследователей в области исследования задач управления расписаниями является M. Grönkvist [70, 71]. M. Grönkvist - руководитель отдела оптимизации Jeppesen, входящего в состав компании Boeing. Jeppesen не только предоставляет информационные услуги авиакомпаниям (выпускает сборники аэронавигационной информации и карт для полетов по правилам визуальных полетов, схем захода на посадку и взлета, литературы по авиации, программного обеспечения, учебных пособий и пр.), но и разрабатывает программное обеспечение для планирования полетов и решения операционных задач авиакомпаний. M. Grönkvist начал исследовать задачи управления расписаниями в 2000-х и продолжает по настоящее время. Целевой функцией в его работах традиционно является стоимостная функция минимизации расходов. В части методов решения были попытки применить программирование в ограничениях, метод генерации столбцов, а также рандомизация при программировании в ограничениях.

В заключении отметим одну из последних работ в области исследования задач управления расписаниями, опубликованную в 2020 J. Lee и соавт. В работе [72] авторами выполнена разработка модели динамической системы принятия решений по переназначению воздушных судов, включающая в себя определение времени вылета рейса; определение того, каким воздушным судном будет выполняться рейс, т.е. принятие решения о переназначении; принятие решение об отмене рейсов.

Решения по планированию рейсов также включают в себя определение графика движения ВС, скорость и высоту полета. Вместе эти два набора решений определяют затраты на восстановление расписания (т. е. затраты на задержку, затраты, связанные с заменой ВС, и затраты на отмену) и эксплуатационные расходы на рейс. В отличие от рассмотренных ранее подходов, предложенная система формирует решения в ответ на наблюдаемые события и дает прогнозы будущих сбоев, обеспечивая, таким образом, проактивный подход к управлению расписанием. В частности, сбои, наблюдаемые в любой момент времени, авторы делят на три категории.

1.Распространенные сбои: прошлые сбои, распространяющиеся по цепочке рейсов из-за недостаточных резервов в расписании, которые позволили бы нивелировать задержки на предыдущих участках (за счет применения МТТ).

2.Системные сбои: загруженность хабовых аэропортов, вызванные снижением пропускной способности.

3.Непредвиденные сбои: другие недостатки в работе авиакомпаний и пассажирских перевозок (например, поломки воздушных судов, опоздание экипажей, поздняя посадка пассажиров).

В любой момент принятия решения лицо, непосредственно принимающее решение, наблюдает за всеми сбоями в работе. Однако будущие сбои известны лишь частично и вероятностно. Предложенная система, в основе которой лежит динамическое стохастическое целочисленное программирование, не только фиксирует распространенные сбои, но также учитывает вероятностные прогнозы

системных сбоев и моделирует непредвиденные сбои в каждой точке принятия решений. Время принятия решения в системе, по расчетам, составляем от 1,5 минут.

Сводная информация по ключевым работам в области исследования задач оптимизации расписаний авиакомпаний представлена в таблице 1.6.

Таблица 1.6 – Сводная информация по ключевым исследованиям

Год	**Автор**	**Цели**		**Стратегия, подход к решению**	**Рейсов**	**ВС**	**Время счета, сек**
	T e o d o r o		Минимизация суммарного времени задержек пассажиров	Эвристика, Метод ветвей и границ Решение путем переназначения рейсов и изменение времени вылета			
	Teodorovic D., Stojković G		Минимизация количества отмененных рейсов, а также суммарного времени задержек пассажиров	Эвристика			
	Teodorovic D., Stojković G		Минимизация количества отмененных рейсов, расходов, связанных с задержками, изменений экипажей	Жадный эвристический алгоритм, основанный на принципе FIFO, динамическое программирование			
1993	Z.		Расходы на задержки, отмены и замены	Двойной алгоритм Бузакера-Гоуэна (в основе 2 сетевые модели)			

				Решение путем переназначения рейсов, изменение времени вылета, перегона ВС			
	Cao J.M., Kanafi A.		Выручка за вычетом расходов на задержки, отмены и замены	Квадратичное программирование			
	Yan S., Yang D-H.		Выручка за вычетом расходов	Симплекс-метод, Лагранжева релаксация			
	Yan S., Tu Y.P.		Общая выручка	Симплекс-метод, Лагранжева релаксация			
	K.T.		Расходы, связанные с заменой ВС	Эвристический алгоритм			
	M F. и соавт.		Расходы, связанные с переназначениями и отменами рейсов	Жадный случайный поиск новых маршрутов			
	S G.		Регулярность	ЛП-релаксация (задачи целочисленного программирования)			
	B.G и соавт.	max	Выручка за вычетом расходов	Метод ЛП-релаксации, в случае, когда оптимальное решение не находилось, использовалась эвристика			

	B a r d	min	Задержки и расходы, связанные с отменами	Метод ветвей и границ, ЛП-релаксация			
2003	Rosenberger J.M. и соавт.	min	Расходы, связанные с переназначениями, задержками и отменами рейсов	Эвристический алгоритм, получил название ASR			
2004	s T P.	max	Выручка за вычетом расходов	метод генерации столбцов (в режиме реального времени)			
	Lan S., Clarke J.P., Barnhart C.	min	Задержки	Метод ветвей и границ, Генерации столбцов, ЛП-релаксации			
	Gao Q., Tang X-W., Zhu J.F.	min	Задержки пассажиров и ожидания пассажирами рейса в случае отмены	GRASP, алгоритм имитации отжига			
	N и соавт.	min	Расходы	Динамическое программирование и метод Генерации столбцов			
2010	Liu T-K., Chou J-H., Chen C-H	min	Время разворота, стыковки рейсов, общее время задержки рейса, замена рейсов, максимальное время задержки в 30 минут	Гибридный мультиобъектный генетический алгоритм			
	Y. и соавт.		Задержки и расходы,	Линейное программирование,			

			связанные с отменами рейсов и обслуживанием пассажиров в сбойных ситуациях	Решение выполнялось путем ЛП-релаксации с последующим применением эвристического алгоритма.			
2012	S и соавт.	min	Выручка, нарушение стыковок	Эвристический алгоритм			
	Wu C., Le M.	min	Расходы, задержки, отмены рейсов	Итеративное выращивание деревьев методом комбинации Nde			-
	Zhao X., Guo Y.	min	Расходы на замену ВС и отмены	GRASP в сочетании с муравьиным алгоритмом			
	Le M., Gao J., Zhan C.		Задержки	Моделирование временных окон и Генетический алгоритм	30	6	98
2014	Aktürk S., Atamtürk A., Gürel S.	min	Задержки и расход топлива	Коническое квадратичное смешанное целочисленное программирование	207	60	3600
2014	Brunner J.O.	min	Расходы на экипажи и пассажиров	Линейное целочисленное программирование			
2015	Arias P. и соавт.	min	Задержки и отмены	Программирование в ограничениях с имитацией	-	11	51
2015	Vos H-W., Santos B. и соавт.	min	Расходы	Эвристика выбора самолета с помощью MILP	-	43	900
2015	Sousa H., Teixeira R.	min	Расходы	Динамическое планирование самолетов с оптимизацией алгоритмом	5722	72	32

				муравьиной колонии			
2015	Zhu B., Zhu J.F., Gao Q.	min	Расходы	Алгоритм стохастической жадной имитации отжига	23	6	900
2015-2016	Xu H., Han S., Zhang Y., Li J.	min	Расходы	Аппроксимация временного диапазона с помощью MILP	11	3	1
2017	Wu Z., Li B., Dang Ch.	min	Расходы	Распределенное целочисленное программирование с фиксированной точкой	140	12	7
2017	Hu Y., Liao H. и соавт.	min	Расходы	Эвристический алгоритм поиска окрестностей с ограничениями	410	104	1200
2017	Zhang Ch.	min	Расходы	Двухэтапная эвристика	638	44	150
2018	Khaled O., Minoux M., Mousseau V., Ceugniet X.	min	Расходы	Многокритериальная задача ЛП с ограничениями в области Парето	111	10	30
2018	Šarčević T., Rocha A., Castro A.	min	Расходы	Алгоритм искусственной пчелиной колонии	-	-	-
2018	Liang Zh., Xiao F. и соавт	min	Расходы	Генерация столбцов	638	44	356
2018	Zhao T., Chen Xi.	min	Расходы	Эвристический алгоритм на основе весовой таблицы	32	6	-
2018	Lin H., Wang Zh.	min	Расходы	Алгоритм последовательного принятия решений	749	151	1

2018	Rhodes-Leader L, Onggo B.S., Worthingto D.J., Nelson B.L.	min	Расходы	Симбиотическое моделирование высокой точности с помощью целочисленной программы низкой точности	83	-	-
2019	Wang D. и соавт.	min	Расходы	Подход, основанный на моделировании			
2019	Yang T., Hu Y.	min	Расходы, связанные с задержкой рейсов, пересадкой / отменой рейса для пассажиров, потерявших стыковку	Новый метод, обозначенный как LBMGA, основанный на генетическом алгоритме			
2019	Grönkvist M.	min	Расходы	Программирование в ограничениях, применение рандомизации, метод генерации столбцов	73	780	-
2020	Lee J.	min	Расходы	Структура динамического стохастического целочисленного программирования	852		300

Анализ работ по теме исследования за период с 1984 года позволяет сформулировать некоторые интересные выводы.

Во-первых, интерес к решению проблем управления расписаниями в сбойных ситуациях растет. Только за последние 10 лет было опубликовано более 50 % работ. Это говорит о том, что в настоящее время исследование задач управления назначениями воздушных судов по-прежнему актуальны.

Во-вторых, нужно отметить, что большинство ранних исследований были сосредоточены на отдельных задачах, как правило, направленных на минимизацию расходов в случае отклонения текущего расписания от планового. Более поздние исследования направлены на решение комплексных задач, которые направлены не только на построение оптимального графика для ВС, но и на разрешение проблем с пассажирскими стыковками, доступностью экипажей и пр. Таким образом, наблюдается все больше публикаций, направленных на анализ смежных процессов управления в сбойной ситуации (например, управление расстановкой ВС, планирование экипажей и работа с пассажирами).

В-третьих, все больше внимания уделяется поиску эффективных алгоритмов с точки зрения быстродействия. Несмотря на то, что сложность и размерности задач возрастают, авиакомпании предъявляют высокие требования к быстродействию и качеству вычислений.

В-четвертых, исходя из анализа существующих подходов к решению поставленной задачи, можно сделать вывод о том, что существующие подходы к решению не гарантируют получения оптимального решения даже при обособленном рассмотрении задачи переназначения ВС. Отсутствует алгоритм нахождения оптимальных назначений для каждого ВС с доказанной эффективностью, что говорит об актуальности настоящего исследования. В большинстве случае используются эвристические подходы к решению, а также алгоритмы, сужающие пространство поиска решений.

И, наконец, стоит отметить, что существующие исследования используют типовые критерии для оценки эффективности расписания. В 20% исследований в качестве критерия используется минимизация расходов, в 80% исследований – максимизация выручки. Только в одном исследовании была попытка учесть регулярность полетов в качестве критерия эффективности. Стоит отметить, что исследования в направлении поиска эффективного критерия требуют развития.

1.4. Выводы

В данной главе рассмотрена общая задача управления расписаниями авиакомпании по этапам. Выполнена содержательная постановка рассматриваемой задачи оптимизации *оперативного* управления назначениями ВС на рейсы.

Исходя из анализа литературных источников, можно сделать вывод о том, что на сегодняшний день алгоритмы нахождения цепочек оптимальных назначений для каждого ВС с доказанной эффективностью и должным быстродействием отсутствуют. Ни один из существующих подходов не гарантирует получения оптимального решения, так как применяются либо весьма приближенные алгоритмы, либо эвристики, неконтролируемо сужающие пространство поиска.

Таким образом, для эффективного *оперативного* управления флотом авиакомпании оказывается актуальной задача разработки инструментария синтеза и регулирования назначений и расписаний флота ВС в режиме реального времени. Разработка и применение эффективного метода в системах поддержки принятия решений позволит минимизировать потери авиакомпаний.

Опираясь на выводы первой главы, определим основную цель диссертационной работы следующим образом: разработка эффективного алгоритмического и программного обеспечения решения задачи оптимизации расписаний параллельных систем с заданными задержками начала обслуживания применительно к флоту авиакомпании.

Для достижения данной цели в последующих главах предлагается разработка формальной постановки задачи (Глава 2), исследование и выбор критерия эффективности (Глава 3), исследование и синтез эффективных алгоритмов решения (Глава 4), а также реализация программного решения и предметная интерпретация вычислительных экспериментов на данных действующей авиакомпании (Глава 5).

ГЛАВА 2 РАЗРАБОТКА ФОРМАЛЬНОЙ ПОСТАНОВКИ ЗАДАЧИ СИНТЕЗА И РЕГУЛИРОВАНИЯ РАСПИСАНИЙ

Данная глава посвящена разработке формальной постановки задачи оперативного оптимального управления расписаниями авиакомпаний и переназначений ВС по рейсам. Основной для разработки постановок является рассмотрение системы (флот ВС – рейсы) на начальном этапе в виде параллельной обслуживающей системы. Выполняется критический анализ полученных оригинальных постановок задач, их модификаций с целью последующей разработки и программной реализации эффективных алгоритмов решения.

2.1. Постановка задачи с рекурсиями в условиях

Существует несколько формальных постановок и подходов к решению задачи синтеза оптимального расписания. Основой нижеследующей формализации является оригинальная постановка в виде задачи оптимизации расписаний системы несвязанных параллельных приборов (ВС) с задержками начала обслуживания заявок (рейсов), адаптированная к рассматриваемой задаче оперативного управления расписаниями движением ВС, которая предложена авторами в работах [3, 38, 40, 41, 92, 93], а также предложенная авторами формализация и алгоритм решения близкой задачи с рекурсиями в условиях [73].

В общем виде задача оптимизации расписаний параллельной системы с задержками поступления заявок может быть представлена следующим образом [94]. Пусть имеется I заявок и J несвязанных параллельных прибора. Прерывания обслуживая заявок запрещены. Пусть также известно расписание поступления заявок в параллельную обслуживающую систему и величины задержек поступления заявок (τ_i^0). Задача заключается в определении назначений ($x_{i,j}$) заявок i на приборы j по заданному критерию эффективности при условии различной производительности (длительности обслуживания заявки i прибором j, обозначенной как $t_{i,j}$). Схематично работа параллельной обслуживающей системы представлена на рисунке 2.1.

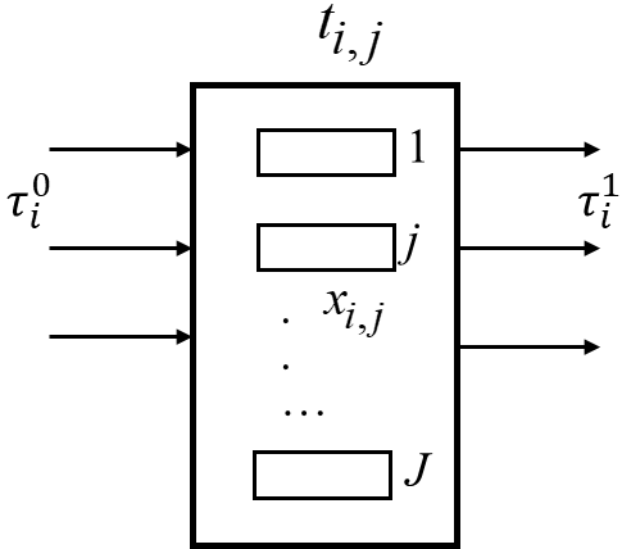


Рисунок 2.1 – Параллельная обслуживающая система с несвязанными приборами

Модель формирования расписания авиакомпании можно описать в виде параллельной обслуживающей системы, в которой заявка – это рейс, а прибор (на который производится назначение) – это воздушное судно. Рассмотрим множество запланированных рейсов авиакомпании, которые необходимо переназначить множеству ВС разных типов при известных (различных) длительностях всех операций таким образом, чтобы минимизировать суммарное отклонение времени выполнения всех рейсов от планового расписания (либо минимизировать максимальное отклонение по всем рейсам от исходных планов). Здесь и далее под длительностью операций понимается сумма времени, необходимого на перелет из пункта отправления в пункт назначения, и время, необходимое на подготовку ВС к рейсу в аэропортах оперирования. На момент корректировки расписания известны текущие задержки начала выполнения рейсов. Величины таких задержек различны для разных ВС и разных рейсов.

Введем условные обозначения:

i - номер рейса $i = \overline{1, I}$;

j - номер ВС $j = \overline{1, J}$;

$\underline{b_j}$ - минимальное количество рейсов, назначаемых ВС;

$\overline{b_j}$ - максимальное количество рейсов, назначаемых ВС *j;*

t_i^0 - расписание вылета i-го рейса, $i=\overline{1,I}$, $T^0=\left\|t_i^0\right\|$;

Здесь и далее, $\|.\|$ обозначает вектор, матрицу или тензор, соответствующей контексту размерности.

t_i^1 - фактическая задержка i-го рейса на момент анализа расписания, $t_i^1 \geq 0$, $i=\overline{1,I}$;

$\tau_i^0 = t_i^0 + t_i^1$ - возможное фактическое время вылета рейса i, в начальный момент времени построения расписания;

$t_{i,j}$ - время обслуживания, подготовки и полета рейса i ВС j, $T=\left\|t_{i,j}\right\|$, $i=\overline{1,I}$, $j=\overline{1,J}$;

$\tau_{i,j}$ - возможное время задержки вылета воздушного судна j рейсом i.

Требуется найти $x_{i,j}$ при условиях:

$$x_{i,j}=\begin{cases}1, & \text{если ВС } j \text{ назначено на рейс } i,\\ 0 & \text{в противном случае,}\end{cases} \quad \text{где } i=\overline{1,I},\ j=\overline{1,J} \tag{2.1}$$

$$\sum_{j=1}^{J} x_{i,j}=1,\ i=\overline{1,I}, \tag{2.2}$$

$$\underline{b}_j \leq \sum_{i=1}^{I} x_{i,j} \leq \bar{b}_j, \quad j=\overline{1,J} \tag{2.3}$$

$$\tau_{i,j}=-\tau_i^0+\sum_{k=1}^{I_k}(\tau_{k,j}+t_{k,j})\cdot x_{k,j},\ i=\overline{I_k+1,I},\ j=\overline{1,J} \tag{2.4}$$

$$\hat{\tau}_{i,j}=\tau_{i,j}+y_{i,j}\geq 0,\ i=\overline{1,I},\ j=\overline{1,J} \tag{2.5}$$

$$y_{i,j}\geq 0,\ i=\overline{1,I},\ j=\overline{1,J} \tag{2.6}$$

$$\sum_{i=1}^{I} \widehat{\tau}_{i,j} \cdot x_{i,j} + \sum_{i=1}^{I} t_{i,j} \cdot x_{i,j} \le \lambda , \quad j = \overline{1,J} \tag{2.7}$$

$$\lambda \to \min . \tag{2.8}$$

Условия (2.1) и (2.2) означают, что на рейс i назначается только одно воздушное судно.

Неравенства (2.3) задают требования к ВС, на которое может быть назначено не менее чем на $\underline{b}_j$ и не более чем на $\bar{b}_j$ рейсов.

Равенства (2.4) означают, что задержка вылета ВС j на текущем рейсе i является рекурсивной функцией задержек предыдущих рейсов этого воздушного судна.

Условия (2.5) и (2.6) компенсируют возможные отрицательные значения задержки $\tau_{i,j}$. Содержательно отрицательная задержка означает вылет рейса из аэропорта оперирования раньше времени, запланированного в расписании. Ввиду того, что на практике вылет ВС ранее запланированного времени не допускается, производится нейтрализация отрицательных задержек за счет переменных-компенсаторов $y_{i,j} \ge 0$. Тогда $\widehat{\tau}_{i,j} \ge 0$ зависимые переменные задачи, имеющие смысл скорректированных задержек между прибытием ВС j и его вылетом рейсом i, с учетом необходимого времени обслуживания на земле.

Условия (2.7) и (2.8) определяют минимаксный критерий быстродействия. Его использование способствует организации равномерной загрузки парка воздушных судов, минимизируя максимальный суммарный простой любого воздушного судна из всего множества ВС флота авиакомпании.

Выражения (2.4), опосредующие ограничения (2.5) и (2.7) содержат рекурсии, поскольку всякие последующие (по времени) значения $\tau_{i,j}$ и $\widehat{\tau}_{i,j}$ зависят от предыдущих.

Расписание, синтезируемое посредством решения задачи (2.1)-(2.8) полностью определяется оптимальными назначениями $x^*_{i,j}, i=\overline{1,I}, j=\overline{1,J}$ и фактическими задержками при таких назначениях $\hat{\tau}^*_{i,j}, i=\overline{1,I}, j=\overline{1,J}$. Расписание на выходе системы: $\tau^1_i=\sum_{j=1}^{\bar{J}}(\hat{\tau}_{i,j}+t_{i,j})x_{i,j}$, $i=\overline{1,I}$. То же при оптимальном решении: $\tau^{1*}_i=\sum_{j=1}^{\bar{J}}(\hat{\tau}^*_{i,j}+t_{i,j})x^*_{i,j}$, $i=\overline{1,I}$.

2.2. Подходы к решению задачи с рекурсиями в условиях и её редукция в milp

Приведем выражения (2.1)-(2.8) к нормальному виду задачи математического программирования. Условия (2.4)-(2.7) содержат рекурсивные функции фактических задержек $\tau_{i,j}$. В явном виде $\tau_{i,j}$, как рекурсивные функции относительно переменных $x_{i,j}$, определяет (2.4). Не раскрыв рекурсии, задачу (2.1)-(2.8) точно решить невозможно. Исключением является метод динамического программирования. Однако, использование динамического программирования «в чистом виде» в данном случае приводит к полному перебору допустимых вариантов с ростом числа таких вариантов в геометрической прогрессии от размерности и для реальных размерностей невозможно фактически.

Приведем подробности раскрытия рекурсий.

Непосредственно из (2.5) следует:

$$\begin{aligned}
&\tau_{1,1}=\tau^0_{1,1},\\
&\tau_{2,1}=\tau^0_{2,1}-\tau_{1,1}x_{1,1}-t_{1,1}=\tau^0_{2,1}-\tau^0_{1,1}x_{1,1}-t_{1,1}x_{1,1}=\tau^0_{2,1}-(\tau^0_{1,1}+t_{1,1})x_{1,1},\\
&\tau_{3,1}=\tau^0_{3,1}-[(\tau_{1,1}x_{1,1}+\tau_{2,1}x_{2,1})+(t_{1,1}x_{1,1}+t_{2,1}x_{2,1})].
\end{aligned} \tag{2.9}$$

Преобразования, аналогичные (2.12)-(2.14), приводят к результату:

$$\tau_{3,1} = \tau^0_{3,1} - [(\tau^0_{1,1} + t_{1,1})(x_{1,1} - x_{1,1}x_{2,1}) + (\tau^0_{2,1} + t_{2,1})x_{2,1})].$$

Далее получаем:

$$\tau_{4,1} = \tau^0_{4,1} - [(\tau_{1,1} + t_{1,1})x_{1,1} + (\tau_{2,1} + t_{2,1})x_{2,1} + (\tau_{3,1} + t_{3,1})x_{3,1}], \text{ или}$$

$$\tau_{4,1} = \tau^0_{4,1} - [(\tau^0_{1,1} + t_{1,1})(x_{1,1} - x_{1,1}x_{2,1} - x_{1,1}x_{3,1} + x_{1,1}x_{2,1}x_{3,1}) +$$

$$+(\tau^0_{2,1} + t_{2,1})(x_{2,1} - x_{2,1}x_{3,1}) + (\tau^0_{3,1} + t_{3,1})x_{3,1}]. \qquad (2.10)$$

Воспользуемся соотношением: $f_1(x_{1,1}, x_{2,1}) = x_{1,1}(1 - x_{2,1}) = x_{1,1}\bar{x}_{2,1}$, и рассмотрим выражение $x_{1,1} - x_{1,1}x_{2,1} - x_{1,1}x_{3,1} + x_{1,1}x_{2,1}x_{3,1}$ в (2.17), как булеву функцию

$$f_2(x_{1,1}, x_{2,1}, x_{3,1}) = x_{1,1}(1 - x_{2,1} - x_{2,1}x_{3,1}) = x_{1,1}(\bar{x}_{2,1} - x_{3,1}(-x_{2,1} + 1)) =$$

$$= x_{1,1}(\bar{x}_{2,1} - x_{3,1}(-x_{2,1} + 1)) = x_{1,1}\bar{x}_{2,1}\bar{x}_{3,1}.$$

Здесь и далее $\bar{x}_{i,j}$ отрицание $x_{i,j}$.

Легко убедиться в истинности $f_2(x_{1,1}, x_{2,1}, x_{3,1}) = x_{1,1}\bar{x}_{2,1}\bar{x}_{3,1}$, из чего следует:

$$\tau_{4,1} = \tau^0_{4,1} - [(\tau^0_{1,1} + t_{1,1})(x_{1,1}\bar{x}_{2,1}\bar{x}_{3,1}) + (\tau^0_{2,1} + t_{2,1})(x_{2,1}\bar{x}_{3,1}) + (\tau^0_{3,1} + t_{3,1})x_{3,1}],$$

. . .

Индукцией по размерности получаем:

$$\tau_{i,j} = \tau^0_{i,j} - [(\tau^0_{1,j} + t_{1,j})\,(x_{1,j}\bigcap_{l=2}^{i-1}\bar{x}_{l,j}) + (\tau^0_{2,j} + t_{2,j})\,(x_{2,j}\bigcap_{l=3}^{i-1}\bar{x}_{l,j}) + \ldots +$$

$$+(\tau^0_{i-1,j} + t_{i-1,j})x_{i-1,j}],\ i = \overline{1,I},\ j = \overline{1,J}, \text{ или}$$

$$\tau_{i,j} = \tau^0_{i,j} - \sum_{k=1}^{i-1}(\tau^0_{k,j} + t_{k,j})\,x_{k,j}\bigcap_{l=k+1}^{i-1}\bar{x}_{l,j}, i = \overline{1,I},\ j = \overline{1,J}. \qquad (2.11)$$

Уравнения (2.11) записаны в смешанной форме с компонентами булевых функций $x_{k,j}\bigcap_{l=k+1}^{i-1}\bar{x}_{l,j}$. Эти условия порождают существенно нелинейные ограничения. И таким образом, при раскрытии рекурсий получается задача (2.1)-

(2.3), (2.11), (2.5)-(2.8), неразрешимая точными методами (с учетом потенциальных размерностей).

В работах [38, 40] близкую задачу удалось редуцировать в задачу частично-целочисленного линейного программирования (milp).

Опишем процедуру применительно к исследуемой задаче. Приведем ограничения, содержащие конъюнкции $x_{k,j}\bigcap_{l=k+1}^{i-1}\bar{x}_{l,j}$, к линейному виду посредством использования вспомогательных переменных, а также системы дополнительных неравенств [38].

Обозначим $f_{k,i,j}(x_{k,j},x_{k+1,j},...,x_{i-1,j})=x_{k,j}\bigcap_{l=k+1}^{i-1}\bar{x}_{l,j}$, $i=\overline{1,I}$, $j=\overline{1,J}$ и введем в рассмотрение переменные $u_{k,i,j}$, значения которых истинны только тогда, когда истинны значения $f_{k,i,j}$.

Примеры таких пар: $u_{1,2,1}$ и $f_{1,2,1}(x_{1,1},x_{2,1})=x_{1,1}\bar{x}_{2,1}$, $u_{2,3,1}$ и $f_{2,3,1}(x_{2,1},x_{3,1})=x_{2,1}\bar{x}_{3,1}$, $u_{1,3,1}$ и $f_{1,3,1}(x_{1,1},x_{2,1},x_{3,1})=x_{1,1}\bar{x}_{2,1}\bar{x}_{3,1}$.

Соответствующие условия истинности можно задать посредством системы ограничений:

$$0\le x_{1,1}+\bar{x}_{2,1}-2u_{1,2,1}\le 1,$$

$$0\le x_{2,1}+\bar{x}_{3,1}-2u_{2,3,1}\le 1,$$

$$0\le x_{1,1}+\bar{x}_{2,1}+\bar{x}_{3,1}-3u_{1,3,1}\le 2,$$

. . .

$$0\le x_{k,j}+\sum_{l=k+1}^{i-1}\bar{x}_{l,j}-Ku_{k,i,j}\le K-1,\ \text{где}\ K=i-k.$$

Возвращаясь к исходным обозначениям ($\bar{x}_{i,j}=1-x_{i,j}$), получим:

$$-1 \le x_{1,1} - x_{2,1} - 2\,u_{1,2,1} \le 0,$$

$$-1 \le x_{2,1} - x_{3,1} - 2\,u_{2,3,1} \le 0,$$

$$-2 \le x_{1,1} - x_{2,1} - x_{3,1} - 3\,u_{1,3,1} \le 0,$$

. . .

$$-K+1 \le x_{k,j} - \sum_{l=k+1}^{i-1} x_{l,j} - K u_{k,i,j} \le 0. \qquad (2.12)$$

Условия (2.11) при этом запишутся следующим образом:

$$\tau_{i,j} = \tau_{i,j}^0 - \sum_{k=1}^{i-1} (\tau_{k,j}^0 + t_{k,j})\, u_{k,i,j}, i = \overline{1,\bar{I}},\ j = \overline{1,\bar{J}}.$$

Тогда

$$\hat{\tau}_{i,j} = y_{i,j} + \tau_{i,j}^0 - \sum_{k=1}^{i-1} (\tau_{k,j}^0 + t_{k,j})\, u_{k,i,j} \ge 0, \text{ или}$$

$$-y_{i,j} + \sum_{k=1}^{i-1} (\tau_{k,j}^0 + t_{k,j}) u_{k,i,j} \le \tau_{i,j}^0,\ i = \overline{1,\bar{I}},\ j = \overline{1,\bar{J}}, \qquad (2.13)$$

Кроме этого:

$$\sum_{i=1}^{\bar{I}} \tau_{i,j} x_{i,j} = \sum_{i=1}^{\bar{I}} [\tau_{i,j}^0 - \sum_{k=1}^{i-1} (\tau_{k,j}^0 + t_{k,j})\, x_{k,j} \bigcap_{l=k+1}^{i-1} \bar{x}_{l,j}\ ] x_{i,j} =$$

$$= \sum_{i=1}^{\bar{I}} \tau_{i,j}^0 x_{i,j} - \sum_{i=1}^{\bar{I}} x_{i,j} \sum_{k=1}^{i-1} (\tau_{k,j}^0 + t_{k,j})\, x_{k,j} \bigcap_{l=k+1}^{i-1} \bar{x}_{l,j} =$$

$$= \sum_{i=1}^{\bar{I}} \tau_{i,j}^0 x_{i,j} - \sum_{i=1}^{\bar{I}} \sum_{k=1}^{i-1} (\tau_{k,j}^0 + t_{k,j})\, x_{i,j} x_{k,j} \bigcap_{l=k+1}^{i-1} \bar{x}_{l,j}.$$

Для того, чтобы привести (2.6) к линейному виду, расширим и дополним, введенные выше булевы функции $f_{k,i,j}$ и соответствующие им переменные $u_{k,i,j}$.

$$f_{k,i,j}(x_{k,j}, x_{k+1,j}, ..., x_{i,j}) = x_{i,j} x_{k,j} \bigcap_{l=k+1}^{i-1} \bar{x}_{l,j},\ i = \overline{1,\bar{I}},\ j = \overline{1,\bar{J}},\ k < i. \qquad (2.14)$$

Тогда (2.24) опосредуют следующие условия:

$$-K+2 \le x_{i,j} + x_{k,j} - \sum_{l=k+1}^{i-1} x_{l,j} - K u_{k,i,j} \le 1,\ i = \overline{1,\bar{I}},\ j = \overline{1,\bar{J}},\ k = \overline{1,\bar{I}},$$

где $K = i - k + 1.$ (2.15)

И неравенства (2.6) примут вид:

$$\sum_{i=1}^{\bar{I}} (\tau_{i,j}^0 + t_{i,j}) x_{i,j} + \sum_{i=1}^{\bar{I}} y_{i,j} - \sum_{i=1}^{\bar{I}} \sum_{k=1}^{i-1} (\tau_{k,j}^0 + t_{k,j})\, u_{k,i,j} \le \lambda,\ j = \overline{1,\bar{J}}.$$ (2.16)

Учтем также вспомогательные переменные:

$$u_{k,i,j} = \begin{cases} 1, \text{ при истинности выражения } x_{i,j} x_{k,j} \bigcap_{l=k+1}^{i-1} \bar{x}_{l,j}, \\ 0 \quad \text{в противном случае,} \end{cases} i,k = \overline{1,\bar{I}},$$

$k < i,\ j = \overline{1,\bar{J}}.$ (2.17)

В конечном итоге постановка (2.1)–(2.8) взаимно однозначно соответствует следующая задача частично–целочисленного программирования с булевыми переменными:

$$\sum_{i=1}^{I} x_{ij} = 1, \quad \forall j = \overline{1,J};$$ (2.18)

$$\underline{b_i} \le \sum_{j=1}^{J} x_{ij} \le \overline{b_i}, \quad \forall i = \overline{1,I};$$ (2.19)

$$x_{ij} = \begin{cases} 1, \text{ если ВС } j \text{ назначено на рейс } i; \\ 0 \quad \text{в противном случае;} \end{cases}$$ (2.20)

$$u_{ijk} = \begin{cases} 1, \textit{ при истинности выражения} \\ \qquad f_{ijk}(x_{ik}, x_{ik+1}, \ldots, x_{ij-1}) = x_{ij} x_{ik} \bigcap_{l=k+1}^{j-1} \bar{x}_{il}, \quad i = \overline{1,I},\ j = \overline{1,J}; \\ 0 \quad \textit{в противном случае;} \end{cases}$$

где $\bar{x}_{il} = 1 - x_{il};$

(2.21)

$$-K+2 \le x_{ij} + x_{ik} - \sum_{l=k+1}^{j-1} x_{il} - K u_{ijk} \le 1, \forall i = \overline{1,I},\ j = \overline{1,J},\ k = \overline{1,J},$$

$$K = j - k + 1; \tag{2.22}$$

$$\widehat{\tau}_{ij} = y_{ij} + \tau_j^0 - \sum_{k=1}^{j-1} (\tau_k^0 + t_{ik})\, u_{ijk} \ge 0, \text{ или}$$

$$-y_{ij} + \sum_{k=1}^{j-1} (\tau_k^0 + t_{ik}) u_{ijk} \le \tau_j^0,\ i = \overline{1,I},\ j = \overline{1,J}; \tag{2.23}$$

$$\sum_{j=1}^{J} (\tau_j^0 + t_{ij}) x_{ij} + \sum_{j=1}^{J} y_{ij} - \sum_{j=1}^{J} \sum_{k=1}^{j-1} (\tau_k^0 + t_{ik})\, u_{ijk} \le \lambda, \forall i = \overline{1,I}; \tag{2.24}$$

$$y_{ij} \ge 0\ \forall i = \overline{1,I},\ j = \overline{1,J}; \tag{2.25}$$

$$Z_1 = \lambda \to \min. \tag{2.26}$$

Определим ее размерность.

Количество булевых переменных-назначений x_{ij}: $n_x = I \cdot J$.

Количество непрерывных переменных компенсации отрицательных задержек y_{ij}: $n_y = (J-1) \cdot I$.

Количество логических переменных u_{ijk}: $n_u = \frac{J \cdot (J-1)}{2} \cdot I$.

Количество ограничений: назначения составит $n_1 = I + J$; логических ограничений $n_2 = I$; и $n_3 = (J-1) \cdot I$; а также, $n_4 = \frac{J \cdot (J-1)}{2} \cdot I$.

Таким образом, размерность релаксированной задачи календарного планирования не позволяет непосредственно, используя стандартные алгоритмы целочисленной оптимизации, находить оптимальные расписания работы систем реальной размерности. В Приложении А приведена модель, исходные данные и результаты счета реализации релаксированной задачи.

Результаты счета являются экспериментальным доказательством неэффективности использования прямой редукции. Так, например, для нахождения даже приближенного решения с не превышающим шести процентов отклонением от оптимума реализации задачи для 20 рейсов и 5 ВС, потребовалось более 16 часов

времени счета 6-ядерного процессора с использованием последней версии IBM ILOG CPLEX optimization studio.

Таким образом, прямое раскрытие рекурсий, как правило, приводит к значительному росту (иногда на много порядков) размерностей за счет появления дополнительных логических переменных и ограничений, что недопустимо для задачи оперативного управления с учетом принадлежности к классу NP.

Известны 2 пути применения постановки (2.1)–(2.8) и ее редукции. Первая основана на формировании упрощенной (релаксированной) задачи с двумя критериями (именуемой бикритериальной релаксацией) позволяющей находить приближенные к оптимальным по быстродействию расписаниям. Подробности можно найти в [38, 39]. Приведем ее краткую запись.

$$\sum_{i=1}^{I} x_{ij} = 1, \quad \forall j = \overline{1, J}; \tag{2.27}$$

$$\underline{b_i} \le \sum_{j=1}^{J} x_{ij} \le \overline{b_i}, \quad \forall i = \overline{1, I}; \tag{2.28}$$

$$x_{ij} = \begin{cases} 1, & \text{если ВС } j \text{ назначается на рейс } i; \\ 0 & \text{в противном случае;} \end{cases} \tag{2.29}$$

$$\sum_{j=1}^{J} \tau^0{}_j x_{ij} \le \beta, \quad \forall i = \overline{1, I}; \tag{2.30}$$

$$\sum_{j=1}^{J} t_{ij} x_{ij} \le \lambda, \quad \forall i = \overline{1, I}; \tag{2.31}$$

$$Z_2 = \lambda \to \min; \tag{2.32}$$

$$Z_3 = \beta \to \min. \tag{2.33}$$

Из контекста понятно, что данный подход к распределению ВС на рейсы предполагает компромиссное решение по «чистому» быстродействию системы без учета задержек (λ) и по равномерности распределения задержек между рейсами (β).

Второй путь – применение идеологии динамического программирования непосредственно к задаче (2.1)–(2.8) с поэтапным отсевом локально наихудших вариантов [41].

2.3. Постановка задачи с дизъюнкциями в ограничениях

Как отмченено ранее, исходная постановка задачи с рекурсиями в ограничениях оказалась неэффективной для разработки алгоритмов поиска приемлемых по точности приближений к оптимумам, что доказано экспериментально применительно к динамическому программированию [74]. В данном разделе описан альтернативный новый подход к решению на основе формализации задачи с дизъюнкциями в ограничениях, позволяющий получить существенные преимущества в быстродействии и точности (близости к оптимумам получаемых расписаний) в сравнении с другими существующим подходами.

Обозначим как $\tau^0_{i,j}$ задержку начала выполнения i-го рейса воздушным судном j.

Упорядочим рейсы для каждого ВС авиакомпании по возрастанию $\tau^0_{i,j}$. Тогда $\mathrm{T}^0 = \left\| \tau^0_{i,j} \right\|$, $j = \overline{1,J}, i = \overline{1,I}$ можно интерпретировать как расписание на входе каждого из ВС, где I - общее число рейсов, J - число ВС.

Через $x_{i,j}$ обозначим булевы переменные-назначения рейса i ВС j, подлежащие определению.

Введем непрерывные переменные $C_{i,j} \geq \tau^0_{i,j}$ - время вылета рейса i при назначении на ВС j (очевидно $C_i = \sum_{j=1}^{J} C_{i,j}$).

Определим номер фиктивного завершающего рейса $\bar{i}$ - для каждого ВС *j*, $j=\overline{1,J}$. А также булевы переменные $w_{i,k,j}$, с истинностью условия следования рейса *k* непосредственно за рейсом *i* для ВС *j*.

Формальная постановка задачи может быть записана следующим образом.

$$\sum_{j=1}^{J} x_{i,j}=1,\ \ i=\overline{1,I} \tag{2.34}$$

$$\underline{b_j} \le \sum_{i=1}^{I} x_{i,j} \le \overline{b_j},\ \ j=\overline{1,J} \tag{2.35}$$

$$x_{i,j}=\begin{cases}1, & \text{если рейс } i \text{ назначен ВС } j,\\ 0 & \text{в противном случае,}\end{cases} \tag{2.36}$$

$$C_{i,j}-C_{k,j}+t_{i,j}x_{i,j}-Mw_{i,k,j}\le 0,\ \ \text{где } C_{i,j}\ge \tau^{0}_{i,j} \tag{2.37}$$

$$C_{k,j}-C_{i,j}+t_{k,j}x_{k,j}+Mw_{i,k,j}\le M,\ \ \text{где } i\ne k, i,k=\overline{1,I},\ j=\overline{1,J} \tag{2.38}$$

$$w_{i,k,j}=\begin{cases}1, & \text{если рейс } i \text{ предшествует } k \text{ для борта } j,\\ 0 & \text{в противном случае}\end{cases} \tag{2.39}$$

$C_{\bar{i},j}+t_{\bar{i},j}x_{\bar{i},j}\le C_{\max}$, $j=\overline{1,J}$,где $\bar{i}$ - номер завершающего рейса для борта *j*, (2.40)

$$C_{\max}\to \min. \tag{2.41}$$

Условия (2.34) - (2.36) характерны для задачи о назначениях. Ограничения (2.34) обеспечивают назначения любого рейса единственному ВС. Условия (2.35) - назначения не менее $\underline{b_j}$ и не более $\overline{b_j}$ рейсов любому ВС *j*.

Ограничения (2.37) - (2.39) обусловливают выбор кратчайших путей посредством определения $w_{i,k,j}$ (с учетом назначений $x_{k,j}$) на полных графов связей между рейсами для каждого ВС.

Условия (2.37), (2.38) определяют, что рейсы *i* и *k* выполняются неодновременно бортом *j*. С их помощью реализуются логические операции «или»

при выборе вариантов последовательностей выполнения рейсов. В ограничениях (2.37), (2.38) M - большое положительное число.

Условие (2.41) определяет критерий эффективности расписания, известный как критерий быстродействия системы параллельных несвязанных приборов с общеупотребимым обозначением $C_{\max}$.

Расписание, синтезируемое посредством решения задачи (2.34)-(2.41) полностью определяется оптимальными назначениями $x^*_{i,j}, i=\overline{1,I}, j=\overline{1,J}$ и фактическими значениями времени вылета каждого рейса при таких назначениях, вычисляемыми как $C^*_{i,j}x^*_{i,j}$, где $C^*_{i,j}$ значения $C_{i,j}$ в оптимальном решении.

Определим класс и актуальные для практики размерности задачи оперативного регулирования графиков вылетов флота авиакомпании.

Класс задачи определить несложно, поскольку даже при существенном упрощении путем исключения условий (2.37)-(2.39) и, соответственно, переменных $w_{i,k,j}$ получается задача оптимизации расписаний несвязанных параллельных машин по критерию $C_{\max}$. Такая упрощенная задача, тем не менее, является NP-трудной, что показано, например, в работах [83, 86, 91].

В задаче (2.34)-(2.41) содержится $I \cdot J$ булевых переменных $x_{i,j}$ и $I \cdot J$ непрерывных переменных $C_{i,j}$. Условия (2.37), (2.38) и (2.39) добавляют $I \cdot J \cdot \binom{2}{I} = J \cdot I^2 \cdot (I-1)/2$ переменных $w_{i,k,j}$ и $2 \cdot I \cdot J \cdot \binom{2}{I} = J \cdot I^2 \cdot (I-1)$ ограничений. Приведенные оценки числа переменных и ограничений формальной постановки с дизъюнкциями в ограничениях показывают, что даже без учета сложности условий (2.37)-(2.39), они многократно утяжеляет исходно труднорешаемую задачу (2.34)-(2.36), (2.40)-(2.41).

Относительно актуальных размерностей реализаций задачи оптимального оперативного регулирования расписания приведем следующие оценки. Для

среднего размера авиакомпании актуальные размерности реализаций рассматриваемой задачи лежат в интервалах: J [10÷30], I [100÷300]. Откуда оценка числа булевых переменных $J \cdot I^2 \cdot (I-1)/2 + J \cdot I = 30 \cdot 300^2 \cdot 299/2 + 30 \cdot 300 = 403.659.000$, т.е. более 400 миллионов.

По этой причине использование постановки (2.34)-(2.41) для построения расписаний даже после элиминации части переменных с применением точных алгоритмов весьма затруднительно, а для задач реальной размерности невозможно из-за фактически бесконечного времени счета, что подтверждается вычислительными экспериментами с IBM CPLEX.

2.4. Релаксация постановки с дизъюнкциями в ограничениях путем априорного назначения последовательности рейсов

Как отмечено ранее, применение одной из предложенных постановок, с рекурсиями в условиях или с дизъюнкциями в ограничениях, весьма затруднительно для решения задачи оперативного регулирования графиков движения ВС. Выполним релаксацию постановки задачи оптимального регулирования расписаний флота авиакомпании с дизъюнкциями в ограничениях путем априорного назначения последовательности выполнения рейсов.

Для каждого ВС упорядочиваем рейсы по возрастанию заданных задержек $\tau^0_{i,j}$. Всякую такую последовательность считаем последовательностью возможного выполнения рейсов каждым воздушным судном.

Вместо (2.37)-(2.39) применим условия:

$C_{i,j} \geq \tau^0_{i,j} x_{i,j}$, где $C_{i,j}$ - время вылета рейса i ВС j, $i = \overline{1,I}$, $j = \overline{1,J}$; (2.42)

$C_{i,j} + t_{i,j} x_{i,j} \leq C_{k,j}$, если рейс i непосредственно предшествует k. (2.43)

Сформулированную задачу обозначим как (2.34)-(2.36), (2.40)-(2.43) [96].

Докажем эквивалентность постановок задачи (2.34)-(2.41) и (2.24)-(2.36), (2.40)-(2.43) при условии упорядочения рейсов для всех бортов по возрастанию исходных задержек $\tau_{i,j}^0$.

1. Рассмотрим случай $\tau_{i,j}^0 = 0,\ \ i = \overline{1,I}\,,\ j = \overline{1,J}\,.$ При таких условиях оптимальное решение $C_{\max} = \lambda,\ \ x_{i,j}^*,\ \ i = \overline{1,I}\,,\ j = \overline{1,J}$ не зависит от порядка следования рейсов, поскольку ограничения (2.43) и (2.40) для $x_{i,j}^*$ справедливы при любом порядке следования, а (2.42) означают $C_{i,j} \ge 0$.

Действительно $C_{i,j} + t_{i,j} x_{i,j}^* = C_{k,j}$,.если рейс i непосредственно предшествует *k*. И тогда время завершения последнего рейса любого ВС $j = \overline{1,J}$ составит величину $C_j = \sum_{i=1}^{I} t_{i,j} x_{i,j}^* \le C_{\max}$. Это непосредственно означает, что при $\tau_{i,j}^0 = 0,\ \ i = \overline{1,I}\,,\ j = \overline{1,J}$ решения задач (2.34)-(2.36), (2.37)-(2.41) и (2.34)-(2.36), (2.40)-(2.43) совпадают.

2. Рассмотрим систему из одного ВС и упорядочим список рейсов по возрастанию величин $\tau_i^0 > 0,\ \ i = \overline{1,I}\,.$ $(0 \le \tau_1^0 \le \tau_2^0 \le ... \le \tau_i^0 \le \tau_k^0 ... \le \tau_I^0)$ Рассмотрим последовательно координаты решения $C_1 = \tau_1^0,\qquad C_2 = \tau_1^0 + t_1 x_1,$ $C_3 = \max\left\{\tau_{2,}^0 C_2\right\} + t_2 x_2,\ C_{i+1} = \max\left\{\tau_{i,}^0 C_i\right\} + t_i x_i,\ i = \overline{1,I}\,.$ В общем случае, если рейс *i* непосредственно предшествует *k*. то $\max\left\{\tau_{i,}^0 C_i\right\} + t_i x_i = C_k$. Очевидно, что для завершающего рейса I $C_I = C_{\max}$.

Изменим порядок следования рейсов *i u k* на обратный (*k* непосредственно предшествует *i*). Не теряя общности, будем полагать выполнение условий

$\tau_k^0 \geq \tau_i^0 > C_i$. Тогда $C_i^{'} = \tau_k^0 + t_k x_k = C_k$ и $C_{k+1}^{'} = C_i^{'} + t_i x_i$, откуда непосредственно следует $C_{\max}^{'} = C_{\max} + t_i x_i$.

Если же $C_i > \tau_k^0 \geq \tau_i^0$, то значения τ_i^0 и τ_k^0 условно можно считать равными нулю. Как показано выше, в этом случае изменение порядка следования рейсов (2.43) не приводит к изменению значения $C_{\max}$ ($C_{\max}^{'} = C_{\max}$).

3. Таким образом, любая замена порядка следования рейсов ВС отличным от заданного последовательностью $0 \leq \tau_1^0 \leq \tau_2^0 \leq ... \leq \tau_i^0 \leq \tau_k^0 ... \leq \tau_I^0$, может привести только к увеличению общего времени обслуживания $C_{\max}$.

4. Распространим результат на общий случай $\tau_{i,j}^0 \geq 0$, $i = \overline{1,I}$, $j = \overline{1,J}$. Упорядочим задержки для каждого ВС $0 \leq ... \leq \tau_{i,j}^0 \leq \tau_{k,j}^0 \leq ... \leq \tau_{I,j}^0$ $j = \overline{1,J}$. Тогда для любого ВС замена порядка следования рейсов отличным от заданного, как это следует из п. 2, может привести только к увеличению общей длины расписания $C_{\max}$. Откуда непосредственно следует эквивалентность постановок (2.34)-(2.36), (2.37)-(2.41) и (2.34)-(2.36), (2.40)-(2.43).

В заключении раздела отмечаем снижение на много порядков размерностей задачи в актуальных приложениях.

Для сравнения оценим число булевых переменных в реализации задачи для J [10÷30], I [100÷300]. $I \cdot J = 300 \cdot 30 = 9000$.

В приложении Б приведены реализация модели (2.34)-(2.36), (2.40)-(2.43) средствами OPL CPLEX и пример файла исходных данных. Результаты вычислительных экспериментов с реализацией модели представлены в главе 4.

2.5. Выводы

В данной главе предложены оригинальные формальные постановки задачи оперативного оптимального управления расписаниями авиакомпаний и модификации этих постановок.

Первоначальная формальная постановка задачи заключалась в описании поставленной задачи с применением рекурсий, поскольку задержка каждого последующего рейса зависит от задержек предыдущих рейсов. Данная формализация оказалась неэффективной для разработки алгоритмов приближенного решения, что доказано экспериментально применительно к динамическому программированию [74]

Описан альтернативный оригинальный подход на основе формализации задачи с дизъюнкциями в ограничениях, который позволил уйти от рекурсий, но привел к значительному увеличению размерности. По этой причине использование постановки с дизъюнкциями в ограничениях в чистом виде для построения расписаний, даже после элиминации части переменных, с применением точных алгоритмов весьма затруднительно. С целью сокращения размерности представлена релаксация задачи с дизъюнкциями в ограничениях путем априорного назначения последовательности выполнения рейсов.

ГЛАВА 3 ВЫБОР КРИТЕРИЕВ ЭФФЕКТИВНОСТИ РАСПИСАНИЯ АВИАКОМПАНИИ

Задача выбора критерия оптимизации требует отдельного внимания, поскольку правильный выбор играет существенную роль с точки зрения оценки эффективности принятия решений. Однако, в теории принятия решений не найдено общего метода выбора критериев эффективности. В данной главе выполнен анализ типовых критериев оценки качества расписания, а также предложен оригинальный критерий минимизации риска нарушения пунктуальности рейсов при поиске оптимального решения дискретной задачи управления назначениями ВС, эффективность применения которого подтверждается вычислительными экспериментами, представленными в главе 4.

3.1. Пунктуальность как показатель эффективности транспортной системы

Как правило, при решении задач оперативного регулирования текущих графиков вылета воздушных судов используются классические подходы к управлению: минимизации количества отмененных рейсов, минимизации количества задержек или суммарного времени задержек, минимизация стоимости перевозки и так далее (таблица 1.6). В сформулированной задаче (2.34)-(2.36), (2.40)-(2.43) в качестве целевого критерия предложена минимизация времени вылета завершающего рейса в расписании. Однако важно понимать, что минимизация времени вылета завершающего рейса не всегда оправдана, поскольку не гарантирует минимального времени задержек по всему расписанию. Более того, в синтезированном расписании может оказаться рейс с длительной задержкой, который приведет к снижению лояльности пассажиров и увеличению риска задержки последующих рейсов при реализации дополнительных факторов (например, ухудшение погодных условий и пр.).

Определение целевого критерия, в первую очередь, зависит от ожиданий бизнеса и от целевых параметров. Так, одним из важнейших показателей для любой авиакомпании является пунктуальность. Пунктуальность (регулярность) – один из

важнейших показателей эффективности транспортной системы, мера способности системы выполнить работу вовремя, которая характеризует работу авиапредприятий. Наилучшим образом транспортная система функционирует именно тогда, когда службы работают точно по графику, так как ЛПР, планирующее использование службы, может согласовать свою деятельность с деятельностью транспортной системы.

В СССР правила учета пунктуальности были определены Руководством по обеспечению и учету регулярности полетов воздушных судов гражданской авиации СССР (РРП ГА-90) и утверждены приказом МГА СССР от 10.01.1990 N 6 "Об утверждении и введении в действие Руководства по обеспечению и учету регулярности полетов воздушных судов гражданской авиации". Данный документ утратил силу на территории Российской Федерации в связи с изданием Приказа Минтранса России от 04.12.2020 N 541. В настоящий момент выполняется разработка новой методики учета задержек и пунктуальности авиарейсов. Несмотря на отсутствие унифицированного подхода, абсолютно все авиакомпании и аэропорты, являясь частью транспортной системы, внимательно следят за своевременным выполнением рейсов, разрабатывают и руководствуются собственными Положениями.

В широком смысле пунктуальность – это выраженное в процентах отношение количества рейсов, выполненных с допустимой задержкой, к общему количеству выполненных рейсов. Пунктуальность может быть рассчитана как по отправлению, так и по прибытию. На примере отправлений в общем виде расчет пунктуальности можно представить следующим образом:

$$R_{dep} = \frac{K_d}{I} \cdot 100\% \qquad (3.1)$$

Расшифровка обозначений:

K_d – количество рейсов, у которых задержка отправления $\tau_i^0 \le T$ (T – время допустимой задержки);

I – общее число отправленных рейсов.

Существует несколько глобальных поставщиков данных по пунктуальности, которые публикуют в сети Интернет рейтинги для авиакомпаний и аэропортов. Далее представлены наиболее популярные из них.

Как упомянуто ранее, OAG - глобальный поставщик данных о перевозках со штаб-квартирой в Великобритании. Рейтинги OAG – наиболее достоверные и повсеместно используются авиакомпаниями для оценки своих позиций относительно конкурентов. В рейтинги OAG попадают все компании, предоставляющие данные не менее, чем по 80% рейсов. Согласно критериям OAG, рейс считаемся нерегулярным, если имело место опоздание более 15 минут от запланированного времени вылета или прилета. Отмененный рейс также считается нерегулярным.

Еще один крупнейший поставщик авиационных данных, наряду с OAG, – FlightStats. Регулярным рейсом по методике FlightStats считаются рейсы, прибывшие к месту стоянки в интервале [-15 минут; +15 минут] от плана.

Важность показателя пунктуальности для авиакомпаний обусловлена множеством факторов. Ключевые из них представлены далее.

Во-первых, пунктуальность напрямую влияет на лояльность клиентов авиакомпании. На рисунке 3.1 представлены статистические данные, показывающие зависимость между пунктуальностью прибытий и NPS одной из действующих авиакомпаний РФ. NPS – это индекс лояльности клиентов, один из ключевых показателей для оценки имиджа [75]. Очевидно, что лояльность пассажиров зависит не только от пунктуальности, однако, несмотря на это, коэффициент корреляции достигает значения 0,74.

Во-вторых, несоблюдение плановых времен вылета и прибытия рейсов приводит к дополнительным расходам для авиакомпаний, связанным с облуживанием пассажиров и рейсов, а также всевозможными штрафными санкциями со стороны обслуживающий компаний.

В-третьих, авиакомпании, выполняющие рейсы с хорошей пунктуальностью могут получить дополнительные скидки на наземное обслуживание от аэропортов. Подобные программы введены в таких аэропортах как Омск, Казань, Домодедово и других.

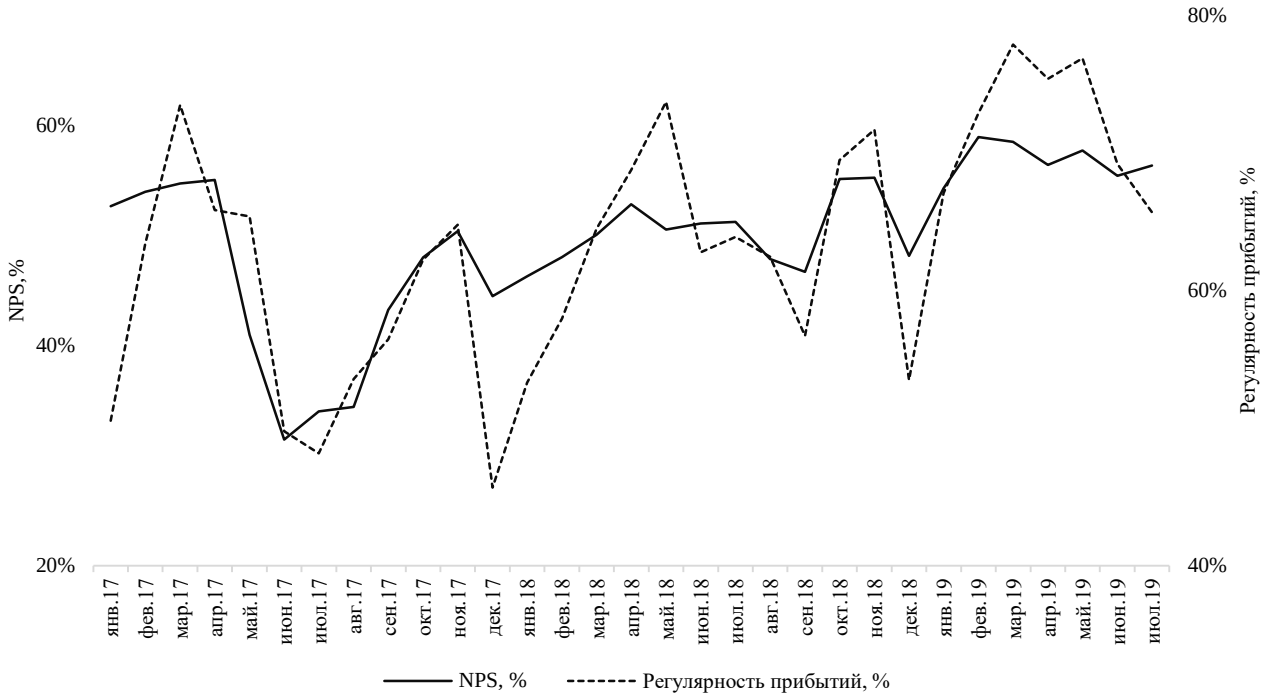


Рисунок 3.1 – Зависимость NPS и пунктуальности прибытий

Цель каждой авиакомпании – максимизировать пунктуальность, предельное значение которой – 100%. В настоящее время показатель пунктуальности отправлений и риск по пунктуальности оценивается по факту исполнения графика полетов. Однако, на пути достижения цели важно отслеживать уровень риска – показатель, позволяющий выявить нестабильность, возникающую в процессе достижения цели, на основе событий об отклонения от плана. Оценка уровня риска нарушения пунктуальности при принятии решения о корректировки расписания в оперативном режиме позволит авиакомпании повысить эффективность управленческих решений. Именно поэтому, при решении прикладных задач, к которым относится задача оперативного оптимального управления расписаниями, имеет смысл ориентироваться на целевые показатели транспортной системы, такие как пунктуальность.

3.2. Подход к оценке риска нарушения пунктуальности

Организации всех типов и размеров сталкиваются с внутренними и внешними факторами и воздействиями, которые порождают неопределенность в отношении того, достигнут ли они своих целей, и когда достигнут. Согласно ГОСТ Р ИСО 31000-2010 (Менеджмент риска. Принципы и руководство), влияние такой неопределенности на цели организации и есть "риск" [76]. В соответствии со Стандартом, процесс оценки рисков содержит три этапа: идентификация риска (процесс обнаружения, распознавания и описания рисков, включает выявление источников риска, событий, последствий), анализ (определение степени риска, в том числе количественная оценка уровня риска), оценка (процесс сравнения результатов анализа с установленными критериями риска для определения является ли риск приемлемыми или допустимыми, критерии определяются организацией и являются индикаторами относительно необходимости воздействия на риск).

Одним из методов оценки рисков является применение «Матрицы последствий и частот возникновения отклонений» [77]. Матрица последствий и частот (далее – матрица рисков) является средством объединения оценок последствий (тяжести событий) и частот возникновения отклонений, применяется для определения и ранжирования уровня риска, дальнейшей оценки того, требуется ли воздействие на риск или нет.

Входными данными для построения матрицы рисков являются шкалы последствий и частот возникновения негативных отклонений, установленные в соответствии с требованиями предприятия, а также матрица, которая их объединяет. Шкала может иметь любое количество точек. Наиболее распространены шкалы, имеющие 3, 4 или 5 точек. Шкала частот также может иметь любое количество точек.

Важно отметить, что в терминологии менеджмента риска термин "частота" (или "возможность") означает шанс того, что что-то может произойти, независимо от того, установлено ли это, измерено, определено объективно или субъективно,

качественно или количественно, и описывается ли с помощью общих понятий или математически. В английском языке в терминологии менеджмента риска используется термин "likelihood", чтобы придать ему широкий смысл, нежели "probability", который часто понимают в узком математическом смысле. С целью однозначного трактования материала, введем понятие ранг, для обозначения степени тяжести и частоты отклонений на шкале матрицы рисков.

В таблице 3.1 представлен пример матрицы рисков (R). По вертикали показана возрастающая серьезность последствий F, по горизонтали - ранг частоты возникновения негативных событий. Элементы матрицы отражают уровни риска и вычисляются следующим образом:

$$R = P \cdot F \tag{3.2}$$

Расшифровка обозначений:

F – степень тяжести по негативным отклонениям факта от плана (в абсолютных или в относительных величинах);

P – оценка частоты возникновения негативных отклонений.

Таблица 3.1 – Матрица рисков

		Ранг частоты негативных отклонений P				
		1	2	3	4	5
Ранг тяжести негативных отклонений F	1	1	2	3	4	5
	2	2	4	6	8	10
	3	3	6	9	12	15
	4	4	8	12	16	20
	5	5	10	15	20	25

Ранжирование уровня риска может быть произведено по шкале, разделенной на три интервала, которые позволяют определить степень приемлемости и необходимость воздействия на риск (таблица 3.2).

Таблица 3.2 – Уровни риска и необходимость воздействия

Количественная оценка уровня риска	**Качественная оценка уровня риска**	**Необходимость воздействия**
(0;5]	Незначительный	Отсутствует
(5;9]	Приемлемый	На усмотрение ответственного лица, принимающее решение
(9;25)	Недопустимый	Обязательна

Один из возможных способов оценки уровня риска по показателю «Пунктуальность отправлений», основан на применении матрицы риска и идеях, заложенных в работах [78, 79]. Описанный подход на практике успешно применяется в крупнейшем предприятии авиатранспортной отрасли РФ, что также подтверждено Актом (Приложение Д).

Определим шкалу тяжести в качестве входных данных для построения матрицы риска по критерию «Пунктуальность отправлений». Для рассматриваемого критерия тяжесть может быть выражена через задержку, то есть отклонение фактического времени вылета от планового. Как отмечено ранее, шкалы устанавливаются в соответствии с требованиями и целями предприятия. В таблице 3.3 представлена шкала, использующая на одном из действующий предприятий авиатранспортной отрасли.

Таблица 3.3 – Соотношение задержек и ранга тяжести отклонений

Ранг тяжести негативного отклонения F_i	**Время задержки, минут** τ_i^0
1	[1; 5]
2	(5;10]
3	(10; 15]
4	(15; 20]
5	Более 20 минут

Опираясь на подход, предложенный в работах [78, 79], ранг тяжести задержки конкретного рейса F_i [0÷5] может быть рассчитан по формуле:

$$F_i = \min\left[5; a \cdot \ln\left(1 + b \cdot \tau_i^0\right)\right] \tag{3.3}$$

Расшифровка обозначений:

τ_i^0 величина задержки начала выполнения *i*-го рейса;

$\tau_{i,j}^0$ – величина задержки начала выполнения *i*-го рейса на ВС *j*, $\tau_i^0 > T$, $\tau_{i,j}^0 > T$;

T – время допустимой задержки в минутах;

a и b - оценка коэффициентов логарифмической функции МНК (Рисунок 3.2). Для рассматриваемого примера $a = 4{,}9$ и $b = 8{,}7$.

Применение логарифмической функции при расчете тяжести позволяет получить наиболее точную эмпирическую оценку ранга в формате непорядкового числительного.

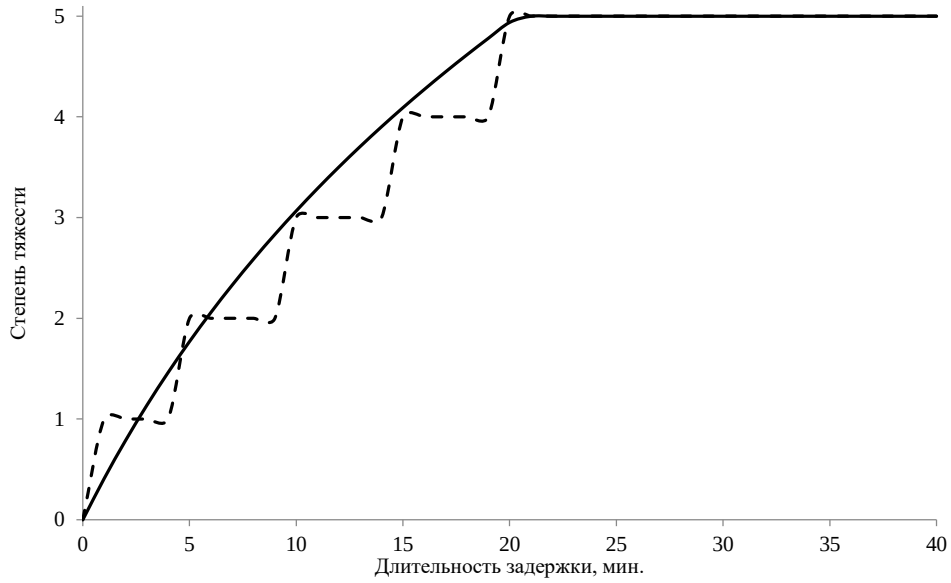


Рисунок 3.2 – График зависимости ранга тяжести от времени задержки

Определим также шкалу частот в качестве входных данных для построения матрицы риска (ранг частоты возникновения негативных отклонений). Для рассматриваемого критерия «Пунктуальность отправлений» ранг частоты возникновения негативных отклонений может быть выражен через долю рейсов,

имеющих задержку отправлений $\tau_i^0 > T$, где T – время допустимой задержки (таблица 3.4).

Таблица 3.4 – Соотношение доли рейсов с задержкой и ранга частоты возникновения негативных событий

Ранг частоты возникновения негативных отклонений P	**Доля рейсов с задержкой отправления** $\left(1-R_{dep}\right)$
1	(0; 0,09]
2	(0,09; 0,18]
3	(0,18; 0,27]
4	(0,27; 0,36]
5	(0,36; 1]

Формула расчета ранга частоты задержек может быть записана следующим образом:

$$P = \min\left[5; a \cdot \ln\left(1 + b \cdot \left(1 - R_{dep}\right)\right)\right] \tag{3.4}$$

Здесь a и b – коэффициенты логарифмической функции, определяющиеся МНК (рисунок 3.3); $\left(1-R_{dep}\right)$ - доля рейсов с задержкой отправления.

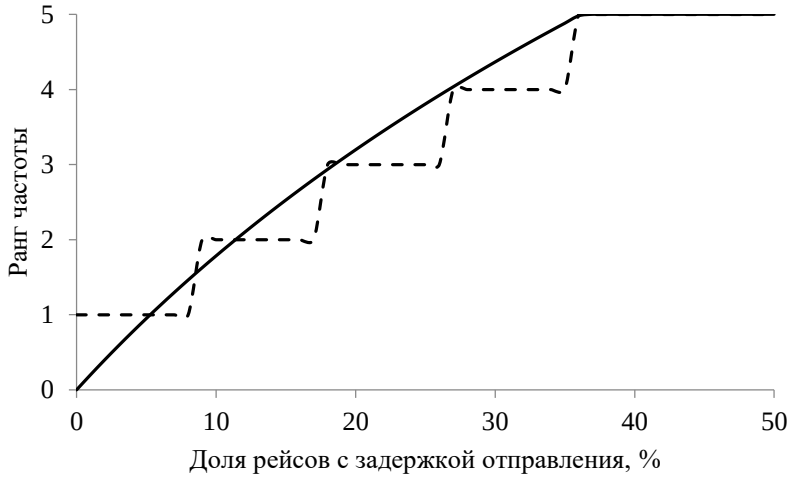


Рисунок 3.3 – График зависимости ранга частоты задержек и доли рейсов с задержкой.

Итоговый уровень риска по критерию «Пунктуальность отправлений» может быть записан как сумма произведения средней степени тяжести F_i по всем рейсам и ранга частоты негативного отклонения P:

$$R = \frac{\sum_{i=1}^{I} F_i}{K} \cdot P \to \min \tag{3.5}$$

Здесь I – общее количество рейсов; K – количество рейсов, имеющих задержку отправления $\tau_i^0 > T$, где T – время допустимой задержки.

3.3. Постановка задачи управления расписанием авиакомпании по критерию минимизации риска нарушения пунктуальности рейсов

Рассмотрим возможность применения в качестве критерия эффективности для регулирования расписания авиакомпании оригинальный проактивный критерий минимизации *риска* нарушения пунктуальности отправлений.

Как отмечено ранее, понятие риска включает в себя не только оценку величину отклонения фактического графика полетов от планового, но и частоту нарушений. Таким образом, применяя в качестве критерия оптимизации уровень риска, можем минимизировать не только задержки, но и систематические нарушения расписания, когда задержка может быть небольшой, но возникать постоянно. Данный подход позволяет синтезировать расписание, в котором определен баланс между *суммарными отклонениями от планового расписания* и *количеством рейсов с негативным отклонением* от запланированного графика через оценку уровня нарушений пунктуальности.

Выполним модификацию постановки задачи с априорным назначением последовательности выполнения рейсов, снабдив критерием минимизации задержек расписания от исходного графика.

Введем дополнительные обозначения.

Через S_i обозначим исходное расписание вылета рейса $i=\overline{1,I}$, пересчитанное в виде лага в минутах от начала суток (00:00 часов).

Если упорядочить рейсы для каждого ВС авиакомпании по возрастанию $\tau^0_{i,j}$ (задержка начала выполнения *i*-го рейса воздушным судном *j*), то $\mathrm{T}^0=\left\|\tau^0_{i,j}\right\|$, $j=\overline{1,J}$, $i=\overline{1,I}$, (I - общее число рейсов, J - число ВС) можно интерпретировать как расписание на входе каждого из ВС. Более точно $\tau^0_{i,j}$ - рассчитанные задержки вылета всех ВС на момент корректировки расписания от 00:00 часов с учетом расписания S_i, $i\in I_V$ и текущих задержек рейсов на всех шагах приведенного ниже алгоритма A_v.

Конечную постановку задачи оптимального регулирования расписаний флота авиакомпании можно представить в следующем виде.

Найти $x_{i,j}$, C_i, $C_{i,j}$, Δ при условиях:

$$\sum_{j=1}^{J} x_{i,j}=1,\ \ i=\overline{1,I}, \tag{3.6}$$

$$\underline{b_j}\le\sum_{i=1}^{I}x_{i,j}\le\overline{b_j},\ \ j=\overline{1,J}, \tag{3.7}$$

$$x_{i,j}=\begin{cases}1, & \text{если рейс } i \text{ назначен ВС } j,\\ 0 & \text{в противном случае,}\end{cases}\ i=\overline{1,I}, j=\overline{1,J}, \tag{3.8}$$

$$C_{i,j}\ge S_i, i=\overline{1,I},\ j=\overline{1,J}, \tag{3.9}$$

$$C_{i,j}\ge\tau^0_{i,j}x_{i,j}\ , i=\overline{1,I},\ \ j=\overline{1,J}, \tag{3.10}$$

$$C_{i,j}+t_{i,j}x_{i,j}\le C_{k,j}, i=\overline{1,I},\ j=\overline{1,J}, \tag{3.11}$$

$$\sum_{j=1}^{J} \tau_{i,j}^{0} x_{i,j} \le C_i \ , i = \overline{1,I} \ , \tag{3.12}$$

$$\Delta = \sum_{i=1}^{I} \left(C_i - S_i \sum_{j=1}^{J} x_{i,j} \right) \to \min . \tag{3.13}$$

Здесь (3.13) определяет условие минимизации отклонения расписания от запланированного. Для того чтобы минимизировать риск нарушения пунктуальности необходимо, с одной стороны обеспечить выполнение условия (3.13), с другой стороны – выполнить условие (3.5). Таким образом, имеет место бикритериальная задача синтеза оптимального графика движения ВС. В главе 3 представлены результаты вычислительных экспериментов применения риск-ориентированного подхода к оптимизации расписания.

3.4. Выводы

В данной главе рассмотрены возможные критерии оптимизации расписания, поскольку правильный выбор играет существенную роль с точки зрения оценки эффективности принятия решений. Исходя из анализа важности отслеживания авиакомпаниями пунктуальности ввиду значимого влияния данного показателя на производственные процессы и лояльность пассажиров, предложен оригинальный критерий минимизации риска нарушения пунктуальности рейсов при решении дискретной задачи управления назначениями ВС. Основная идея заключается в нахождении баланса между суммарными отклонениями от планового расписания и количеством рейсов с негативным отклонением от запланированного графика, который может быть выражен через расчет уровня риска нарушения пунктуальности и его оценки с применением «Матрицы последствий и частот возникновения отклонений».

Предложенный риск-ориентированный подход к управлению расписанием на практике успешно применяется в крупнейшем предприятии авиатранспортной отрасли РФ, что также подтверждено соответствующим Актом. В главе 4

представлены результаты и содержательная интерпретация вычислительных экспериментов, подтверждающих эффективность предложенного критерия.

ГЛАВА 4 ЭФФЕКТИВНЫЕ МЕТОДЫ РЕШЕНИЯ ЗАДАЧИ

Данная глава посвящена разработке вычислительно эффективных методов решения дискретных задач оптимизации управления сложными производственными процессами, к которым относится оперативное управление расписанием авиакомпании. Под вычислительной эффективностью понимается полиномиальная трудоемкость поиска решения рассматриваемой задачи дискретной оптимизации. Приводятся результаты вычислительных экспериментов алгоритмов принятия решений по переназначению воздушных судов, а также их содержательная интерпретация.

4.1. Средства реализации и алгоритмы решения задачи регулирования расписаний с дизъюнкциями в ограничениях

Существует множество алгоритмов решения рассматриваемой задачи регулирования расписаний с использованием нескольких формальных постановок. Для поиска решений используются различные подходы, от разновидностей локального поиска [83, 84, 85], аппроксимационных алгоритмов [85, 86], программирования в ограничениях [87] до точных алгоритмов, включая динамическое программирование и методы ветвей и отсечений [3, 73, 74]. Однако, несмотря на принадлежность к классу NP, неплохие результаты показывает применение точных численных методов оптимизации. Это относится, в частности, к развиваемому в работах [88, 89] АБОВ, и набору «классических» средств решения milp, реализованных в IBM CPLEX optimization studio и GURUBI optimization. Практические результаты применения АБОВ для решения задачи оптимизации расписаний несвязанных параллельных приборов с задержками начала обслуживания по критерию в постановке с рекурсиями для относительно небольших размерностей (от 5x100 до 30x100) представлены в [90]. В [90] также приведено сравнение применений АБОВ с параметрическими алгоритмами на основе динамического программирования с отсевом локально наихудших расписаний на каждом шаге динамического программирования, которое выявило полное преимущества АБОВ на задачах таких размерностей.

Для иллюстрации результатов далее приведены результаты тестирования модели в постановке (2.34)-(2.36),(2.40)-(2.43) посредством набора компонент IBM CPLEX optimization studio. Тестирование модели для составления оптимальных расписаний флота авиакомпании проведено на сгенерированных исходных данных, максимально приближенных к реальным размерностям.

Использованы 2 группы тестов, по 10 в каждой группе [90]. Первая группа содержит данные для 500 рейсов, осуществляемых 10 бортами, вторая – для 300 рейсов, осуществляемых 30 бортами. Границы 10 и 30 бортов соответствуют среднему и максимальному количеству ВС одного типа действующей авиакомпании. Верхние границы 300 и 500 рейсов существенно превосходят существующие потребности оперативного регулирования действующей авиакомпании. В тестах учтено, что воздушные суда могут обладать одинаковой вместимостью, но разным временем выполнения рейсов за счет различий в скорости и времени обслуживания.

Структуру данных и результатов счета тестов можно описать посредством небольшого иллюстративного примера теста на составление расписания выполнения 10 рейсов 3 бортами. Исходные данные примера содержатся в таблицах 4.1-4.3. Таблицы 4.4-4.6 отображают оптимальное решение.

Таблица 4.1 – Данные о времени $\tau_{i,j}$ выполнения рейсов ВС

j \ i	1	2	3	4	5	6	7	8	9	10
1	2	4	1	9	7	9	6	7	3	3
2	1	4	3	5	3	8	3	4	5	3
3	6	1	3	1	3	6	3	4	6	9

Таблица 4.2 – Данные о задержках рейсов $\tau^0_{i,j}$ без упорядочения

j \ i	1	2	3	4	5	6	7	8	9	10
1	9	10	14	7	3	3	6	14	7	0
2	12	11	0	10	13	14	12	12	11	15
3	9	10	14	7	3	3	6	14	7	0

Таблица 4.3 – Порядок $L_j = \|l_i\|_j$ следования задержек рейсов по возрастанию $\tau^0_{i,j}$

j \ i	1	2	3	4	5	6	7	8	9	10
1	4	10	8	1	5	2	7	9	6	3
2	10	5	6	7	4	9	1	2	3	8
3	3	4	2	9	1	7	8	5	6	10

Таблица 4.4 – Оптимальные назначения рейсов $x^*_{i,j}$

j \ i	1	2	3	4	5	6	7	8	9	10
1	0	0	0	0	0	0	0	1	1	0
2	1	0	0	0	1	1	0	0	0	1
3	0	1	1	1	0	0	1	0	0	0

Таблица 4.5 – Оптимальное время вылетов $C^*_{i,j}$

j \ i	1	2	3	4	5	6	7	8	9	10
1	12	12	15	0	12	15	12	5	12	5
2	14	15	15	14	3	6	14	15	14	0
3	12	11	0	10	15	15	12	15	12	15

Таблица 4.6 – Время начала выполнения назначенных рейсов $C^*_{i,j}x^*_{i,j}$

j \ i	1	2	3	4	5	6	7	8	9	10
1	0	0	0	0	0	0	0	5	12	5
2	14	0	0	0	3	6	0	0	0	0
3	0	11	0	10	0	0	12	0	0	0

Результаты счета обеих групп тестов отражены соответственно в таблице 4.7 и таблице 4.8. В обеих таблицах графы «Время» отображают время счета в формате «часы:минуты:секунды». Графы $C_{\max}$ содержат достигнутые значения критерия эффективности для каждого теста. Графы «Макс. откл» отражают максимальное

возможное отклонение полученного решения от оптимума в процентах значения $C_{\max}$.

Таблица 4.7 – Результаты счета тестовых задач (500 рейсов 10 ВС)

Номер теста	Имя теста	Время (ч:мм:сс)	$C_{\max}$	Макс.откл (%)
1	500101	0:11:39	727	0,5
2	500102	0:25:10	717	0,5
3	500103	0:01:42	709	0,5
4	500104	0:00:20	733	0,5
5	500105	0:09:01	709	0,5
6	500106	0:01:00	700	0,5
7	500107	0:03:14	697	0,5
8	500108	0:02:30	723	0,5
9	500109	0:00:35	704	0,5
10	500110	0:02:36	710	0,5

Таблица 4.8 – Результаты счета тестовых задач (300 рейсов 30 ВС)

Номер теста	Имя теста	Время (ч:мм:сс)	$C_{\max}$	Макс.откл (%)
1	300301	0:00:07	225	0,5
2	300302	0:47:18	170	2,5
3	300303	0:30:49	163	2,2
4	300304	0:28:44	172	2,4
5	300305	0:16:09	171	2,5
6	300306	0:12:29	166	0,5
7	300307	0:00:01	172	0,5
8	300308	0:08:00	160	0,5
9	300309	0:56:38	167	2,7
10	300310	0:37:36	174	1,7

Первая группа тестов (10 бортов, 500 рейсов) гарантирует отклонение от оптимума не более 0,5% значения критерия, что практически можно считать точным результатом. Время решения находится в интервале от 20 секунд до 25 минут 10 секунд. Десятиминутный интервал счета превышен для 2-х тестов из 10.

Вторая группа тестов (30 бортов, 300 рейсов) ожидаемо показала ухудшение гарантированной точности (в интервале от 0,5% до 2,7% максимального

отклонения от оптимума). Продолжительность решения находится в интервале от 1 секунды до 56 минут 38 секунд. Получасовой интервал превышен только для 3 тестов из 10. В целом, можно констатировать вполне обнадеживающие результаты исследований и разработок, относящихся к разряду фундаментальных, и перспективы их прикладного применения.

Таким образом, результаты экспериментально доказывают перспективность внедрения разработанного инструментария для реальных авиакомпаний [95].

4.2. Декомпозиция задачи регулирования расписаний с дизъюнкциями в ограничениях

Анализ результатов применения формализации (3.6) – (3.13) на реальных данных (таблица 4.9) выявил существенный недостаток постановки, неочевидный для сгенерированных исходных данных, на которых задача и алгоритмы ее решения тестировалась.

Таблица 4.9 – Результаты применения формализации задачи с дизъюнкциями в ограничениях на реальных данных

Маршрут рейса	Номер рейса	Бортовой номер воздушного судна			
		VP-BTN	VP-BOM	VP-BPC	VQ-BDI
DME-LED	25	1			
DME-ROV	1151				1
DME KRR	1149		1		
DME-LED	23			1	
LED DME	44			1	
DME KUF	29	1			
AER DME	1030				1
DME MRV	1219		1		
DME KZN	69			1	
DME OMS	167			1	
DME BQS	125				1
DME NSK	2251		1		

Полученные назначения ВС на рейсы содержат противоречия, в частности, аэропорт вылета последующего рейса должен совпадать с аэропортом прибытия

предыдущего рейса. Так, например, борт VP-BTN имеет цепочку из двух рейсов: DME-LED и DME-KUF. Данная цепочка не может быть выполнена, так как аэропорт прибытия первого рейса (LED) не совпадает с аэропортом вылета последующего рейса (DME).

Данная ситуация возникает из-за того, что в параллельной обслуживающей системе приборы не связаны и логически связанные цепочки рейсов автоматически возникают только при неизменных значениях задержек $\tau_{i,j}^{0}$. В случае параллельно последовательной системы $\tau_{i,j}^{0}$ могут меняться и, как следствие, связанные цепочки могут формироваться ошибочно. В нашем случае значения начальных задержек $\tau_{i,j}^{0}$, считавшиеся постоянными для любых назначений [2], таковыми не оказались. Причинно-следственные связи в цепочках опосредуют рекурсивную зависимость задержек $\tau_{i,j}^{0}$ от предшествующих назначений. Логический вывод, следующий из выявленных противоречий, заключается в том, что система «ВС-рейсы» в общем случае не может быть отнесена к разряду параллельных систем.

Цепочка рейсов - это технологический маршрут, что относится к последовательной системе. При этом, поскольку в нашем случае присутствует также множество несвязанных между собой параллельных приборов (ВС), и для каждого из них формируется последовательная подсистема с заданными маршрутами, система является параллельно-последовательной (рисунок 4.1).

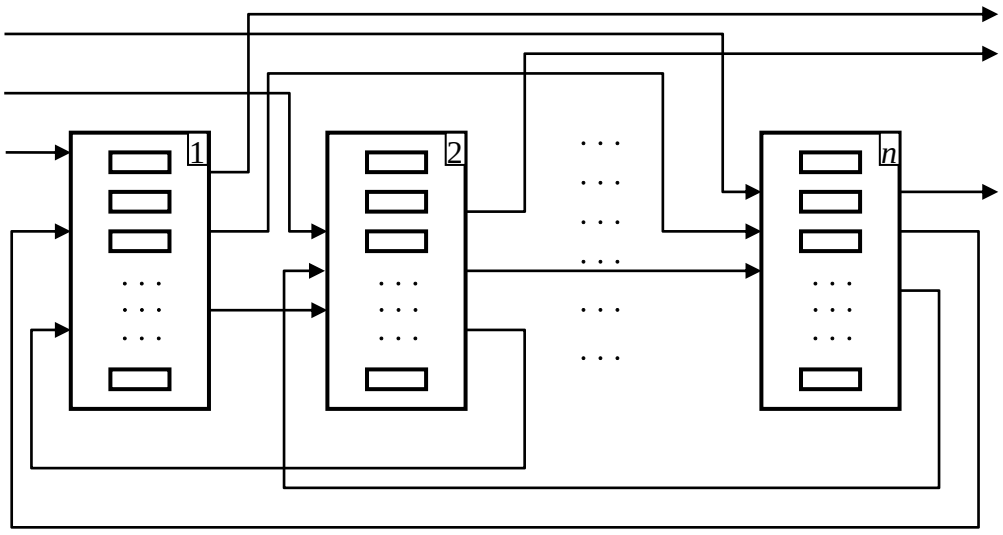


Рисунок 4.1 – Параллельно-последовательная обслуживающая система

Общую постановку такой задачи управления расписаниями в ППОС, связанных с ней подзадач и описание некоторых алгоритмов решения можно найти в работах [38-41]. Прямое раскрытие рекурсий, как правило, приводит к значительному росту (иногда на много порядков) размерностей за счет появления дополнительных логических переменных и ограничений (см., например, [74]), что недопустимо для задачи оперативного управления с учетом принадлежности к классу NP. Для решения задач управления ППОС естественно применение соответствующих подходов. Поэтому далее рассматривается применение неполной декомпозиции с релаксацией и использование быстрого эвристического многошагового алгоритма впоследствии.

Сформулируем подзадачу назначения единственного рейса на каждое из P_ν воздушных судов на шаге ν.

Как отмечалось ранее, исходно имеем:

J - число ВС, где j - номер ВС, $j = \overline{1, J}$;

I - число рейсов, i -номер рейса, $i = \overline{1, I}$, положим $I \geq J$;

S_i - исходное расписание вылета рейса $i = \overline{1, I}$;

Пусть I_ν - подмножество рейсов, переназначаемых на шаге ν, $\nu = \overline{1,\mathrm{N}}$, где N - число подзадач и шагов алгоритма. Тогда номера рейсов на шаге ν принадлежат I_ν $(i \in I_\nu)$, $\bigcap_{\nu=1}^{\mathrm{N}} I_\nu = \varnothing$, это условие фактически означает, что назначения ВС на рейсы на шаге ν не могут быть переназначены на последующих шагах.

Через Δ_ν обозначим суммарное отклонение расписаний на шаге ν от исходного, через C_i - пересчитанное время вылета рейса i на шаге ν, $(i \in I_\nu)$.

Как и ранее, определим переменные назначений ВС на рейсы.

$$x_{i,j} = \begin{cases} 1, & \text{если рейс } i \text{ назначен ВС } j \text{ на шаге } \nu, \\ 0 & \text{в противном случае,} \end{cases} \text{, где } j = \overline{1,J}, i \in I_\nu . \tag{3.14}$$

Тогда подзадача назначения единственного рейса на каждое из P_ν воздушных судов на шаге ν, $P_\nu \le [J/2], \nu = \overline{1,\mathrm{N}}$ принимает вид:

Найти $x_{i,j}$, C_i , Δ_ν при условиях:

$$0 \le x_{i,j} \le 1, \ j = \overline{1,J}, i \in I_\nu , \tag{3.15}$$

$$\sum_{j=1}^{J} x_{i,j} \le 1, \ i \in I_\nu , \tag{3.16}$$

$$\sum_{i \in I_\nu} x_{i,j} \le 1, \ j = \overline{1,J} , \tag{3.17}$$

$$\sum_{i \in I_\nu} \sum_{j=1}^{J} x_{i,j} \ge P_\nu , \ P_\nu \le [J/2], \tag{3.18}$$

$$\sum_{j=1}^{J} \tau_{i,j}^{0} x_{i,j} \le C_i , i \in I_\nu , \tag{3.19}$$

$$\Delta_\nu = \sum_{i \in I_\nu} \left(C_i - S_i \sum_{j=1}^{J} x_{i,j} \right) \to \min, \tag{3.20}$$

где $[\cdot]$ целая часть числа, и $\sum_{\nu=1}^{N} P_\nu = I$.

С учетом абсолютной унимодулярности условий (3.15) – (3.18), булевы переменные можно заменить непрерывными с интервальными ограничениями: $0 \le x_{i,j} \le 1,\ j = \overline{1,J}, i \in I_\nu$.

После любого шага ν задержки $\tau^0_{i,j},\ i \in I_\nu,\ j = \overline{1,J}$ должны быть пересчитаны.

Координирующая задача:

$$\sum_{\nu=1}^{N} P_\nu = I \tag{3.21}$$

$$\Delta = \sum_{\nu=1}^{N} \Delta_\nu \to \min \tag{3.22}$$

Совершенно очевидна полиномиальная разрешимость обеих задач в виду отсутствия цикличности обращений к подзадачам (3.15)-(3.20) класса линейного программирования при решении координирующей задачи (3.21)-(3.22). Соответственно, пошаговый алгоритм приближенного решения задачи оптимального регулирования расписания вылетов ВС имеет полиномиальную трудоемкость. Обозначим этот алгоритм, как A_ν и приведем его запись.

4.3. Декомпозиционный алгоритм оперативного оптимального регулирования расписания

Алгоритм A_ν.

1. Исходные данные – множества ВС $\{1,2,...,J\}$ и рейсов $\{1,2,...,I\}$, известно исходное расписание вылетов рейсов S_i, $i=\overline{1,I}$ и $t_{i,j}$ - время перелета, разворота и подготовки рейса i ВС j. Разбиение множества рейсов, упорядоченных по времени вылета S_i, $i=\overline{1,I}$ на последовательность подмножеств $\{I_1,I_2,...,I_\nu,...,I_{\mathrm{N}}\}$ априорно не определено, однако можно заранее задать числа элементов этих подмножеств посредством последовательности, $\{P_1,P_2,...,P_\nu,...,P_{\mathrm{N}}\}$. Дополнительные условия такого разбиения $P_\nu \le [J/2], \nu=\overline{1,\mathrm{N}}$, $[\cdot]$ целая часть числа, и $\sum_{\nu=1}^{\mathrm{N}} P_\nu = I$.

2. Задаем номер шага $\nu := 0$.

3. Увеличиваем номер шага $\nu := \nu+1$. Вычисляем возможные задержки рейсов для всех ВС на шаге ν $\tau^0_{i,j}$, $i\in I_\nu$, $j=\overline{1,J}$.

4. Решаем подзадачу (3.15)-(3.20). Фиксируем оптимальные назначения $x^*_{i,j}$, $j=\overline{1,J}, i\in I_\nu$, скорректированное расписание вылетов C^*_i, $i\in I_\nu$ и значение критерия Δ^*_ν.

5. Проверка на окончание алгоритма: При выполнении условия $\nu=\mathrm{N}$ следующий пункт. Иначе возврат к 3.

6. Объединение назначений, полученных на всех предшествующих шагах $x^*_{i,j}$, $j=\overline{1,J}, i\in I_\nu$, $\nu=\overline{1,\mathrm{N}}$, расписаний вылета, C^*_i, $i\in I_\nu$, $\nu=\overline{1,\mathrm{N}}$ и вычисление общего времени отклонения от исходного расписания $\Delta=\sum_{\nu=1}^{\mathrm{N}}\Delta^*_\nu$.

Конец алгоритма.

Совершенно очевидна зависимость конечного результата решения общей задачи оптимального регулирования расписания от исходного разбиения

$\{I_1, I_2, ..., I_\nu, ..., I_N\}$, и, следовательно, от набора $\{P_1, P_2, ..., P_\nu, ..., P_N\}$. Вариантов выбора числа переназначаемых рейсов по шагам $\{P_1, P_2, ..., P_\nu, ..., P_N\}$ может быть сколь угодно много. В то же время вычислительные эксперименты показывают незначительность изменений значений критерия $\Delta = \sum_{\nu=1}^{N} \Delta_\nu$ при варьировании значений P_ν для равных N. Заметим, что минимальное значение N зависит от максимальной длины выстраиваемых цепочек рейсов и предпочтительно должно быть четным числом. Проиллюстрируем применение формализации (3.15) – (3.22) и алгоритма A_ν на небольшом примере.

4.3.1. Результаты тестирования декомпозиционного алгоритма

Применение описанного выше инструментария проиллюстрировано на поэтапном переформировании небольшого фрагмента суточного расписания авиакомпании.

Результаты применения A_ν при пересчете фрагмента расписания по шагам сведены в таблицы 4.10-4.14. Рассматривается 10 рейсов и 11 бортов. Обозначения рейсов (столбцы таблиц) и бортов (строки) соответствуют отраслевым стандартам. Для рассматриваемого примера рациональное значение N равно 2, поскольку во временной интервал фрагмента расписания укладываются цепочки рейсов длиной не более 2. Соответственно, алгоритм A_ν реализуется за два шага. Задаваемые на входе алгоритма наборы числа рейсов по шагам $\{4,6\}$, $\{5,5\}$, $\{6,4\}$. Последние два варианта приводят к одинаковым локально наилучшим результатам. Результаты расчетов для последнего варианта числа рейсов по шагам приведены в таблицах.

Таблица 4.10 – Задержки $\tau^0_{j,i}$, на шаге 1 ($i \in I_1$)

Рейс / Борт	KZN - DME	DME - LED	LED - DME	DME - ROV	ROV - DME	DME - KZN	KZN - DME	DME - KUF	KUF - DME	DME - KZN
VP-BHF	1055	1195	1385	1090	10000	1160	1205	1125	10000	1400
VP-BHG	1160	1090	1330	995	1280	1065	1300	1030	1285	1305
VP-BHI	1150	1090	1330	985	1270	1055	1290	1020	1275	1295
VP-BHJ	1160	1100	1340	995	1280	1065	1300	1030	1285	1305
VP-BHL	1155	1095	1335	980	1265	1060	1295	1025	1280	1300
VP-BHP	1155	1095	1335	990	1275	1060	1295	1025	1280	1300
VP-BHQ	1155	1095	1335	990	1275	1060	1295	1025	1280	1300
VP-BTN	1165	1105	1345	1000	1285	1060	1295	1035	1290	1310
VP-BTP	1167	1107	1347	1002	1287	1072	1307	1037	1292	1312
VP-BTS	1150	1090	1330	985	1270	1055	1290	1020	1275	1295
VP-BTU	1157	1097	1337	992	1277	1062	1297	1017	1272	1302

Таблицы 4.10 и 4.11 содержит вычисляемые пошагово значения возможных задержек вылета в минутах каждого из бортов по каждому из маршрутов $\tau_{j,i}^{0}$, формально являющиеся элементами транспонированной матрицы $\mathrm{T}^{0} = \left\| \tau_{i,j}^{0} \right\|$. Время задержки $\tau_{i,j}^{0}$ вычисляется от начала суток с учетом времени появления в аэропорту вылета по действующему расписанию, фактической задержки на момент пересчета расписания, времени разворота и подготовки к вылету.

Таблица 4.11 – Задержки $\tau_{j,i}^{0}$, на шаге 2 ($i \in I_2$)

Рейс / Борт	LED -DME	ROV -DME	KZN -DME	KUF -DME
VP-BHF	1385	10000	1305	10000
VP-BHG	1330	1380	1400	1385
VP-BHI	1430	1270	1295	1275
VP-BHJ	1340	1280	1300	1285

VP-BHL	1335	1265	1295	1280
VP-BHP	1335	1275	1295	1280
VP-BHQ	1335	1275	1295	1280
VP-BTN	1345	1285	1295	1290
VP-BTP	1347	1287	1307	1292
VP-BTS	1550	1370	1290	1375
VP-BTU	1337	1277	1297	1272

Перед каждым шагом $\tau^0_{i,j}$ пересчитываются с учетом состояния системы после предыдущего шага. Заливкой в таблицах 4.10 и 4.11 выделены задержки, соответствующие переназначениям бортов на рейсы.

Таблица 4.12 – Результирующие назначения

Рейс / Борт	KZN - DME	DME - LED	LED - DME	DME - ROV	ROV - DME	DME - KZN	KZN - DME	DME - KUF	KUF - DME	DME - KZN
VP-BHF	1	0	0	0	0	0	0	0	0	0
VP-BHG	0	1	**1**	0	0	0	0	0	0	0
VP-BHI	0	0	0	0	0	0	0	0	0	1
VP-BHJ	0	0	0	0	0	0	0	0	0	0
VP-BHL	0	0	0	1	**1**	0	0	0	0	0
VP-BHP	0	0	0	0	0	0	0	0	0	0
VP-BHQ	0	0	0	0	0	0	0	0	0	0
VP-BTN	0	0	0	0	0	0	0	0	0	0
VP-BTP	0	0	0	0	0	0	0	0	0	0
VP-BTS	0	0	0	0	0	1	**1**	0	0	0
VP-BTU	0	0	0	0	0	0	0	1	**1**	0

Таблица 4.12 аккумулирует сделанные назначения бортов на рейсы $x^*_{i,j}$ за два шага алгоритма. Назначения первого шага выделены светлой, второго – более темной заливкой.

В результате сформированы четыре двухзвенные цепочки рейсов, сопряженные во времени и пространстве. Еще два рейса, совместимые для стыковки по ВС, назначены разным бортам.

В результате минимальное суммарное отклонение нового расписания от исходного Δ^* составило 799 минут. Наибольший ущерб выявляется на последних этапах работы алгоритма (на втором для рассмотренного примера). Заливкой выделены отклонения второго этапа. Кроме этого во все значения $\tau^0_{i,j}$ в программной реализации A_v включены получасовые резервы времени.

Таблица 4.13 – Минимальные отклонения от планового расписания

	KZN - DME	DME - LED	LED - DME	DME - ROV	ROV - DME	DME - KZN	KZN - DME	DME - KUF	KUF - DME	DME - KZN
S_i =	1015	1050	1195	945	1110	1015	1155	980	1130	1255
$\Delta_{v,i}$ =	40	40	135	35	155	40	135	37	142	40
C_i =	1055	1090	1330	980	1265	1055	1290	1017	1272	1295

Поэтому для рассмотренного примера фактическое суммарное отклонение скорректированного расписания от исходного составит 499 минут.

Тестирование программной реализации алгоритма проводилось на данных суточного расписания действующей авиакомпании. В расписании 90 рейсов и 86 бортов. По оперативным данным констатируется множество существенных сбоев в реализации исходного расписания и ряд относительно некритических задержек вылетов. В частности, 26 рейсов задержаны на время от 80 до 215 минут и 27 рейсов – от 15 до 75 минут.

В результате корректировок общую длительность расписания удается сократить более чем на 5% относительно исходного. Тестирование также подтвердило существенную зависимость качества пересчитываемого расписания от исходного разбиения $\{P_1, P_2, ..., P_v, ..., P_N\}$, тем самым выявив факт зависимости эффективности применения аппарата от наличия набора рациональных разбиений и, следовательно, от опыта диспетчера. Возникший при этом вопрос выбора более

глубоко не анализировался, являясь отдельной темой исследования. Время счета одного варианта корректировки расписания для любого разбиения $\{P_1, P_2, ..., P_V, ..., P_N\}$ и исходных данных обозначенной выше размерности составляет несколько секунд при использовании настольного компьютера.

4.3.2. Исследование эффективности применения эвристических подходов к разбиению множества рейсов

Декомпозиционный подход к решению предполагает наличие эвристики в части выбора подхода к разбиению множества рейсов. Далее приведены результаты вычислительного эксперимента регулирования расписания в соответствии с постановкой (3.6) – (3.13) с использованием различных подходов к разбиению множества рейсов (таблица 4.14). Входные данные дня расчетов – суточный план полетов действующей авиакомпании, который содержит 90 рейсов и 85 бортов в парке ВС.

Таблица 4.14 – Перечень эвристических правил разбиения множества рейсов

			Шаг				
Алго-ритм	**Подход к разбиению**		**1**	**2**	**3**	**4**	**5**
A1	Порядок выборок соответствует исходному расписанию. На первых 4 шагах выбираются первые 30 рейсов из списка оставшихся.	Количество рейсов	30	30	30	30	10
		Количество бортов	20	20	20	20	10
A2	На каждом шаге в перераспределении участвуют все оставшиеся рейсы.	Количество рейсов	90	70	50	30	10
		Количество бортов	20	20	20	20	10
A3	Порядок соответствует исходному расписанию	Количество рейсов	90	50	10	-	-

		Количество бортов	40	40	10	-	-
A_4	Упорядочение выборки рейсов по времени вылета.	Количество рейсов	30	30	30	30	10
		Количество бортов	20	20	20	20	10

В таблицах 4.15 – 4.18 приведены расчеты, выполненные на срезе (7 рейсов) суточного расписания.

Каждая таблица содержит плановое (S_i) и расчетное (C_i) расписание, отклонение расчетного графика от планового ($\Delta_{v,i}$) и соответствующие значения степени тяжести отклонения (F_i), а также исходное время вылета каждого рейса и расчетное (выделено жирным шрифтом). Заливкой выделены скорректированное время вылета по рейсам.

Таблица 4.15 - Назначения $x^*_{i,j}$, полученные по алгоритму A_1

Номер рейса	64	69	29	98	44	24	23
Направление	KZN - DME	DME - KZN	DME - KUF	UFA - DME	LED - DME	LED - DME	DME - LED
$S_i =$	1155	1255	1255	1050	1025	1130	985
$C_i =$	1155	1255	1255	1059	1072	1330	985
$\Delta_{v,i} =$	0	0	0	9	47	200	0
$F_i =$	0	0	0	2,83	5	5	0
	Распределение ВС по рейсам						
VP-BHF							
VP-BTN	19:15						
VP-BTS			20:55				
VP-BTU					**17:52**		
VQ-BRC		20:55		17:30			
VQ-BYL							
VP-BTP					17:05		
VQ-BYD							
VP-BHQ						18:50	16:25

VP-BHK						**22:10**	

Для подмножества рейсов, представленных в таблице 4.15, ранг частоты задержек составил 5. Уровень риска R_{A_1} по критерию «Пунктуальность отправлений» составил 4,5.

Таблица 4.16 – Назначения $x^*_{i,j}$, полученные по алгоритму A_2

Номер рейса	64	69	29	98	44	24	23
Направление	KZN - DME	DME - KZN	DME - KUF	UFA - DME	LED - DME	LED - DME	DME - LED
$S_i =$	1155	1255	1255	1050	1025	1130	985
$C_i =$	1155	1312	1255	1059	1032	1190	985
$\Delta_{v,i} =$	0	57	0	9	7	60	0
$F_i =$	0	5	0	2,83366	2,3305	5	0
	Распределение ВС по рейсам						
VP-BHF							
VP-BTN	19:15						
VP-BTS			20:55				
VP-BTU							
VQ-BRC		20:55		17:30			
VQ-BYL							
VP-BTP		**21:52**			17:05		
VQ-BYD							
VP-BHQ						18:50	16:25
VP-BHK							
VP-BDG (резерв)						**19:50**	

В таблице 4.16 представлен фрагмент расписания, сформированного при использовании разбиения рейсов по алгоритму ***A₂***. Не трудно заметить, что назначения ВС на рейсы при различных подходах к разбиению отличаются. В данном примере борт VP-BDG – это резервное воздушное судно, которое на момент формирования расписания находилось в базовом аэропорту.

В таблице 4.17 приведено расписание, синтезированное при разбиении рейсов по алгоритму разбиения ***A₃***.

Таблица 4.17 – Назначения $x^*_{i,j}$, полученные по алгоритму A_3

Номер рейса	64	69	29	98	44	24	23

Направление	KZN - DME	DME - KZN	DME - KUF	UFA - DME	LED - DME	LED - DME	DME - LED
$S_i =$	1155	1255	1255	1050	1025	1130	985
$C_i =$	1155	1312	1255	1059	1032	1330	985
$\Delta_{v,i} =$	0	57	0	9	7	200	0
$F_i =$	0	5	0	2,83366	2,3305	5	0
	Распределение ВС по рейсам						
VP-BHF							
VP-BTN	19:15						
VP-BTS			20:55				
VP-BTU							
VQ-BRC		20:55		17:30			
VQ-BYL							
VP-BTP		**21:52**			17:05		
VQ-BYD							
VP-BHQ						18:50	16:25
VP-BHK						**22:10**	
VP-BDG (резерв)							

В таблице 4.18 приведено расписание, синтезированное при разбиении рейсов по алгоритму разбиения ***A₄***.

Таблица 4.18 – Назначения $x^*_{i,j}$, полученные по алгоритму A_4

Номер рейса	64	69	29	98	44	24	23
Направление	KZN - DME	DME - KZN	DME - KUF	UFA - DME	LED - DME	LED - DME	DME - LED
$S_i =$	1155	1255	1255	1050	1025	1130	985
$C_i =$	1155	1255	1255	1059	1127	1190	985
$\Delta_{v,i} =$	0	0	0	9	102	60	0
$F_i =$	0	0	0	2,83366	5	5	0
	Распределение ВС по рейсам						
VP-BHF							
VP-BTN	19:15						
VP-BTS			20:55				
VP-BTU							
VQ-BRC		20:55		17:30			
VQ-BYL							
VP-BTP					17:05		
VQ-BYD							
VP-BHQ						18:50	16:25
VP-BHK							

VP-BDG (резерв)						**19:50**	

В таблице 4.19 представлено сравнение эффективности работы алгоритма корректировки расписания и решений, принимаемых сотрудниками ЦУП авиакомпании (представленные данные являются реальными, предоставлены действующей авиакомпанией). Здесь R_{A_n} – это уровень риска нарушения пунктуальности отправлений, рассчитанный для четырех различных вариантов синтеза расписания, отличающихся различным подходом к разбиению множества рейсов.

Таблица 4.19– Сравнение параметров расчетного оптимального расписания и фактического расписания.

	Факт	R_{A_1}	R_{A_2}	R_{A_3}	R_{A_4}
Доля рейсов с задержкой	50,0%	26,7%	26,7%	27,8%	24,4%
Суммарное время задержек	2405	975	1083	938	1003
Уровень риска	9,5	4,5	4,4	4,9	3,5
Всего рейсов	90	90	90	90	90

Отметим очевидное преимущество применения алгоритма оптимизации расписания. Наиболее эффективное решение по критерию минимизации уровня риска нарушения пунктуальности получено при разбиении рейсов по алгоритму A_4, который предполагает упорядочивание рейсов по плановому времени вылета. Без использования автоматизированных средств действия сотрудников ЦУП приводят к дополнительным расходам авиакомпании, связанным с задержками рейсов. Для данной выборки рейсов время задержки рейсов могло быть сокращено на 50% относительно фактически реализовавшегося расписания за счет применения оптимизационных решений [97].

4.4. Выводы

Выполнена разработка нового вычислительно эффективного метода решения поставленной задачи дискретной оптимизации, использующего декомпозиционный подход, линейную релаксацию и быстрое поэтапное решение

релаксированных подзадач, что позволяет перевести NP-трудную задачу управления назначения ВС в класс полиномиально разрешимых. Предложенный подход принципиально отличается от всех известных ранее.

Приведенные результаты вычислительных экспериментов свидетельствуют о высокой эффективности решения, как с точки зрения быстродействий, так и с точки зрения принятия оптимального решения. Выполнено сравнение результатов решения задачи с применением декомпозиционного алгоритма, в основе которого заложены различные варианты декомпозиции, в том числе произведено сравнение с фактически реализовавшимся расписанием. Наиболее эффективным оказался принцип разбиения задачи регулирования расписания при упорядочивании рейсов по времени вылета. Результаты расчетов говорят о том, что применение разработанного алгоритма может сократить расходы, связанные с задержками рейсов, на 50 %. Это говорит о крайне высокой практической ценности достигнутых результатов.

ГЛАВА 5 ПРОГРАММНАЯ РЕАЛИЗАЦИЯ МЕТОДОВ ОПТИМИЗАЦИИ РАСПИСАНИЯ

В данной главе представлена программная реализация предложенного подхода к принятию решений по переназначению ВС с применением декомпозиционного алгоритма. Представляемая программа является прототипом СППР и ориентирована, в первую очередь, на решение задачи сотрудника ЦУП авиакомпании, а именно - на принятие решений по корректирование расписания в случае сбойной ситуации, разбалансировки и отклонении расписания от запланированного графика. Полученные в этой главе результаты отражены в публикации [95].

5.1. Программное обеспечение при оперативном управлении расписаниями авиакомпании

В настоящее время авиакомпании все чаще проявляют интересе к использованию специализированного ПО при принятии решений о переназначении воздушных судов в случае сбойных ситуаций. Рассмотрим типовой процесс принятия решения о корректировке расписания диспетчером ЦУП авиакомпании. Деятельность ЦУП регламентируется Воздушным кодексом РФ, Федеральными авиационными правилами, нормативными документами Уполномоченного органа РФ в области ГА, международными стандартами, а также внутренними документами авиакомпании.

Основные задачи ЦУП представлены далее.

1.Координация производственной деятельности для обеспечения целевых показателей в соответствии с политикой авиакомпании, определенной для реализации коммерческой целесообразности функционирования авиакомпании в рамках, установленных действующим законодательством в области гражданской авиации;

2. Реализация и мониторинг организационных мероприятий, направленных на выполнение суточного плана полетов и утвержденного расписания полетов воздушных судов авиакомпании;

3. Координация действий структурных подразделений авиакомпании, участвующих в организационном обеспечении полетов, организации наземного обслуживания ВС, отслеживании местоположения воздушных судов, контроле организации обслуживания пассажиров, обеспечении соблюдения стандартов авиакомпании в области качества и безопасности полетов;

4. Передача исходных данных для планирования и контроля оборота летных экипажей, бортпроводников и включаемого в задание на полет персонала.

ЦУП, как правильно, включает в себя ряд структурных подразделений, одним из которых является оперативное управление. Задача оперативного управления ЦУП является организация контроля и обеспечение оперативного управления производственной деятельностью подразделений авиакомпании для выполнения программы полетов авиакомпании и рейсов заказчиков на период до 72 часов.

Основные задачи оперативного управления полетами:

- организация и контроль выполнения плана полетов на период до 72 часов от текущего времени;
- отслеживание местоположения воздушных судов и своевременная оперативная корректировка плана полетов;
- координация взаимодействия структурных подразделений и организаций, участвующих в подготовке воздушных судов и организации наземного обслуживания;
- контроль обеспечения регулярности полетов;
- анализ и согласование в установленном порядке оперативных решений о замене типов (модификаций) воздушных судов;
- определение причин, классификация задержек отправления рейсов.

Ключевым инструментом для работы диспетчера оперативного управления ЦУП является производственная система, в которой выполняется функция ведения расписания. На рисунке 5.1. представлен интерфейс производственной системы авиакомпании, на котором отображены цепочки рейсов, выполняемым различными бортами. Вертикальной линией обозначено текущее время, отрезками обозначены рейсы. В случае наличия задержки рейса на прилет или на вылет, в интерфейсе отображается красным цветом время задержки.

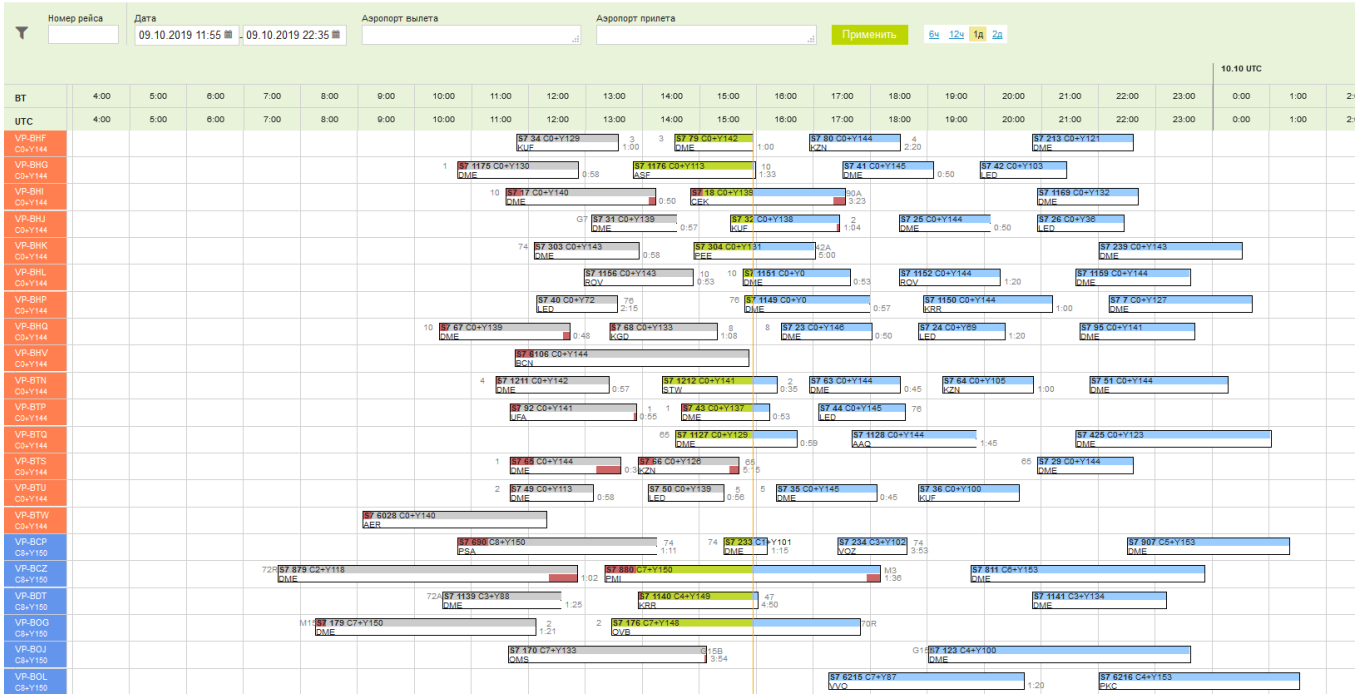


Рисунок 5.1 – Интерфейс производственной системы авиакомпании

Сотрудник ЦУП анализирует выполнение графика движения ВС и в случае возникновения длительной задержки принимает решение о перестановке бортов, основываясь на собственной экспертизе, а также на экспертном анализе возможных вариантов изменения назначений.

Разработанный прототип СППР призван облегчить процесс принятия решений о корректировке расписания, а также повысить его эффективность.

5.2. Архитектура и описание программного комплекса

Под архитектурой системы, согласно стандарта ГОСТ Р 57100-2016/ISO/IEC/IEEE42010:2011, понимают «основные понятия или свойства

системы в окружающей среде, воплощенной в ее элементах, отношениях и конкретных принципах ее проекта и развития» [81].

Функционально прототип системы поддержки принятия решений состоит из программы расчета задержек и корректировки расписания, включающей в себя интерфейс пользователя, модуль расчета задержек и оптимизатора (рисунок 5.2). Программа расчета задержек и корректировки расписания реализована на языке C# в интегрированной среде разработки Microsoft Visual Studio (исходные коды представлены в Приложении В). Оптимизатор реализован на языке OPL с использованием расширений языка сценариев IBM ILOG Script в интегрированной среде разработки IBM ILOG CPLEX Optimization Studio (Исходные коды представлены в Приложении Г).

Рисунок 5.2 – Общая функциональная структура прототипа системы поддержки принятия решений о переназначении ВС

Входными данными для работы СППР являются таблица с актуальным расписанием и справочник полетного времени.

На первом этапе пользователь выполняет подготовку данных для оптимизатора, путем загрузки исходного расписания, а также справочников полетного времени и минимальных времен наземного обслуживания в модуль

расчета задержек. Далее пользователь должен выбрать схему декомпозиции исходного расписания, определить количество итераций, указать диапазоны входных данных и произвести запуск пошагового алгоритма оптимизации назначений ВС на рейсы. На заключительном этапе с помощью модуля корректировки расписания производится объединение назначений ВС на рейсы с целью формирования результирующего расписания для внесения в производственную систему авиакомпании.

В таблице 5.1 представлено актуальное расписание. Здесь каждая строка обозначает конкретный полетный сегмент.

Таблица 5.1 – Пример расписания для работы выгрузки в СППР

RECID	1278541996	1278544632	1278546496	1278549396
Carrier	S7	S7	S7	S7
FlightNo	34	79	80	213
DEP	KUF	DME	KZN	DME
ARR	DME	KZN	DME	GSV
AIRCRAFT	319	319	319	319
BORT	VP-BHF	VP-BHF	VP-BHF	VP-BHF
STD	09.10.2019 11:55	09.10.2019 14:35	09.10.2019 16:55	09.10.2019 20:50
STA	09.10.2019 13:40	09.10.2019 16:10	09.10.2019 18:30	09.10.2019 22:35
ATD	09.10.2019 11:47	09.10.2019 14:33	NULL	NULL
ATA	09.10.2019 13:33	NULL	NULL	NULL
MTT	45	40	45	50
BLKPlan	105	95	95	105
CurrentDelay	NULL	80	0	0
FlightStatus	Завершен	В полете	Ожидает вылета	Ожидает вылета

Приведенное в таблице 5.1 расписание можно представить в виде цепочки рейсов (рисунок 5.3).

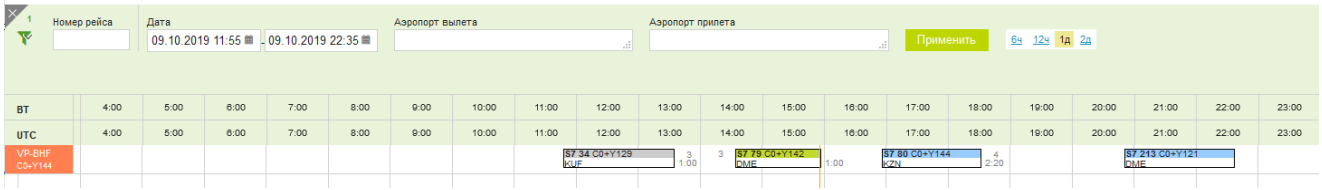


Рисунок 5.3 – Пример визуализации расписания в виде цепочки рейсов

Содержательное описание входного массива данных для СППР приведено в таблице 5.2.

Таблица 5.2 – Описание полей входного массива данных для СППР

Атрибут	**Расшифровка**	**Содержательное описание**
RECID	-	Уникальный идентификатор записи в производственное системе авиакомпании.
Carrier	IATA-код авиакомпании	Двухсимвольный уникальный код авиакомпании (например, SU – аэрофлот)
FlightNo	Номер рейса	Уникальный числовой идентификатор рейса.
DEP	IATA-код аэропорта вылета	Уникальный идентификатор аэропорта Трёхбуквенный уникальный идентификатор, присваиваемый IATA. Например, KZN – Казань.
ARR	IATA-код аэропорта прилета	
AIRCRAFT	Тип ВС	Трехсимвольный код типа ВС, присваиваемый IATA (например, 320 – Airbus 320). Обозначает категорию, объединяющую ВС по ряду технико-экономическим характеристикам (дальность полета, количество пассажиров, схемы рассадки, период эксплуатации и др.).
BORT	Бортовой номер ВС	Уникальный идентификатор воздушного судна (например, VP-BQS). Присваивается заводом-изготовителем ВС. По бортовому номеру можно проследить историю полетов.
STD	Schedule Time of Departure	Время отправления рейса по расписанию. Время, установленное для начала движения ВС со стоянки. Согласовывается в аэропорту отправления, как слот для отправления и указывается в системах бронирования и авиабилете.
STA	Schedule Time of Arrival	Время прибытия рейса по расписанию. Время, установленное для остановки ВС на месте стоянки. Согласовывается в аэропорту прибытия, как слот для прибытия.

ATD	Actual Time of Departure	Фактическое время отправления рейса из аэропорта отправления (отправление с места стоянки). Указывается только для рейсов, которые уже вылетели. Для остальных сегментов принимает значение NULL.
ATA	Actual Time of Arrival	Фактическое время прибытия рейса в аэропорт назначения. Указывается только для рейсов которые уже прилетели. Для остальных сегментов принимает значение NULL.
MTT	Minimum Turnaround Time	Минимальное время разворота в аэропорту прибытия, которое требуется для обслуживания ВС в аэропорту оперирования. Предоставляется в минутах.
BLKPlan	Полетное время по плану	Это время от отправления рейса по расписанию до прибытия рейса по расписанию, предоставляется в минутах.
CurrentDelay	Текущая задержка рейса	Задержка рейса на момент выгрузки расписания, т.е. в момент принятия решения по переназначению ВС. Для уже выполненных рейсов принимает значение NULL.
FlightStatus	Статус рейса	Может принимать три значения: в полете (ATD<>NULL и ATA=NULL), ожидает вылета (ATD = NULL), выполнен (ATA <> NULL).

Для решения задачи корректировки графиков движения ВС также требуется информация о всевозможных временах полета между аэропортами оперирования. Данная информация особо важна в случае наличия длительной задержки, когда может оказаться целесообразным выполнить перегон ВС из одного аэропорта в другой.

В таблицах 5.3 и 5.4 представлены пример справочника полетного времени и содержательное описание его атрибутов.

Таблица 5.3 – Справочник полетного времени.

STN1	STN2	AC	Months	Year	FromTo	ToFrom
AAQ	DME	319	10	2019	130	135
AAQ	DME	320	10	2019	130	140
AAQ	DME	321	10	2019	130	135
AAQ	DME	32N	10	2019	130	135
AAQ	DME	32Q	10	2019	130	140
AAQ	DME	738	10	2019	130	140

Таблица 5.4 – Описание атрибутов справочника полетного времени

Атрибут	Содержательное описание
STN1	IATA-код первого аэропорта
STN2	IATA-код второго аэропорта
AC	Тип ВС
Months	Месяц и год, на которые задается полетное время. В различное время года полетное время на одном и том же направлении может отличаться ввиду изменения розы ветров, аэронавигационных маршрутов и т.д.
Year	
FromTo_min	Полетное время из первого аэропорта во второй аэропорта в минутах
ToFrom_min	Полетное время из второго аэропорта в первый аэропорт в минутах

Как следует из архитектурной схемы прототипа СППР, на первом этапе пользователь выполняет подготовку данных для оптимизатора, путем загрузки исходного расписания в программу расчета задержек, далее выполняется поэтапная корректировка расписания с помощью оптимизатора. На заключительном этапе с помощью основной программы производится объединение назначения ВС на рейсы с целью формирования результирующего расписания.

5.2.1. Начало работы с программой расчета задержек и корректировки расписания

При входе в систему пользователь видит диалоговое окно, представленное на рисунке 5.4, на которо отображаются доступные функции:

- расчет задержек;
- пересчет расписания на определенное время;
- корректировка расписания с учетом результатов работы оптимизатора;
- объединение назначений.

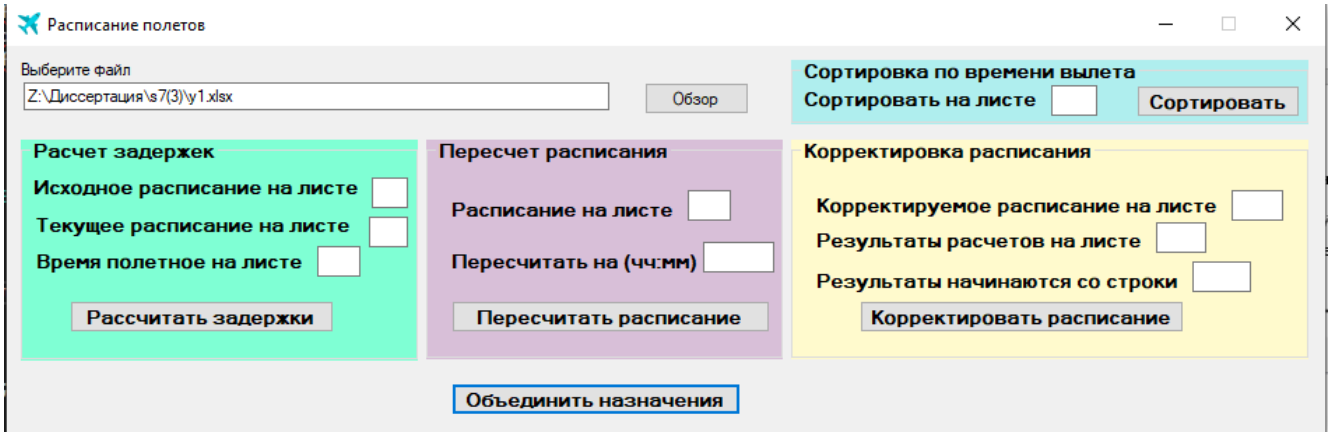


Рисунок 5.4 – Интерфейс программы расчета задержек и корректировки расписания движения воздушных судов авиакомпании.

Перед решением любой из задач необходимо выбрать файл, в который выгружены исходные данные, т.е. текущее расписание, требующее корректировки (на отдельном листе) и справочник полетного времени (на отдельном листе).

Сделать это можно с помощью кнопки «Обзор» или прописав путь к файлу в соответствующем поле вручную. Перед запуском задачи нужно убедиться, что выбранный файл закрыт. Если в ходе работы произошел какой-либо сбой, и на экран не был выведен файл Excel с результатами работы, желательно с помощью диспетчера задач удалить из списка процессов процесс “Microsoft Excel”. Если в памяти несколько таких процессов, нужно удалить их все.

5.2.2. Расчет задержек

Для расчета задержек необходимо указать порядковый номер листа (рисунок 5.5), на котором находится текущее расписание (таблица 5.5), требующее корректировки, и порядковый номер листа, на котором расположен справочник полетного времени.

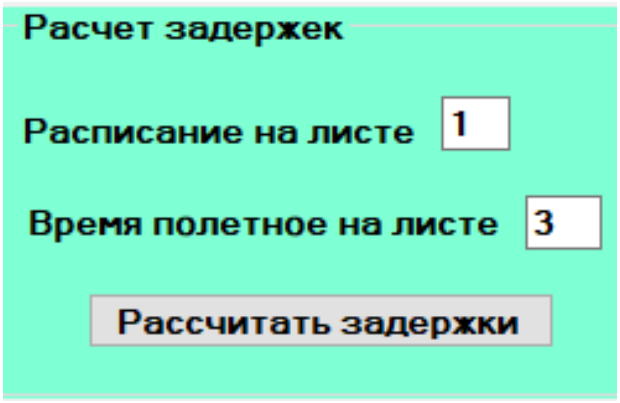


Рисунок 5.5 – Пример заполнения формы для расчета задержек

Таблица 5.5 – Пример входного файла для расчета задержек в СППР

REC_ID	Carrier	FlightNo	DEP	ARR	AIRCRAFT	BORT	STD	STA	ATD	ATA	MTT	BLK_Plan	CurrentDelay	FlightStatus
1	S7	5021	OVB	SVX	E70	VQ-BYH	13:05	15:30	13:43	-	35	145	30	В полете
2	S7	5022	SVX	OVB	E70	VQ-BYH	16:10	18:40	-	-	40	150	0	Ожидает вылета
3	S7	5355	OVB	TLK	E70	VQ-BYH	20:40	23:30	-	-	60	170	0	Ожидает вылета
4	S7	5336	NUX	OVB	E70	VQ-BYL	10:15	12:30	10:14	12:16	40	135	-	Завершен
5	S7	5037	OVB	KZN	E70	VQ-BYL	13:30	17:00	13:25	-	40	210	145	В полете
6	S7	5038	KZN	OVB	E70	VQ-BYL	17:45	20:55	-	-	40	190	0	Ожидает вылета
7	S7	5040	PEE	OVB	E70	VQ-BYM	11:45	14:30	11:43	14:27	40	165	-	Завершен
8	S7	5051	OVB	GOJ	E70	VQ-BYM	15:10	19:25	15:23	-	30	255	45	В полете
9	S7	5052	GOJ	OVB	E70	VQ-BYM	20:10	23:55	-	-	40	225	0	Ожидает вылета
10	S7	5232	IKT	OVB	E70	VQ-BYN	10:40	13:25	10:29	13:07	40	165	-	Завершен
11	S7	5231	OVB	IKT	E70	VQ-BYN	14:20	16:50	14:18	-	45	150	95	В полете

После нажатия кнопки «Рассчитать задержки» на экран будет выведен файл Excel со сформированным листом «Задержки» (таблица 5.6). Если лист с таким названием существовал и ранее, необходимо до запуска задачи его удалить или переименовать. Лист «Задержки» содержит ряд массивов, последовательно следующих друг за другом, структура и смысловое содержание которых описаны

далее. Пользователь должен сохранить файл с рассчитанными задержками на диске.

Таблица 5.6 – Лист «Задержки». Матрица полетного времени

Направление	**SVX -OVB**	**OVB -TLK**	**KZN -OVB**	**GOJ -OVB**
Рейс / Борт	**5022**	**5355**	**5038**	**5052**
VQ-BYH	150	170	190	220
VQ-BYL	150	170	190	220
VQ-BYM	150	170	190	220
VQ-BYN	150	170	190	220

В матрице полетного времени представлена информация о времени выполнения каждого рейса в расписании каждым бортом. В случае, если выполнение рейса невозможно на каком-либо типе ВС (например, ввиду отсутствия у аэропорта отправления или прибытия допуском на обслуживание заданного типа ВС), то для всех бортов данного типа ВС на данном направлении в качестве полетного времени будет указано большое число.

Таблица 5.7 – Лист «Задержки». Матрица полетного времени

Направление	**SVX -OVB**	**OVB -TLK**	**KZN -OVB**	**GOJ -OVB**
Рейс / Борт	**5022**	**5355**	**5038**	**5052**
VQ-BYH	970	1390	10000	10000
VQ-BYL	10000	1470	1065	10000
VQ-BYM	10000	1508	10000	1218
VQ-BYN	1160	1285	1320	1495

Далее на листе «задержки» приводится расчет задержек для каждого рейса при выполнении каждым ВС (таблица 5.7). В случаях, если борт ВС не может быть назначен на рейс в матрице проставляется большое число. Это случаи, когда, например, для выполнения рейса требуется выполнить перегон ВС из другого аэропорта вне расписания, т.е. с нулевой загрузкой и, как следствие, отрицательной эксплуатационной прибылью.

Дополнительно на листе «Задержки» представляется служебная информация с указанием порядкового номера рейса после упорядочения по задержкам.

Таблица 5.8 – Лист «Задержки». Порядок следования рейсов

Направление	**SVX -OVB**	**OVB -TLK**	**KZN -OVB**	**GOJ -OVB**
Рейс / Борт	**5022**	**5355**	**5038**	**5052**
VQ-BYH	8	11	14	17
VQ-BYL	8	11	14	17
VQ-BYM	7	8	11	17
VQ-BYN	1	2	3	4

Ключевым информационным блоком на листе «Задержки» является сводная информация по задержкам каждого рейса (Таблица 5.9). Здесь «SHED» - это плановое расписание вылетов, «korr» - скорректированные расписание вылетов, а «objective» - отклонение расчетного расписания от планового, которое должно быть минимизировано согласно условию (2.26). В дополнение отображены все расчетные задержки для каждого рейса и борта ВС.

Таблица 5.9 – Лист «Задержки». Расчет задержек и параметров расписания

Направление	**SVX -OVB**	**OVB -TLK**	**KZN -OVB**	**GOJ -OVB**
Рейс / Борт	**5022**	**5355**	**5038**	**5052**
SHED	970	1240	1065	1210
objective	0	45	0	8
korr	970	1285	1065	1218
VQ-BYH	970	0	0	0
VQ-BYL	0	0	1065	0
VQ-BYM	0	0	0	1218
VQ-BYN	0	1285	0	0

Далее представлена матрица назначений бортов на рейсы (таблица 5.10), где 1 означает выполнение рейса на данном ВС. Строка «objective» содержит признак наличия назначения ВС на рейс, а также позволяет выполнить контроль выполнения ограничения (2.19).

Таблица 5.10 – Лист «Задержки». Матрица назначений ВС на рейсы

Направление	**SVX -OVB**	**OVB -TLK**	**KZN -OVB**	**GOJ -OVB**
Рейс / Борт	**5022**	**5355**	**5038**	**5052**
objective	1	1	1	1

VQ-BYH	1	0	0	0
VQ-BYL	0	0	1	0
VQ-BYM	0	0	0	1
VQ-BYN	0	1	0	0

5.2.3. Реализация декомпозиционного алгоритма A_V с использованием lp оптимизатора

После того, как в программе расчёта задержек сформировался файл с рассчитанными задержками, для решения текущей подзадачи lp (3.1)-(3.6) требуется запустить модуль линейного программирования оптимизатора IBM CPLEX. IBM CPLEX Optimization Studio – набор инструментов с поддержкой аналитических решений, включающие в себя математическое программирование и программирование в ограничениях, благодаря чему осуществима быстрая разработка и развертывание моделей оптимизации. Программное средство состоит из интегрированной среды разработки (IDE), мощного языка программирования для оптимизаций (OPL), высокопроизводительных модулей решения CPLEX и CP оптимизаторов. OPL является языком моделирования для комбинаторной оптимизации, направленной на упрощение решения задач оптимизации. Среда IDE также предоставляет прямой доступ из OPL к внедренным реализациям алгоритмов линейного, целочисленного и квадратичного программирования [82].

Перед запуском оптимизатора пользователь должен выбрать схему декомпозиции исходных данных задачи оптимизации (3.15)-(3.22) с определением количества итераций по корректировке расписания и диапазонов входных данных для модулей lp оптимизатора CPLEX. В вычислительных экспериментах применялось несколько простых схем декомпозиции с равномерным разбиением исходного множества рейсов на подмножества, порождавшим от 3-х до 5-и шагов алгоритма A_V. При запуске оптимизатора пользователь также должен указать имя сохраненного файла (формат xls) с рассчитанными задержками, наименование листа и диапазонов, в которых размещены входные данные для задачи (3.15)-(3.22).

Исходные коды OPL-модели и файла данных для оптимизатора CPLEX, реализующих пошагово алгоритм A_v, представлены в приложении В.

Результаты работы ПО после обращения к модулям CPLEX оптимизатора выводятся на новую страницу файла с исходными данными. Наименование новой страницы по умолчанию - «Корректировки» может изменяться пользователем.

Пользователь выполняет представленный пересчет и корректировку расписания в соответствии с A_v, столько раз, сколько итераций было определено при выборе схемы декомпозиции.

5.2.4. Формирование результирующего скорректированного расписания

На заключительном этапе пересчета, для того, чтобы получить конечное расписание, скорректированное с учетом текущих задержек, пользователь должен воспользоаться функцией «Объединить назначения». На Рисунке 4.6 представлен фрагмент программного интерфейса и пример применения этой функции после пяти последовательных шагов декомпозиционного алгоритма.

Формируемые сводные данные по назначениям и задержкам после всех шагов алгоритма отображаются на странице «Свод» того же файла.

Интеграция программы подготовки данных для расчета задержек и корректировки расписания движения воздушных судов авиакомпании с модулями оптимизатора CPLEX осуществляется посредством разработанного проекта OPL, позволяющего загрузить и выполнить расчет оптимальных назначений и задержек на каждом шаге декомпозиционного алгоритма оперативного оптимального регулирования расписания. В дальнейшей работе планируется реализовать интегрированное решение, где модуль оптимизации будет встроен в общую архитектуру СППР и дополнительный действий со стороны пользователя в ЦУП не потребуется.

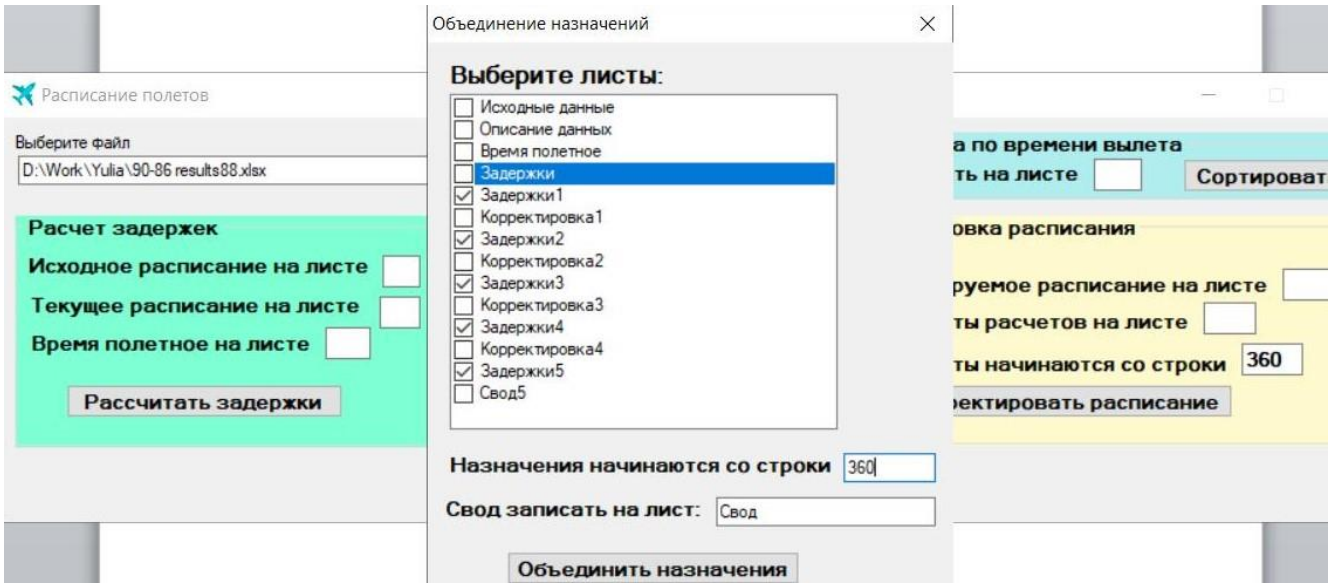


Рисунок 5.6 – Пример формы для объединения расчитанных назначений

5.2.5. Корректировка расписания с учетом результатов работы оптимизатора

Для корректировки расписания необходимо указать (рисунок 5.7), на каких листах находится расписание полетов и результаты работы оптимизатора, а также, с какой строки на листе результатов начинается матрица назначений ВС на рейсы (таблица 5.10).

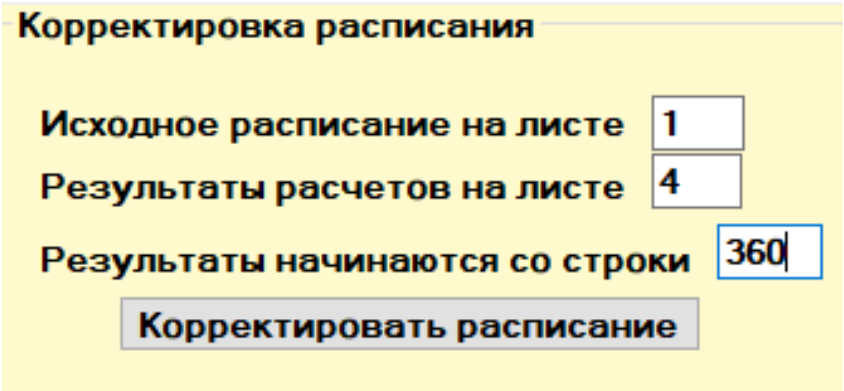


Рисунок 5.7 – Пример заполнения формы для корректировки расписания

После нажатия кнопки «Корректировать расписание» на экран будет выведен файл Excel со сформированным листом «Корректировка» (рисунок 5.7). Если лист с таким названием существовал и раньше, нужно до запуска задачи его удалить или переименовать.

Для приведенного в таблице 5.5 фрагмента расписания лист «Корректировки» может выглядеть следующим образом (таблица 5.11).

Таблица 5.11 – Лист «Корректировка»

REC_ID	Carrier	FlightNo	DEP	ARR	AIRCRAFT	BORT	STD	STA	ATD	ATA	MTT	BLK_Plan	CurrentDelay	FlightStatus
1	S7	5021	OVB	SVX	E70	VQ-BYH	13:05	15:30	13:43	-	35	145	30	В полете
2	S7	5022	SVX	OVB	E70	VQ-BYH	16:10	18:40	-	-	40	150	0	Переназначен/Ожидает вылета
3	S7	5355	OVB	TLK	E70	VQ-BYN	21:25	0:15	-	-	60	170	0	Переназначен/Ожидает вылета
4	S7	5336	NUX	OVB	E70	VQ-BYL	10:15	12:30	10:14	12:16	40	135	-	Завершен
5	S7	5037	OVB	KZN	E70	VQ-BYL	13:30	17:00	13:25	-	40	210	145	В полете
6	S7	5038	KZN	OVB	E70	VQ-BYL	17:45	20:55	-	-	40	190	0	Переназначен/Ожидает вылета
7	S7	5040	PEE	OVB	E70	VQ-BYM	11:45	14:30	11:43	14:27	40	165	-	Завершен
8	S7	5051	OVB	GOJ	E70	VQ-BYM	15:10	19:25	15:23	-	30	255	45	В полете
9	S7	5052	GOJ	OVB	E70	VQ-BYM	20:10	23:55	-	-	40	225	0	Переназначен/Ожидает вылета
10	S7	5232	IKT	OVB	E70	VQ-BYN	10:40	13:25	10:29	13:07	40	165	-	Завершен
11	S7	5231	OVB	IKT	E70	VQ-BYN	14:20	16:50	14:18	-	45	150	95	В полете

Статус «переназначен» присваивается рейсу после любого шага алгоритма. В случае, если изменилось назначение борта ВС и/или время отправления/прибытия, в таблице будут внесены корректировки в соответствующие поля (см. выделенные ячейки). Если назначение борта не изменилось после шага алгоритма, значит борт переназначен на тот же рейс, что и в исходном расписании. В таком случае программа изменит только статус рейса на «Переназначен/Ожидает вылета», рейсы с таким статусом из дальнейших шагов алгоритма исключаются.

Общее время выполнения корректировки расписания с использованием разработанного протоипа не превышает 4 минут с учетом трудозатрат на загрузку файлов в оптимизатор и сборку результатов. Время подготовки данных для работы оптимизатора (быстродействие программы расчета задержек и корректировки расписания) составляет 1,5 минуты. Время работы оптимизатор – не более 1,5 секунд. Полученные оценки быстродействия позволяют сделать вывод о

применимости предложенного инструментария для принятия решения в реальном масштабе времени при условии доработки прототипа в части подготовки данных для оптимизатора.

5.2.6. Пересчет расписания на определенное время

Для пересчета расписания на определенное время необходимо указать (рисунок 5.8), на каком листе находится расписание полетов, и на какой момент времени нужно пересчитать расписание. Время вводится в формате «чч:мм».

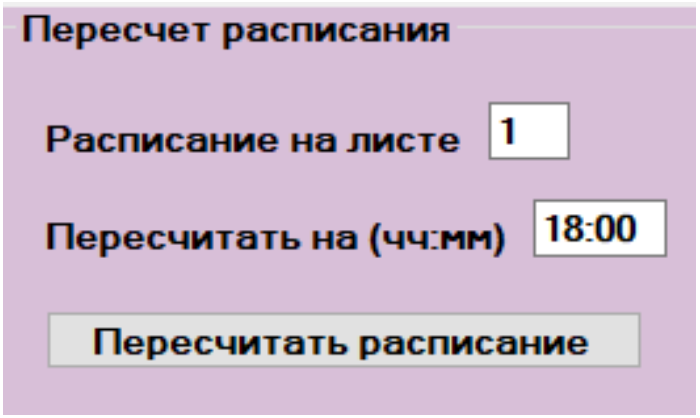


Рисунок 5.8 – Пример заполнения формы для пересчета расписания на определенное время (18:00)

После нажатия кнопки «Пересчитать расписание» на экран будет выведен файл Excel со сформированным листом «18-00». Если лист с таким названием существовал и раньше, нужно до запуска задачи его удалить или переименовать.

5.3. Выводы

В пятой главе данной научной работы было представлено описание программной реализации прототипа интеллектуальной СППР для диспетчеров оперативного управления ЦУП авиакомпании. Разработанный прототип СППР базируется на представленном во второй и третьей главах подходе, основанном на декомпозиции, линейной релаксации и быстром поэтапном решении, переводящим задачу в класс полиномиально разрешимых.

Разработанный программный инструментарий прошел государственную регистрацию в качестве программ для ЭВМ, о чём Федеральной службой по интеллектуальной собственности выдано свидетельство № 2020663452 от 28 октября 2020 г (Приложение Д.1) и ЭВМ свидетельство № 2021665658 от 12 октября 2021 г (Приложение Д.2).

Апробации использованных в прототипе СППР моделей и методов подтверждается актами о внедрении результатов исследования (Приложение Е).

Прототип СППР направлена на повышение скорости принятия решения, а также на повышение эффективности принимаемого решения в отношении графиков движения ВС. Оценка результатов использования прототипа СППР показала, что его применение в сравнении с ранее используемым подходом, основанном на применении экспертизы конкретного диспетчера оперативного управления ЦУП, позволяет сократить задержки ≈58% путем реализации более устойчивого расписания. Как следствие, применение СППР минимизирует расходы и репетиционные риски, связанные с задержками.

ЗАКЛЮЧЕНИЕ

Проведенное в рамках научной работы исследование представляет собой теоретическую и практическую базу разработанного нового подхода к поддержке процесса оперативного управления расписаниями авиакомпаний.

Выполнено структурированное описание предмета исследования, разработана формальная постановка задачи управления назначениями воздушных судов, относящейся к классу задач теории расписаний, а именно – к задачам оптимизации расписаний параллельно-последовательных систем с задержками начала обслуживания и неопределенными маршрутами обслуживания.

Выполнен глубокий анализ существующих подходов к решению задачи управления назначениями воздушных судов и обоснование актуальности разработки новых методов и алгоритмов решения. Показана принадлежность описываемой технологической системы числу параллельно-последовательных систем, порождающих наиболее трудоемкие задачи теории расписаний класса NP.

Разработан новый вычислительно эффективный декомпозиционный алгоритм, позволяющий перевести задачу в класс полиномиально разрешимых и пригодный для решения задач сколь угодно больших размерностей.

Предложен принципиально новый критерий для оценки эффективности полученного расписания, основанный на применении риск-ориентированного подхода.

Разработан прототип системы поддержки принятия решений, реализующий представленные методы управления назначениями ВС. Использование прототипа позволяет сократить время на принятие решения по корректировке назначений до нескольких секунд.

Выполнено экспериментальное обоснование эффективности применение предложенных технических решений, согласно которому время потенциальных задержек может быть сокращено на 50%, а время на принятие решения о корректировке расписания – до нескольких секунд

Определены перспективы развития исследования по теме.

Разработанные методы направлены на повышение скорости принятия решения по оперативной корректировке графика движения ВС в сбойной ситуации, а также на повышение эффективности принимаемого решения в отношении графиков движения. Применение предложенного подхода позволит авиакомпаниям минимизировать отклонения расписания от запланированных параметров и, тем самым, минимизировать потенциальные расходы, связанные с задержками рейсов и обслуживанием пассажиров в сбойных ситуациях.

Дальнейшая разработка представленного в настоящей научной работе подхода может быть направлена на модификацию предложенной постановки в части перечня производственных ограничений, а также исследование наиболее эффективного принципа декомпозиции ввиду выявленного влияния на получаемые результаты.

СПИСОК СОКРАЩЕНИЙ И УСЛОВНЫХ ОБОЗНАЧЕНИЙ

AEV – Adaptive Evaluation Vector

ARR – Airport of Arrival

ATA – Actual Time of Arrival

ATD – Actual Time of Departure

ASR – Aircraft Selection Heuristic

BLK – Block Time

CAPY – Seating capacity of Economy class

CAPC – Seating capacity of Business class

DEP – Airport of Departure

DSS – Disruption Set Solver

ETD – Estimated Time of Departure

FAM – Flight Assignment Model

FIFO – First Input First Output

FLT – Flight Time

FLTNO – Flight Number

GRASP – General Responsibility Assignment Software Patterns

IATA – International Air Transport Association

LOC – Local Time

MTT – Minimum Turnaround Time

MILP – Задача частично-целочисленного линейного программирования

NPS – Net Promoter Score

SSIM – Standard Schedules Information Manual

STD – Scheduled Time of Departure

STA – Scheduled Time of Arrival

UTC - Coordinated Universal Time

АБОВ - алгоритм бинарных отсечений и ветвлений

АО – Акционерное Общество

ВПП – Взлетно-Посадочная Полоса

ВС – Воздушное Судно

ГА – Гражданская Авиация

ДО – Дискретная Оптимизация

ЛП – Линейное программирование

ЛПР – Лицо, Принимающее Решение

МНК – метод Наименьших Квадратов

НО – Наземное Обслуживание

ПОС – Параллельная обслуживающая система

ППОС – Параллельно-Последовательная Обслуживающая Система

ППС – Позднее Прибытие Самолета

СППР – Система Поддержки Принятия Решений

ТО – Техническое Обслуживание

ЦУП – Центр Управления Полетами

ФАС – Федеральная Антимонопольная Служба

СПИСОК ЛИТЕРАТУРЫ

1. Rushmeier, R. A. Recent advances in exact optimization of airline scheduling problems / R. A. Rushmeier, K. L. Hoffman, M. Padberg. – New York: New York University, 1995. – 41 p.

2. Мезенцев, Ю. А. Задача и инструменты оптимального регулирования расписаний флота авиакомпании / Ю. А. Мезенцев, Ю. Л. Короткова, И. В. Эстрайх // Информационные технологии. – 2020. – Т. 26, № 8. – С. 450-459. – ISSN 1684-6400. – DOI 10.17587/it.26.450-459.

3. Avdeenko, T. V. Heuristic approach to unrelated parallel machines scheduling under availability and resource constraints / T. V. Avdeenko, Y. A. Mezentsev, I. V. Estraikh // IFAC-PapersOnline. – 2017. – Vol. 50, Is. 1. – P. 13096-13101. – ISSN 2405-8963/ – DOI 10.1016/j.ifacol.2017.08.1998.

4. Decision support framework for airline flight cancellations and delays / A. I. Z. Jarrah, G. Yu, N. Krishnamurthy, A. Rakshit // Transportation science. – 1993. – Vol. 27 (3). – P. 266-280. – ISSN 0041-1655. – DOI 10.1287/trsc.27.3.266.

5. Cao, J. M. Real-time decision support for integration of airline flight cancellations and delays. Part I : mathematical formulation / J. M. Cao, A. Kanafi // Transportation planning and technology. – 1997. – Vol. 20. – P. 183-199. – ISSN 1029-0354. – DOI 10.1080/03081069708717588.

6. Talluri, K. T. Swapping applications in a daily airline fleet assignment / K. T. Talluri // Transportation science. – 1996. – Vol. 30 (3). – P. 237-248. – ISSN 0041-1655. – DOI 10.1287/trsc.30.3.237.

7. Yan, S. A decision support framework for handling schedule perturbation / S. Yan, D-H. Yang // Transportation research. – 1996. – Vol. 30 (6). – P. 405-419. – ISSN 2352-1465. – DOI 10.1016/0191-2615(96)00013-6.

8. Yan, S. Multifleet routing and multistop flight scheduling for schedule perturbation / S. Yan, Y. P. Tu // European journal of operational research. – 1996. – Vol. 103. – P. 155-169. – ISSN 0377-2217. – DOI 10.1016/S0377-2217(96)00260-3.

9. Lou, S. On the airline schedule perturbation problem caused by the ground delay program / S. Lou, G. Yu // Transportation science. – 1997. – Vol. 31 (4). – P. 298-311. – ISSN 0041-1655. – DOI 10.1287/trsc.31.4.298.

10. Arguello, M. F. A GRASP for aircraft routing in response to grounding and delays / M. F. Arguello, J. F. Bard, G. Yu // Journal of combinatorial optimization. – 1997. – Vol. 5. – P. 211-228. – ISSN 1382-6905. – DOI 10.1023/A:1009772208981.

11. Bard, J. F. Optimizing aircraft routings in response to groundings and delays / J. F. Bard, G. Yu, M. F. Arguello // IEE Transactions. – 2000. – Vol. 33. – P. 931-947. – ISSN 0018-9383. – DOI 10.1023/A:1010987008497.

12. Thengvall, B. G. Balancing user preferences for aircraft schedule recovery during irregular operations / B. G. Thengvall, J. F. Bard, G. Yu // IIE Transactions. – 2000. – Vol. 32. – P. 181-193. – ISSN 0018-9383. – DOI 10.1023/A:1007618928820.

13. Rosenberger, J. M. Rerouting aircraft for airline recovery / J. M. Rosenberger, E. L. Johnson, G. L. Nemhauser // Transportation science. – 2003. – Vol. 37 (4). – P. 408-421. – ISSN 0041-1655. – DOI 10.1287/trsc.37.4.408.23271.

14. Andersson, T. The flight perturbation problem / T. Andersson, P. Varbrand // Transportation planning and technology. – 2004. – Vol. 27 (2). – P. 91-118. – ISSN 1029-0354. – DOI 10.1080/0308106042000218195.

15. Eggenberg, N. Constraint-specific recovery network for solving airline recovery problems / N. Eggenberg, M. Salani, M. Bierlaire // Computers and operations research. – 2010. – Vol. 37. – P. 1014-1026. – ISSN 0305-0548. – DOI 10.1016/j.cor.2009.08.006.

16. Jafari, N. Simultaneous recovery model for aircraft and passengers / N. Jafari, S. H. Zegordi // Journal of the franklin institute. – 2010. – Vol. 348 (7). – P. 20-22. – ISSN 0016-0032. – DOI 10.1016/j.jfranklin.2010.03.012.

17. A large neighbourhood search heuristic for the aircraft and passenger recovery problem / S. Bisaillon, J.-F. Cordeau, G. Laporte, F. Pasin // 4OR-A quarterly journal of operations research. – 2011. – Vol. 9. – Is. 2. – P. 139-157. – ISSN 1614-2411. – DOI 10.1007/s10288-010-0145-5.

18. Sinclair, K. Improvements to a large neighborhood search heuristic for an integrated aircraft and passenger recovery problem / K. Sinclair, J.-F. Cordeau, G. Laporte // European journal of operational research. – 2014. – Vol. 233, Is. 1. – P. 234-245. – ISSN 0377-2217. – DOI 10.1016/j.ejor.2013.08.034.

19. Sinclair, K. A column generation post-optimization heuristic for the integrated aircraft and passenger recovery problem / K. Sinclair, J.-F. Cordeau, G. Laporte // Computers & operations research. – 2016. – Vol. 65. – P. 42-52. – ISSN 0305-0548. – DOI 10.1016/j.cor.2015.06.014.

20. The study on the aircraft recovery with consideration of passenger transiting / Y. Hu, X. Baoguang, M. Gao, H. Chi // The Fifth international conference on management science and engineering management. – Macau, China: World Academic Press, 2011. – P. 3-7.

21. Yang, T. Considering passenger preferences in integrated postdisruption recoveries of aircraft and passengers / T. Yang, Y. Hu // Mathematical problems in engineering. – 2019. – Vol. (9). – P. 1-19. – ISSN 1024-123X. – DOI 10.1155/2019/9523610.

22. Mariani, C. Disruption management in the airline industry : master thesis / C. Mariani // Term Paperware House : website. – URL: https://www.termpaperwarehouse.com/essay-on/Disruption-Management-in-the-Airline-Industry/480935 (date of application: 01.12.2021).

23. Worldwide Scheduling Guidelines / International Air Transport Association // Montreal, Geneva, 2005. – 77 p. // World Wide Airport Coordinators Group : website. – URL: http://www.wwacg.org/up/files/docsWSG/WORLWIDE_SCHEDULING_GUIDELINES/WSG_12th%20Ed.pdf_040309_032834.pdf (date of application: 01.12.2021).

24. Abara, J. Applying integer linear programming to the fleet assignment problem / J. Abara // Interfaces. – 1989. – Vol. 19 (4). – P. 20-28. – ISSN 0920-5489. – DOI 10.1287/inte.19.4.20.

25. Coldstart : fleet assignment at delta air lines / R. Subramanian [et al.] // Interfaces. – 1994. – Vol. 24 (1). – P. 104-120. – ISSN 0920-5489. – DOI 10.1287/inte.24.1.104.

26. Павлова, Л. В. Моделирование расстановки парка ВС на рейсы авиакомпании / Л. В. Павлова // Научный Вестник МГТУ ГА. – 2016. – Т 19, № 5. – С. 186-192. – ISSN 2079-0619.

27. Васильев, Ю. М. Решение задачи равномерного разбиения рейсов летного расписания авиакомпании: точная математическая постановка / Ю. М. Васильев, С. В. Уният, Г. М. Фридман // Известия СПбГЭУ. – 2016. – № 3 (99). – С. 68-74. – ISSN 2311-3464.

28. Петрунин, С. В. Методология управления использованием воздушных судов в российских авиакомпаниях: дис. ... докт. техн. наук: 05.02.22 / Петрунин Станислав Владимирович ; Моск. гос. техн. ун-т гражд. авиации. – М., 2009. – 243 с.

29. Ageeva, Y. Approaches to incorporating robustness into airline scheduling / Y. Ageeva // Master thesis operations research center, massachusetts institute of technology. – Massachusetts: Massachusetts institute of technology, 2000. – P. 93-94.

30. Liang, Z. The Aircraft maintenance routing problem / Z. Liang, W. A. Chaovalitwongse // Optimization and logistics challenges in the enterprise. – 2009. – Vol. 30. – P. 327-348. – DOI 10.1007/978-0-387-88617-6_12.

31. The aircraft rotation problem / L.W. Clarke [et al.] // Annals of operations research. – 1997. – Vol. 69. – ISSN 0254-5330. – DOI 10.1023/A:1018945415148.

32. Техническое обслуживание самолета // Сервис AVIADO : сайт. – URL: https://aviado.ru/guide/planes/checks/ (дата обращения: 01.12.2021).

33. Lan, S. Planning for robust airline operations: optimizing aircraft routings and flight departure times to minimize passenger disruptions / S. Lan, J. P. Clark, C. Barnhart // Transportation science. – 2006. – Vol. 40 (1). – ISSN 0041-1655. – DOI 10.1287/trsc.1050.0134.

34. Воздушный кодекс Российской Федерации : текст с изменениями на 14 марта 2022 г. : федеральный закон от 19 марта 1997 г. № 60-ФЗ [принят

Государственной Думой 19 февраля 1997 года ; одобрен Советом Федерации 5 марта 1997 года] // Собрание законодательства Российской Федерации. – 1997. – № 12. – Ст. 1383.

35. Об утверждении положения об особенностях режима рабочего времени и времени отдыха членов экипажей воздушных судов гражданской авиации Российской Федерации : текст с изменениями на 17 сентября 2010 г. : приказ Минтранса Российской Федерации от 21 ноября 2005 г. № 139 // Бюллетень нормативных актов федеральных органов исполнительной власти. – 2006. – № 6.

36. Recent advances in Crew Pairing optimization at American Airlines / R. Anbil, E. Gelman, B. Patty, R. Tanga // Interfaces. – 1991. – Vol. 21 (1). – ISSN 0920-5489. – P. 32-74.

37. Carlotta, M. Disruption management in the airline industry: master thesis. / M. Carlotta // NTNU Open : website. – URL: https://ntnuopen.ntnu.no/ntnu-xmlui/bitstream/handle/11250/2351128/13986_FULLTEXT.pdf?sequence=1&isAllowed=y (date of application: 01.12.2021).

38. Мезенцев, Ю. А. Оптимизация расписаний параллельных динамических систем в календарном планировании / Ю. А. Мезенцев // Информационные технологии. – 2008. – № 2. – С. 24-33.

39. Мезенцев, Ю. А. Эффективные вычислительные методы решения дискретных задач оптимизации управления производственными процессами / Ю. А. Мезенцев. – Новосибирск : Издательство НГТУ, 2015. – 274 с.

40. Мезенцев, Ю. А. Оптимизация расписаний параллельно-последовательных систем в календарном планировании / Ю. А. Мезенцев // Информационные технологии. – 2009. – № 6. – С. 35-41. – ISSN 1684-6400.

41. Мезенцев, Ю. А. Прикладные задачи и алгоритмы оптимизации расписаний параллельных обслуживающих систем / Ю. А. Мезенцев // Научный вестник Новосибирского государственного технического университета. – 2016. – № 1 (62). – С. 49-73. – ISSN 1814-1196.

42. Hassan, L. K. Airline disruption management: a literature review and practical challenges / L. K. Hassan, B. F. Santos, J. Vink // Computers & operations research. – 2021. – Vol. 127. – ISSN 0305-0548. – DOI 10.1016/j.cor.2020.105137.

43. Teodorovic, D. Optimal dispatching strategy on an airline network after a schedule perturbation / D. Teodorovic, S. Guberinić // European journal of operational research. – 1984. – Vol. 15, Is. 2. – P. 178-182. – ISSN 0377-2217.

44. Teodorovic, D. Model for operational daily airline scheduling / D. Teodorovic, G. Stojkovic // Transportation planning and technology. – 1990. – Vol. 14. – P. 273- 285. – ISSN 1029-0354.

45. Teodorović, D. Model for operational daily airline scheduling / D. Teodorović, G. Stojković // Transportation planning and technology. – 1995. – Vol. 121 (4). – P. 324-331. – ISSN 1029-0354.

46. Rakshit, A. System operations advisor: a real-time decision support system for managing airline operations at United Airlines / A. Rakshit, N. Krishnamurthy, G. Yu // Interfaces. – 1996. – Vol. 26. – Is. 2. – P. 50-58. – ISSN 0920-5489. – DOI 10.1287/inte.26.2.50.

47. Qiang, G. Research on greedy simulated annealing algorithm for irregular flight schedule recovery model / G. Qiang, X. W. Tang, J. F. Zhu // 2009 IEEE International conference on Grey Systems and Intelligent Services, GSIS 2009. – 2009. – P. 1469-1475. – DOI 10.1109/GSIS.2009.5408145.

48. Liu, T-K. Optimization of short-haul aircraft schedule recovery problems using a hybrid multiobjective genetic algorithm / T-K. Liu, J-H. Chou, C-H. Chen // Expert systems with applications. – 2010. – Vol. 37 (3). – P. 2307-2315. – ISSN 0957-4174. – DOI 10.1016/j.eswa.2009.07.068.

49. Congcong, W. A New approach to solve aircraft recovery problem / W. Congcong, M. Le // Proceedings of the second international conference on advanced communications and computation (INFO- COMP 2012). – IARIA, 2012. – P. 148-154.

50. Zhao, X. Study on GRAPS-ACO algorithm for irregular flight rescheduling / X. Zhao, Y. Guo // International conference on computer science and service system, IEEE. – 2012. – P. 266-269. – DOI 10.1109/CSSS.2012.74.

51. Gao, M. Solving the airline recovery problem based on vehicle routing problem with time window modeling and genetic algorithm / M. Gao, M. Le, C. Zhan // Ninth international conference on natural computation (ICNC). – Shenyang, China, 2013. – DOI 10.1109/ICNC.2013.6818089.

52. Aktürk, M. S. Gürel aircraft rescheduling with cruise speed control / M. S. Aktürk, A. Atamtürk, S. Gürel // Operations research. – 2014. – Vol. 62 (4). – P. 829-845. – DOI 10.1287/opre.2014.1279.

53. Brunner, J. O. Rescheduling of flights during ground delay programs with consideration of passenger and crew connections / J. O. Brunner // Transportation research. Part E Logistics and transportation review. – 2014. – Vol. 72. – P. 236-252. – DOI 10.1016/j.tre.2014.10.004.

54. A methodology combining optimization and simulation for real applications of the stochastic aircraft recovery problem / D. Guimarans, P. Arias, M. M. Mota [et al.] // Proceedings – 8th UROSIM Congress on modelling and simulation, EUROSIM. – 2013. – P. 265-270. – DOI 10.1109/EUROSIM.2013.55.

55. Vos, H. W. M. Aircraft schedule recovery problem – a dynamic modeling framework for daily operations / H. W. M. Vos, B. F. Santos, T. Omondi // Transportation research Procedia. – 2015. – Vol. 10. – P. 931-940. – ISSN 2352-1465. – DOI 10.1016/j.trpro.2015.09.047.

56. Airline disruption management – dynamic aircraft scheduling with ant colony optimization / H. Sousa, R. Teixeira, H. L. Cardoso, E. Oliveira // Proceedings of the International conference on agents and artificial intelligence. – 2015. – P. 398-405. – ISBN 978-989-758-073-4. – DOI 10.5220/0005205303980405.

57. Zhu, B. A Stochastic programming approach on aircraft recovery problem / B. Zhu, J.F. Zhu, Q. Gao // Mathematical problems in engineering. – 2015 (1). – P. 1-9. – ISSN 1024-123X. – DOI 10.1155/2015/680609.

58. The time-band approximation model on flight operations recovery model considering random flight flying time in China / H. Xu, S. Han, Y. Zhang [et al.] // Proceedings – 2015 IEEE International Conference on Systems, Man, and Cybernetics, SMC 2015. – 2015. – P. 695-700. – DOI 10.1109/SMC.2015.131.

59. Xu, H. Weighted time-band approximation model for flight operations recovery considering simplex group cycle approaches in China / H. Xu, S. Han // Mathematical problems in engineering. – 2016. – P. 1-17. – ISSN 1024-123X. – DOI 10.1155/2016/3201490.

60. Wu, Z. Solving multiple fleet airline disruption problems using a distributed-computation approach to integer programming / Z. Wu, B. Li, C. Dang // IEEE Access. – 2017. – Vol. 5. – P. 19116-19131. – ISSN 2169-3536. – DOI 10.1109/ACCESS.2017.2747155.

61. Multiple objective solution approaches for aircraft rerouting under the disruption of multi-aircraft / Y. Hu, H. Liao, S. Zhang, Y. Song // Expert systems with applications. – 2017. – Vol. 83. – P. 283-299. – ISSN 0957-4174. – DOI 10.1016/j.eswa.2017.04.031.

62. Zhang, C. Two-stage heuristic algorithm for aircraft recovery problem / C. Zhang // Discrete dynamics in nature and society. – 2017. – P. 1-12. – ISSN 1607-887X. – DOI 10.1155/2017/9575719.

63. A multi-criteria repair/recovery framework for the tail assignment problem in airlines / O. Khaled, M. Minoux, V. Mousseau [et al.] // Journal of air transport management. – 2017. – Vol. 68. – P. 137-151. – ISSN 0969-6997. – DOI 10.1016/j.jairtraman.2017.10.002.

64. Šarčević, T. Artificial Bee Colony algorithm for solving the flight disruption problem / T. Šarčević, A. Rocha, A. Castro // Communications in computer and information science. – 2018. – Vol. 887. – P. 72-81. – DOI 10.1007/978-3-319-94779-2_7.

65. A column generation-based heuristic for aircraft recovery problem with airport capacity constraints and maintenance flexibility / Z. Liang, F. Xiao, X. Qian, L. Zhou // Transportation research record journal of the transportation research board. – 2018. – Vol. 113. – P. 70-90. – ISSN 0361-1981. – DOI 10.1016/j.trb.2018.05.007.

66. Zhao, T. A weight-table based heuristic algorithm for aircraft recovery problem. Proceedings of the 37th Chinese control conference / T. Zhao, X. Chen //

Technical committee on control theory, Chinese Association of Automation. – Wuhan, China, 2018. – P. 2242-2246.

67. Lin, H. Flight scheduling for airport closure based on sequential decision / H. Lin, Zh. Wang // 4th International Conference on information management (ICIM). – 2018. – P. 241-245. – DOI 10.1109/INFOMAN.2018.8392843.

68. Airline disruption recovery using symbiotic simulation and multi-fidelity modelling / L. Rhodes-Leader, D. J. Worthington, B. L. Nelson // Proceedings of the operational research society simulation workshop 2018. – SW, 2018. – P. 146-155.

69. Flight schedule recovery : a simulation-based approach / D. Wang, Y. Wu, J. Hu, M. Liu // Asia pacific journal of operational research. – 2019. – Vol. 36 (06). – P. 1940010. – DOI 10.1142/S0217595919400104.

70. Grönkvist, M. The tail assignment problem. PhD thesis, Chalmers University of Technology and Göteborg University / M. Grönkvist. – Göteborg, Sweden, 2005. – 296 p.

71. Applying the quantum approximate optimization algorithm to the tail-assignment problem / P. Vikstal, M. Gronkvist, M. Svensson [et al.] // Physical review applied. – 2020. – Vol. 14(3). – ISSN 2331-7019. – DOI 10.1103/PhysRevApplied.14.034009.

72. Lee, J. Dynamic disruption management in airline networks under airport operating uncertainty / J. Lee, L. Marla, A. Jacquillat // Transport science. – 2020. – Vol. 54 (4). – P. 973-997. – ISSN 0361-1981. – DOI 10.1287/trsc.2020.0983.

73. Mezentsev, Y. A. An optimal fleet assignment and flight scheduling problem for an airline company / Y. A. Mezentsev, I. V. Estraykh // CEUR Workshop proceedings. – 2018. – № 2098. – С. 277-290. – ISSN 1613-0073. – DOI 10.17212/1727-2769-2018-3-74-90.

74. Mezentsev, Y. A. Implementation of an efficient parametric algorithm for optimal scheduling on parallel machines with release dates / Y. A. Mezentsev, I. V. Estraykh, N. Y. Chubko // Journal of physics : conference series. – 2019. – Vol. 1333. – P. 1-7. – ISSN 1742-6596. – DOI 10.1088/1742-6596/1333/2/022002.

75. Симонян, Т. В. Современный метод измерения лояльности клиентов Net Promoter Score / Т. В. Симонян, М. В. Довгалева // Научный альманах. – 2016. – № 1-1 (15). – С. 267 – 272. – ISSN 2411-7609. – DOI 10.17117/na.2016.01.01.267.

76. ГОСТ Р ИСО 31000-2019. Менеджмент риска. Принципы и руководство. – Москва : Стандартинформ, 2020. – 19 с.

77. ГОСТ Р 58771-2019. Менеджмент риска. Технологии оценки риска. – Москва : Стандартинформ, 2020. – 90 с.

78. Коротченко, Е. А. Метод оценки рисков «Критерии. События. Правила» / Е. А. Коротченко, Ю. Л. Петрунина // International journal of open information technologies. – 2016. – Vol. 4. – № 5. – С. 52-58. – ISSN 2307-8162.

79. Коротченко, Е. А. Метод оценки рисков «Критерии. События. Правила» / Е. А. Коротченко, Ю. Л. Петрунина // Инновации в жизнь. – 2017. – № 2 (21) – С. 187-197. – ISSN 2227-6300.

80. Наумова, Д. А. Методики оценки регулярности полетов компаний / Д. А. Наумова // Научный Вестник МГТУ. – 2012. – № 187 (1). – С. 90-94. – ISSN 2079-0619.

81. ГОСТ Р ИСО 57100-2016. Системная и программная инженерия. Описание архитектуры. – Москва : Стандартинформ, 2019. – 36 с.

82. IBM ILOG CPLEX Optimization studio // IBM : website. – URL: https://www.ibm.com/products/ilog-cplex-optimization-studio (date of application: 01.12.2021).

83. Lin, Y. K. Unrelated parallel machine scheduling problem using an ant colony optimization approach / Y. K. Lin, H. T. Hsieh, F. Y. Hsieh // World Academy of science, engineering and technology. – 2012. – Vol. 6. – P. 1798-1803. – ISSN 2010-376X.

84. Lin, Y. K. Particle swarm optimization algorithm for unrelated parallel machine scheduling with release dates / Y. K. Lin // Mathematical problems in engineering. –2013. – Vol. 1-4. – ISSN 1024-123X. – DOI 10.1155/2013/409486.

85. Lee, W. C. A simulated annealing approach to makespan minimization on identical parallel machines/ W. C. Lee, C. C. Wu, P. Chen // International journal of

advanced manufacturing technology. – 2006. – Vol. 31, Is. 3-4. – P. 328-334. – ISSN 0268-3768.

86. Lenstra, J. K. Approximation algorithms for scheduling unrelated parallel machines / J. K. Lenstra, D. B. Shmoys, E. Tardos // Mathematical programming. – 1987. – Vol. 46 (1-3). – P. 217-224. – ISSN 0025-5610.

87. Kaabi, J. A Survey of parallel machine scheduling under availability constraints / J. Kaabi, Y. Harrath // International journal of computer and information technology. – 2014. – Vol. 03, Is. 02. – P. 238-245. – ISSN 2279-0764/

88. Mezentsev, Y. A. Binary Cut-and-Branch method for solving linear programming problems with boolean variables / Y. A. Mezentsev // CEUR Workshop proceedings. – 2016. – Vol. 1623. – P. 72-85. – ISSN 1613-0073.

89. Mezentsev, Y. Binary cut-and-branch method for solving mixed integer programming problems Constructive Nonsmooth Analysis and Related Topics (dedicated to the memory of V.F. Demyanov) / Y. Mezentsev. – CNSA, 2017. – DOI 10.1109/cnsa.2017.7973989.

90. Link to tests and best solutions achieved // Google.drive : website. – URL: https://drive.google.com/drive/folders/16ez7NQd1cPPIymf0ALYnJzgzquGBXHga?ogsrc=32 (date of application: 01.12.2021).

91. Vazirani, V. Approximation algorithms / V. Vazirani. – Springer, 2001. – 375 p. – ISBN 978-3-642-08469-0/

92. Avdeenko, T. V. Efficient approaches to scheduling for unrelated parallel machines with release dates / T. V. Avdeenko, Y. A. Mesentsev // IFAC conference on manufacturing modelling, management and control MIM. – 2016. – Vol. 49, Is. 12. – P. 8.

93. Mezentsev, Y. A. Problems and optimization algorithms of schedules of parallel-serial systems with undefined service routes / Y. A. Mezentsev, I. V. Estraikh // Abstracts of the International conference «Constructive Nonsmooth Analysis and Related Topics». Part II. – May 22-27 2017. – Saint-Petersburg : BBM, 2017. – P. 79-83. – ISBN 978-5-9651-1059-9.

94. Мезенцев, Ю. А. Математические задачи оптимального управления реализацией проектов: монография / Ю. А. Мезенцев. – Новосибирск : Изд-во НГТУ, 2013. – 146 с. – ISBN 978-5-7782-2276-2.

95. Мезенцев, Ю. А. Эффективный алгоритм решения прикладной задачи оптимизации расписаний параллельно-последовательной системы/ Ю. А. Мезенцев, Ю. Л. Короткова, И. В. Эстрайх // Информационные технологии. – 2021. – Т. 27, № 12. – С. 642-650. – ISSN 1684-6400. – DOI 10.17587/it.27.642-650.

96. Korotkova, Y. L. Application Problem and Effective Algorithm of the Parallel-Sequential System Schedule Optimization / Y. L. Korotkova, Y. A. Mezentsev // XV International Scientific-Technical Conference on Actual Problems Of Electronic Instrument Engineering (APEIE). – Novosibirsk : IEEE, 2021. – 6 p. – DOI: 10.1109/APEIE52976.2021.9647516.

97. Мезенцев, Ю. А. Риск-ориентированный подход к решению задачи оперативного управления расписанием авиакомпании / Ю. А. Мезенцев, Ю. Л. Короткова // Системы анализа и обработки данных. – 2021. – № 4 (84). – С. 19-36. – ISSN 2782-2001. – DOI 10.17212/2782-2001-2021-4-19-36.

СПИСОК ИЛЛЮСТРИРОВАННОГО МАТЕРИАЛА

Перечень рисунков

Перечень таблиц

ПРИЛОЖЕНИЕ А

```
/*********************************************
 * OPL 12.9 Model
 * Author:
 * Creation Date: 12.07.2020 at 19:07:39
 *********************************************/
int NumberOfMachines = ...;           //число приборов
int NumberOfRequests = ...;           //число заявок
range NOM = 1..NumberOfMachines;      //описание диапазона изменения индекса прибора
range NOR = 1..NumberOfRequests;      //описание диапазона изменения индекса заявки
int ServiceTime[NOR][NOM] = ...;    //матрица времени обслуживания заявки в зависимости от
выбранного прибора
int Delays[NOR] = ...;              //вектор задержек поступления заявок
//dvar int+ lambda;
dvar float lambda;
//dvar int+ Assignments[NOR][NOM];      //матрица распределения заявок по приборам
dvar boolean Assignments[NOR][NOM];
int MinMaxAssignmentsOnEachMachine[NOM][1..2]=...; //матрица значений минимального и
максимального количества заявок на каждый прибор
//dvar float+ U[NOM][2..NumberOfRequests][1..NumberOfRequests-1]; //Uijk
dvar boolean U[NOM][2..NumberOfRequests][1..NumberOfRequests-1];
//dvar int+ Y[NOM][2..NumberOfRequests]; //Yij
dvar float Y[NOM][2..NumberOfRequests];
minimize (lambda);
subject to
   {forall(j in NOR)
      sum(i in NOM)
        Assignments[j][i]==1;
   forall(i in NOM)
        forall(j in 2..NumberOfRequests)
          forall(k in 1..NumberOfRequests-1)
            U[i][j][k]<=1;

    forall(i in NOM)
      sum(j in NOR)
  Assignments[j][i]>=MinMaxAssignmentsOnEachMachine[i][1];
    forall(i in NOM)
      sum(j in NOR)
  Assignments[j][i]<=MinMaxAssignmentsOnEachMachine[i][2];

    forall(i in NOM)
     forall(j in 2..NumberOfRequests)
       forall(k in 1..j-1)
         Assignments[j][i]+Assignments[k][i]-
         (sum(l in k+1..j-1) Assignments[l][i])-(j-k+1)*U[i][j][k]>=-j+k+1;
    forall(i in NOM)
     forall(j in 2..NumberOfRequests)
       forall(k in 1..j-1)
         Assignments[j][i]+Assignments[k][i]-
         (sum(l in k+1..j-1) Assignments[l][i])-(j-k+1)*U[i][j][k]<=1;

    forall(i in NOM)
     forall(j in 2..NumberOfRequests)
       Y[i][j]>=0;
    forall(i in NOM)
```

```
      forall(j in 2..NumberOfRequests)
       (sum(k in 1..j-1) (Delays[k]+ServiceTime[k][i])*U[i][j][k])-Y[i][j]<=Delays[j];

    forall(i in NOM)
     (sum(j in NOR) (Delays[j]+ServiceTime[j][i])*Assignments[j][i])
      +(sum(l in 2..NumberOfRequests) Y[i][l])
      -(sum(j in NOR)
         sum(k in 1..j-1) (Delays[k]+ServiceTime[k][i])*U[i][j][k])<=lambda;
  }
main
{
  thisOplModel.generate();
  cplex.solve();
  var produce = thisOplModel;
  var ofile = new IloOplOutputFile("решение.txt");
  var NofM=produce.NumberOfMachines;
  var NofR=produce.NumberOfRequests;
  writeln("Распределение заявок по станкам");
  ofile.writeln("Распределение заявок по станкам");
  writeln("Станки");
  ofile.writeln("Станки");
  write("   ");
  ofile.write("   ");
  for (var i=1; i<=NofM; i++)
      {write(i," ");
       ofile.write(i," ");
      }
  writeln("");
  ofile.writeln("");
  for (var j=1; j<=NofR; j++)
     {
      write(j,") ");
      ofile.write(j,") ");
      for (i=1; i<=NofM; i++)
         {write(produce.Assignments[j][i]," ");
          ofile.write(produce.Assignments[j][i]," ");
         }
        writeln("");
        ofile.writeln("");
      }
      writeln("");
      ofile.writeln("");
  writeln('Время завершения последней операции равно ', produce.lambda);
  ofile.writeln('Время завершения последней операции равно ', produce.lambda);

}
```

Файлы исходных данных

```
/*********************************************
 * OPL 12.9 Data
 * Author:
 * Creation Date: 19.09.2019 at 11:55:21
 *********************************************/
NumberOfMachines = 5;
NumberOfRequests = 20;
ServiceTime = [
  [5, 6, 4, 6, 5]
```

```
    [4, 5, 5, 5, 4]
    [3, 4, 5, 4, 3]
    [4, 5, 3, 5, 4]
    [4, 4, 3, 4, 5]
    [5, 3, 4, 3, 4]
    [4, 4, 5, 4, 6]
    [6, 7, 6, 7, 5]
    [6, 8, 7, 8, 6]
    [7, 8, 8, 8, 9]
    [11, 9, 9, 9, 8]
    [8, 10, 9, 10, 8]
    [9, 10, 9, 10, 11]
    [7, 10, 10, 10, 11]
    [8, 8, 8, 8, 7]
    [7, 8, 9, 8, 7]
    [10, 9, 10, 9, 11]
    [10, 10, 10, 10, 9]
    [8, 8, 9, 8, 9]
    [8, 9, 9, 8, 8]
];

Delays = [0, 0, 1, 2, 5, 5, 5, 7, 8, 10, 10, 10, 12, 14, 15, 15, 18, 18, 18, 20];
MinMaxAssignmentsOnEachMachine = [
    [0, 15]
    [0, 15]
    [0, 15]
    [0, 15]
    [0, 15]
];

/*********************************************
 * OPL 12.9 Data
 * Author:
 * Creation Date: 17.07.2019 at 11:55:21
 *********************************************/
NumberOfMachines = 5;
NumberOfRequests = 40;
ServiceTime = [
    [5, 6, 4, 6, 5]
    [4, 5, 5, 5, 4]
    [3, 4, 5, 4, 3]
    [4, 5, 3, 5, 4]
    [4, 4, 3, 4, 5]
    [5, 3, 4, 3, 4]
    [4, 4, 5, 4, 6]
    [6, 7, 6, 7, 5]
    [6, 8, 7, 8, 6]
    [7, 8, 8, 8, 9]
    [11, 9, 9, 9, 8]
    [8, 10, 9, 10, 8]
    [9, 10, 9, 10, 11]
    [7, 10, 10, 10, 11]
    [8, 8, 8, 8, 7]
    [7, 8, 9, 8, 7]
    [10, 9, 10, 9, 11]
    [10, 10, 10, 10, 9]
    [8, 8, 9, 8, 9]
```

```
    [8, 9, 9, 8, 8]
    [5, 6, 4, 6, 5]
    [4, 5, 5, 5, 4]
    [3, 4, 5, 4, 3]
    [4, 5, 3, 5, 4]
    [4, 4, 3, 4, 5]
    [5, 3, 4, 3, 4]
    [4, 4, 5, 4, 6]
    [6, 7, 6, 7, 5]
    [6, 8, 7, 8, 6]
    [7, 8, 8, 8, 9]
    [11, 9, 9, 9, 8]
    [8, 10, 9, 10, 8]
    [9, 10, 9, 10, 11]
    [7, 10, 10, 10, 11]
    [8, 8, 8, 8, 7]
    [7, 8, 9, 8, 7]
    [10, 9, 10, 9, 11]
    [10, 10, 10, 10, 9]
    [8, 8, 9, 8, 9]
    [8, 9, 9, 8, 8]
];
Delays = [0, 0, 1, 2, 5, 5, 5, 7, 8, 10, 10, 10, 12, 14, 15, 15, 18, 18, 18, 20, 21, 21, 21, 22, 23, 23, 23, 24, 26, 26,
26, 27, 27, 27, 28, 28, 28, 29, 30, 30];

MinMaxAssignmentsOnEachMachine = [
    [0, 20]
    [0, 15]
    [0, 20]
    [0, 15]
    [0, 20]
];
```

Решение1: (5 приборов, 20 заявок)

Распределение заявок по приборам

Приборы

```
  1 2 3 4 5
1) 0 0 0 1 0
2) 0 0 0 0 1
3) 0 1 0 0 0
4) 0 0 1 0 0
5) 0 0 0 0 1
6) 0 1 0 0 0
7) 1 0 0 0 0
8) 0 0 1 0 0
9) 1 0 0 0 0
10) 0 0 0 1 0
11) 0 0 0 0 1
12) 0 1 0 0 0
13) 0 0 1 0 0
14) 1 0 0 0 0
15) 0 0 0 0 1
16) 0 0 0 0 1
17) 0 1 0 0 0
```

18) 1 0 0 0 0
19) 0 0 0 1 0
20) 0 0 1 0 0
Время завершения последней операции равно 32

Время счета 1,5 часа. Возможное отклонение от оптимума не более 12%.

Решение2: (5 приборов, 40 заявок)

Распределение заявок по приборам

Приборы
1 2 3 4 5
1) 0 0 0 0 1
2) 0 1 0 0 0
3) 0 0 0 1 0
4) 0 0 1 0 0
5) 1 0 0 0 0
6) 0 0 0 1 0
7) 0 1 0 0 0
8) 0 0 0 0 1
9) 0 0 1 0 0
10) 0 1 0 0 0
11) 0 0 0 0 1
12) 1 0 0 0 0
13) 0 0 1 0 0
14) 0 0 0 1 0
15) 0 0 0 0 1
16) 0 0 0 0 1
17) 0 0 0 1 0
18) 0 1 0 0 0
19) 1 0 0 0 0
20) 0 0 0 1 0
21) 0 0 1 0 0
22) 1 0 0 0 0
23) 0 0 0 0 1
24) 0 0 1 0 0
25) 0 0 1 0 0
26) 0 0 0 1 0
27) 0 1 0 0 0
28) 0 0 0 0 1
29) 1 0 0 0 0
30) 1 0 0 0 0
31) 0 1 0 0 0
32) 0 0 1 0 0
33) 0 0 1 0 0
34) 1 0 0 0 0
35) 0 0 0 0 1
36) 1 0 0 0 0
37) 0 1 0 0 0
38) 0 0 0 0 1
39) 0 1 0 0 0
40) 0 0 0 1 0
Время завершения последней операции равно 58. Наилучший результат 56 с гарантией не менее 51 получен после 14 часов счета

ПРИЛОЖЕНИЕ Б

Файл редукции модели с дизъюнкциями в ограничениях

```
/*********************************************
 * OPL 12.7.0.0 Model
 * Author: administrator
 * Creation Date: 25 февр. 2019 г. at 17:33:39
 *********************************************/
int I = ...;
int J = ...;
int j2;
int i2;
int jj;
range j = 1..J;
range i = 1..I;
range i1 = 1..I-1;
// range k = 1..I;
float Sh = 0.0;
float T[j][i] = ...;
float tau[j][i] = ...;
int num[j][i] = ...; // порядок следования заявок по возрастанию задержек
float shed[i] = ...;
// достигается сортировками tau по всем приборам
dvar boolean x[j][i];
 dvar float C[j][i];
 dvar float Cmax[j];
/*
minimize
sum( j2 in j ) Cmax[j2];

subject to {
    ct13:
  forall( jj in j )
   forall( r in i )
    C[jj][r]>= tau[jj][r]*x[jj][r];
//
    ct12:
   forall( jj in j )
     forall( r in i )
     C[jj][r] >= shed [r];

    ct1:
  forall( r in i )
        sum( jj in j )
       x[jj][r]   == 1;

    ct14:
  forall( jj in j )
     forall( r in i1 )

       C[jj][num[jj][r]]+T[jj][num[jj][r]]*x[jj][num[jj][r]]-C[jj][num[jj][r+1]]<=0.0;

     ct11:
  forall( jj in j )
         C[jj][num[jj][I]]+T[jj][num[jj][I]]*x[jj][num[jj][I]]-Cmax[jj]<=0.0;
}
```

Файл данных задачи с дизъюнкциями в ограничениях

```
/*********************************************
 * OPL 12.9.0.0 Data
 * Author: administrator
 * Creation Date: 25 февр. 2020 г. at 17:33:39
 *********************************************/
I = 90;  // I<=J  ?
J = 86;
SheetConnection sheet("90-86 results80.xlsx");

T from SheetRead(sheet,"Задержки18!b3:az88");

tau from SheetRead(sheet,"Задержки18!b91:az176");

num from SheetRead(sheet,"Задержки18!b179:az264");
//num from SheetRead(sheet,"Задержки16!b179:ci264");

shed from SheetRead(sheet,"Задержки18!b267:az267");
//shed from SheetRead(sheet,"Задержки16!b267:ci267");

betta to SheetWrite(sheet,"Задержки18!b269:az269");
//betta to SheetWrite(sheet,"Задержки16!b269:ci269");

C to SheetWrite(sheet,"Задержки18!b270:az355");
//C to SheetWrite(sheet,"Задержки16!b270:ci355");

x to SheetWrite(sheet,"Задержки18!b360:az445");
//x to SheetWrite(sheet,"Задержки16!b360:ci445");

delta to SheetWrite(sheet,"Задержки18!b446:b446");
//delta to SheetWrite(sheet,"Задержки16!b446:b446");
```

ПРИЛОЖЕНИЕ В

Файл декомпозиционной модели

```
/*********************************************
 * OPL 12.7.0.0 Model
 * Author: administrator
 * Creation Date: 25 февр. 2020 г. at 17:33:39
 *********************************************/
 // PARSshedUno применяется, когда:
 // рейсов меньше, чем ВС
 // рейсов не больше, чем ВС
int I = ...;
int J = ...;
int P = ...;
//int r;
//int i2;
//int jj;
range j = 1..J;
range i = 1..I;
//range i1 = 1..J;
range i1 = 1..I-1;
//float Sh = 0.0;
float T[j][i] = ...;
float tau[j][i] = ...;
int num[j][i] = ...; // порядок следования заявок по возрастанию задержек
float shed[i] = ...;
dvar float x[j][i] in 0..1;
dvar float+ C[j][i];
// dvar float+ Cmax[j];
 dvar float+ betta[i];
 dvar float+ delta;
/*
minimize

sum( r in i ) ( betta[r] - sum( jj in j )(x[jj][r])* shed[r]);
//minimize
// delta;

subject to {

    ct3:
  forall( jj in j )
   forall( r in i )
    x[jj][r]>= 0;

    ct7:
        sum( r in i )
         sum( jj in j )
           x[jj][r]   >= P;

    ct13:
  forall( jj in j )
   forall( r in i )
    C[jj][r]>= tau[jj][r]*x[jj][r];
//
    ct2:
```

```
  forall( jj in j )
        sum( r in i )
       x[jj][r]   <= 1;     //  в случае  , когда рейсов не больше, чем самолетов
 //  ct2 - назначение ВС
     ct1:
    forall( r in i )
//    forall( r in i1 )
        sum( jj in j )
       x[jj][r]   <= 1;  //  в случае  , когда рейсов меньше, чем самолетов
 //       x[jj][r]   <= 1;   //  в случае, когда рейсов не меньше, чем ВС
 // ct1 -  назначение рейсов
      ct9:
   forall( r in i )
        sum( jj in j )
       tau[jj][r]*x[jj][r]    <= betta[r];

      ct8:
   forall( r in i )
      betta[r] - shed[r]  <= delta;

}
```

Файл данных декомпозиционной модели

```
/*********************************************
 * OPL 12.9.0.0 Data
 * Author: administrator
 * Creation Date: 25 февр. 2021 г. at 17:33:39
 *********************************************/
//I = 90;  // первый шаг число рейсов
//I = 50; // второй шаг
I = 10;  // третий шаг
J = 85; // число ВС первый шаг
//J = 82;   // число ВС второй шаг
//P = 40; // число выбранных рейсов для переназначения первый и второй шаги
P = 10;   // число выбранных рейсов для переназначения третий шаг
//SheetConnection sheet("Pars100-5-1-new.xlsx");
SheetConnection sheet("90-86 results84.xlsx");

T from SheetRead(sheet,"Задержки3!b3:k87");
//T from SheetRead(sheet,"Задержки2!b3:ay87");
//T from SheetRead(sheet,"Задержки1!b3:cm87");

tau from SheetRead(sheet,"Задержки3!b90:k174");
//tau from SheetRead(sheet,"Задержки2!b90:ay174");
//tau from SheetRead(sheet,"Задержки1!b90:cm174");

num from SheetRead(sheet,"Задержки3!b177:k261");
//num from SheetRead(sheet,"Задержки2!b177:ay261");
//num from SheetRead(sheet,"Задержки1!b177:cm261");

shed from SheetRead(sheet,"Задержки3!b264:k264");
//shed from SheetRead(sheet,"Задержки2!b264:ay264");
//shed from SheetRead(sheet,"Задержки1!b264:cm264");

betta to SheetWrite(sheet,"Задержки3!b268:k268");
//betta to SheetWrite(sheet,"Задержки2!b268:ay268");
//betta to SheetWrite(sheet,"Задержки1!b268:cm268");
```

```
C to SheetWrite(sheet,"Задержки3!b270:k354");
//C to SheetWrite(sheet,"Задержки2!b270:ay354");
//C to SheetWrite(sheet,"Задержки1!b270:cm354");

x to SheetWrite(sheet,"Задержки3!b360:k444");
//x to SheetWrite(sheet,"Задержки2!b360:ay444");
//x to SheetWrite(sheet,"Задержки1!b360:cm444");

delta to SheetWrite(sheet,"Задержки3!b446:b446");
//delta to SheetWrite(sheet,"Задержки2!b446:b446");
//delta to SheetWrite(sheet,"Задержки1!b446:b446");
```

ПРИЛОЖЕНИЕ Г

Исходные коды программы расчета задержек

Program.cs – точка входа в приложение

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Windows.Forms;

namespace s7
{
   static class Program
   {
      /// <summary>
      /// Главная точка входа для приложения.
      /// </summary>
      [STAThread]
      static void Main()
      {
         Application.EnableVisualStyles();
         Application.SetCompatibleTextRenderingDefault(false);
         Application.Run(new Form1());
      }
   }
}
```

S7main.cs – главная форма приложения

```
using System;
using System.Collections.Generic;
using System.Windows.Forms;
using System.Globalization;


namespace s7
{
   public partial class Form1 : Form
   {
      const int TIME_INFINITY = 10000;
      ExcelApp exapp;
      public Form1()
      {
         InitializeComponent();
         openFileDialog1.Filter = "Text Files|*.xlsx;*.xls";
      }
```

```
private void button1_Click(object sender, EventArgs e)
{
   if (openFileDialog1.ShowDialog() == System.Windows.Forms.DialogResult.OK)
      textBox1.Text = openFileDialog1.FileName;
}

private void button2_Click(object sender, EventArgs e)
{
   if (textBox1.Text == "")
   {
      MessageBox.Show("Не задано имя файла");
      return;
   }

   if (textBox4.Text == "")
   {
      MessageBox.Show("Не указан номер листа с расписанием");
      return;
   }

   if (textBox5.Text == "")
   {
      MessageBox.Show("Не указано время");
      return;
   }

   int from = Convert.ToInt32(textBox4.Text);
   string for_time = textBox5.Text;
   int column_count = 15;
   exapp = new ExcelApp(textBox1.Text, column_count);
   string[] s = new string[column_count];
   flight fl;

   exapp.SetActiveSheet(from);
   int last_row = exapp.GetLastRow();

   if (exapp.IsSheetExist(for_time.Replace(':', '-')))
   {
```

```
            MessageBox.Show("Расчет на это время уже сделан.\n Для повторения расчета удалите или переименуйте лист в файле");
            exapp.Quit();
            return;
        }

        int to = exapp.AddNewPage(for_time.Replace(':', '-'));
        exapp.CopyCells("A1:Q" + last_row.ToString(), from, "A1", to);

        exapp.PrepareReadSheet(to, 1);
        while (exapp.ReadNextShedRow(s))
        {
            fl = new flight(s);
            bool a = fl.Change(for_time);
            exapp.WriteChanges(a, fl.StatusToString());
        }
        exapp.SetVisible();
        this.TopMost = true;
    }

    public void CreateAirPortList(List<string> AirPortList, List<flight_time> FlightTimeList)
    {
        foreach (flight_time fl in FlightTimeList)
            if (AirPortList.IndexOf(fl.STN1) < 0) AirPortList.Add(fl.STN1);

        foreach (flight_time fl in FlightTimeList)
            if (AirPortList.IndexOf(fl.STN2) < 0) AirPortList.Add(fl.STN2);
        AirPortList.Sort();
    }

    public void ExcelRead(List<flight> FlightList, int slist, List<flight_time> FlightTimeList, int tlist)
    {
        int column_count = 15;
        string[] s = new string[column_count];
        ExcelRead(FlightList, slist);
        //Считываем третий лист с третьей строки
        exapp.PrepareReadSheet(tlist, 2);

        while (exapp.ReadNextTimeRow(s))
```

```
            FlightTimeList.Add(new flight_time(s));
        }

        public void ExcelRead(List<flight> FlightList, int list)
        {
            int column_count = 15;
            exapp = new ExcelApp(textBox1.Text, column_count);
            string[] s = new string[column_count];

            exapp.PrepareReadSheet(list, 1);

            //Считываем первый лист со второйй строки
            while (exapp.ReadNextShedRow(s))
                FlightList.Add(new flight(s));
        }

        public void ExcelWrite2(List<flight> FlightList, List<flight_time> FlightTimeList)
        {
            List<flight> FlightJobs = new List<flight>();
            List<flight> FlightDevice = new List<flight>();
            List<string> Borts = new List<string>();

            int num_device = 0;
            int num_jobs = 0;

            if (exapp.IsSheetExist("Задержки"))
            {
                MessageBox.Show("Лист Задержки уже существует.\n Для повторения расчета удалите или переименуйте лист в файле");
                exapp.Quit();
                return;
            }

            exapp.PrepareReadSheet(exapp.AddNewPage("Задержки"), 1);

            //Выводим аэропорты по горизонтали
            exapp.SetActiveCell("B1");

            foreach (flight fl in FlightList)
```

```
        {
            if (/*!IsReassigned(fl.BORT, FlightList) &&*/ fl.FlightStatus == flight.FS.Wait && !fl.Duplicate)
            {
                if (fl.FlightStatus != flight.FS.Other) FlightJobs.Add(fl);
                exapp.SetValue(fl.DEPPORT + "-" + fl.ARRPORT);

                exapp.MoveDown(1);
                exapp.SetValue(fl.FlightNo.ToString());

                exapp.MoveRight(1);
                exapp.MoveUp(1);
                num_jobs++;
            }
        }

        foreach (flight fl in FlightList)
        {
            if (fl.FlightStatus == flight.FS.Other || !Borts.Contains(fl.BORT) && fl.Next < 0 /*&&
!fl.Duplicate*/)
            {
                FlightDevice.Add(fl);
                num_device++;
                Borts.Add(fl.BORT);
            }
        }

        int[] bs = BubbleSort(Borts);

        exapp.SetActiveCell("A3");
        for (int i = 1; i < 4; i++)
        {
            for (int j1 = 0; j1 < Borts.Count; j1++)
            {
                exapp.SetValue(Borts[bs[j1]]);
                exapp.MoveDown(1);
            }
            exapp.MoveDown(2);
        }
```

```
int[][] times, delay, sort_delay;
int[] time_from_midnight = new int[num_jobs];
times = new int[num_device][];
for (int i = 0; i < num_device; i++)
   times[i] = new int[num_jobs];

sort_delay = new int[num_device][];
for (int i = 0; i < num_device; i++)
   sort_delay[i] = new int[num_jobs];

delay = new int[num_device][];
for (int i = 0; i < num_device; i++)
   delay[i] = new int[num_jobs];

int d = 0;
int j = 0;
foreach (flight fl2 in FlightJobs)
{
   time_from_midnight[j] = TimeFromMidnight(fl2.STD);
   j++;
}

List<flight> Assignments = new List<flight>();
foreach (flight fl1 in FlightDevice)
{
   j = 0;
   IsReassigned(fl1.BORT, FlightList, Assignments);
   foreach (flight fl2 in FlightJobs)
   {
      times[d][j] = Find_Time(fl2.DEPPORT, fl2.ARRPORT, fl1.AIRCRAFT, FlightTimeList);
      if (times[d][j] == 0) times[d][j] = TIME_INFINITY;

      if (Assignments.Count > 0)
      {
         BubbleSort(Assignments);
         delay[d][j] = AssignmentsDelay(fl1, Assignments, FlightTimeList);
      }
      else
      {
```

```
            string dp;
            if (fl1.FlightStatus == flight.FS.Other) dp = fl1.DEPPORT;
            else dp = Find_CurrAirPort(fl1, FlightList);
            if (dp == fl2.DEPPORT) delay[d][j] = fl1.Delay + fl1.MTT;
            else
            {
               delay[d][j] = Find_Time(dp, fl2.DEPPORT, fl1.AIRCRAFT, FlightTimeList);
               if (delay[d][j] == 0) delay[d][j] = TIME_INFINITY;
               else delay[d][j] += fl1.Delay + fl1.MTT;
            }
         }
         if (delay[d][j] != TIME_INFINITY) delay[d][j] += time_from_midnight[j];
         if (fl1.BORT == fl2.BORT && delay[d][j] != TIME_INFINITY) delay[d][j] -= fl1.MTT;
         if (delay[d][j] < time_from_midnight[j]) delay[d][j] = time_from_midnight[j];
         j++;
      }
      d++;
   }

   exapp.SetActiveCell("A3");

   for (int i = 0; i < num_device; i++)
   {
      for (j = 0; j < num_jobs; j++)
      {
         exapp.MoveRight(1);
         exapp.SetValue(times[bs[i]][j].ToString());
      }
      exapp.MoveDown(1);
      exapp.MoveLeft(num_jobs);
   }

   exapp.MoveDown(2);

   for (int i = 0; i < num_device; i++)
   {
      for (j = 0; j < num_jobs; j++)
      {
         exapp.MoveRight(1);
```

```
            exapp.SetValue(delay[bs[i]][j].ToString());
        }
        exapp.MoveDown(1);
        exapp.MoveLeft(num_jobs);
    }

    for (int i = 0; i < num_device; i++)
        sort_delay[i] = BubbleSort(delay[i]);

    exapp.MoveDown(2);

    for (int i = 0; i < num_device; i++)
    {
        for (j = 0; j < num_jobs; j++)
        {
            exapp.MoveRight(1);
            exapp.SetValue(sort_delay[bs[i]][j].ToString());
        }
        exapp.MoveDown(1);
        exapp.MoveLeft(num_jobs);
    }

    exapp.MoveDown(2);

    for (j = 0; j < num_jobs; j++)
    {
        exapp.MoveRight(1);
        exapp.SetValue(time_from_midnight[j].ToString());
    }

    exapp.SetVisible();
    this.TopMost = true;
}

public int AssignmentsDelay(flight fl, List<flight> A, List<flight_time> FlightTimeList)
{
    int delay = TIME_INFINITY;

    if (A[0] > fl)
```

```
        {
            delay = FlightDelay(fl, A[0], FlightTimeList);
            if (delay != TIME_INFINITY) delay = fl.Delay + fl.MTT;
            return delay;
        }
        else
        {
            if (A.Count > 1)
                for (int i = 0; i < A.Count - 1; i++)
                    if (fl > A[i] && A[i + 1] > fl)
                    {
                        delay = FlightDelay(A[i], fl, FlightTimeList);
                        if (delay != TIME_INFINITY)
                            if (FlightDelay(fl, A[i + 1], FlightTimeList) == TIME_INFINITY) delay =
TIME_INFINITY;
                        if (delay != TIME_INFINITY) break;
                    }
        }

        if (delay == TIME_INFINITY && fl > A[A.Count - 1])
            delay = FlightDelay(A[A.Count - 1], fl, FlightTimeList);

        return delay;
    }

    public int FlightDelay(flight fl1, flight fl2, List<flight_time> FlightTimeList)
    {
        int delay;
        if (fl1.ARRPORT == fl2.DEPPORT) delay = fl1.Delay + fl1.MTT;
        else
        {
            delay = Find_Time(fl1.ARRPORT, fl2.DEPPORT, fl1.AIRCRAFT, FlightTimeList);
            if (delay == 0) delay = TIME_INFINITY;
            else delay += fl1.Delay + fl1.MTT;
        }

        //if (fl1.BORT == fl2.BORT && delay != TIME_INFINITY) delay -= fl1.MTT;
        if(delay + TimeFromMidnight(fl1.STA) > TimeFromMidnight(fl2.STD)) delay = TIME_INFINITY;
        if(delay > TIME_INFINITY) delay = TIME_INFINITY;
```

```
    return delay;
}
public int TimeFromMidnight(string s)
{
    int h = Convert.ToInt32(s.Substring(s.IndexOf(":") - 2, 2));
    int m = Convert.ToInt32(s.Substring(s.IndexOf(":") + 1, 2));
    return h * 60 + m;
}

private void BubbleSort(List<flight> s)
{
    flight temp;
    int len = s.Count;
    for (int i = 0; i < len - 1; i++)
    {
        for (int j = i + 1; j < len; j++)
        {
            if (s[i] > s[j])
            {
                temp = s[i];
                s[i] = s[j];
                s[j] = temp;
            }
        }
    }
}

private int[] BubbleSort(int[] s)
{
    int temp;
    int len = s.Length;
    int[] r = new int[len];
    for (int i = 0; i < len; i++)
        r[i] = i + 1;
    for (int i = 0; i < len - 1; i++)
    {
        for (int j = i + 1; j < len; j++)
        {
            if (s[i] > s[j] || s[i] == s[j] && r[i] > r[j])
```

```
            {
                temp = s[i];
                s[i] = s[j];
                s[j] = temp;
                temp = r[i];
                r[i] = r[j];
                r[j] = temp;
            }
        }
    }
    return r;
}

private int[] BubbleSort(List<string> Bort)
{
    string temp;
    int temp1;
    int len = Bort.Count;
    int[] r = new int[len];
    string[] s = new string[len];

    for (int i = 0; i < len; i++)
        s[i] = Bort[i];
    for (int i = 0; i < len; i++)
        r[i] = i;

    for (int i = 0; i < len - 1; i++)
    {
        for (int j = i + 1; j < len; j++)
        {
            if (s[i].CompareTo(s[j]) > 0)
            {
                temp = s[i];
                s[i] = s[j];
                s[j] = temp;
                temp1 = r[i];
                r[i] = r[j];
                r[j] = temp1;
            }
```

```
        }
    }
    return r;
}

private void button3_Click(object sender, EventArgs e)
{
    if (textBox1.Text == "")
    {
        MessageBox.Show("Не задано имя файла");
        return;
    }

    if (textBox9.Text == "")
    {
        MessageBox.Show("Не указан номер листа с исходным расписанием");
        return;
    }

    if (textBox2.Text == "")
    {
        MessageBox.Show("Не указан номер листа с текущим расписанием");
        return;
    }
    if (textBox3.Text == "")
    {
        MessageBox.Show("Не указан номер листа с полетным временем");
        return;
    }

    int nlist = Convert.ToInt32(textBox9.Text);
    int slist = Convert.ToInt32(textBox2.Text);
    int tlist = Convert.ToInt32(textBox3.Text);
    List<flight> FlightList = new List<flight>();
    List<flight_time> FlightTimeList = new List<flight_time>();

    ExcelRead(FlightList, slist, FlightTimeList, tlist);

    foreach (flight fl in FlightList)
```

```
{
   fl.Next = -1;
   fl.Previous = -1;
   fl.MTT_Dop = 0;
   fl.Duplicate = false;
}

Find_Duplicate_Entries(FlightList);

for (int i = FlightList.Count - 1; i > 0; i--)
   if (!FlightList[i].Duplicate) Find_Previous(i, FlightList);

foreach (flight fl in FlightList)
{
   if (fl.FlightStatus == flight.FS.Over || fl.FlightStatus == flight.FS.In)
      fl.Diff_Actual_Planned_Departure_Time();  //ATD-STD
   if (fl.FlightStatus == flight.FS.In)
      fl.Calc_Estimated_Arrival_Time(fl.ATD);  //ETA
   if (fl.FlightStatus == flight.FS.Wait)
   {
      int prev = fl.Previous;
      if (prev > -1)
      {
         if (FlightList[prev].FlightStatus == flight.FS.Over)
            fl.Calc_Estimated_Departure_Time(FlightList[prev].ATA);
         else fl.Calc_Estimated_Departure_Time(FlightList[prev].ETA);
         fl.Calc_Estimated_Arrival_Time(fl.ETD);  //ETA
         fl.Diff_Estimated_Planned_Departure_Time();
      }
   }
}

if (nlist != slist)
{
   List<flight> FlightList_Old = new List<flight>();
   ExcelRead(FlightList_Old, nlist);
   foreach (flight fl in FlightList_Old)
   {
      if (!IsBortInList(fl.BORT, FlightList))
```

```
            {
                fl.FlightStatus = flight.FS.Other;
                fl.Delay = 0;
                fl.DEPPORT = Find_CurrAirPort(fl, FlightList_Old);
                FlightList.Add(fl);
            }
        }
    }

    ExcelWrite(slist, FlightList);
    ExcelWrite2(FlightList, FlightTimeList);

    return;
}

public bool IsBortInList(string bort, List<flight> FlightList)
{
    bool exist = false;
    foreach (flight fl in FlightList)
        if (fl.BORT == bort)
        {
            exist = true;
            break;
        }
    return exist;
}

public void ExcelWrite(int sheet, List<flight> FlightList)
{
    exapp.SetActiveSheet(sheet);
    exapp.SetActiveCell("U2");
    foreach (flight fl in FlightList)
    {
        exapp.SetValue(fl.Delay.ToString());
        exapp.MoveDown(1);
    }
}

private int Find_Time(string ap1, string ap2, string type, List<flight_time> FlightTimeList)
```

```
    {
        int time_min = 0;
        foreach (flight_time fl in FlightTimeList)
        {
            if (fl.AC.Trim() == type.Trim())
            {
                if (fl.STN1.Trim() == ap1.Trim() && fl.STN2.Trim() == ap2.Trim())
                {
                    time_min = fl.FromTo;
                    if (time_min != 0) break;
                }
                if (fl.STN1.Trim() == ap2.Trim() && fl.STN2.Trim() == ap1.Trim())
                {
                    time_min = fl.ToFrom;
                    if (time_min != 0) break;
                }
            }
        }
        return time_min;
    }

    private bool IsReassigned(string bort, List<flight> FlightList)
    {
        bool flag = false;
        foreach (flight fl in FlightList)
            if (fl.BORT == bort && (fl.FlightStatus == flight.FS.RA_Over || fl.FlightStatus == flight.FS.RA_In ||
fl.FlightStatus == flight.FS.RA_Wait))
            {
                flag = true;
                break;
            }
        return flag;
    }

    private void IsReassigned(string bort, List<flight> FlightList, List<flight> Assignments)
    {
        Assignments.Clear();
        foreach (flight fl in FlightList)
            if (fl.BORT == bort && fl.FlightStatus == flight.FS.RA_Wait)
```

```
            Assignments.Add(fl);
        }

        private void Find_Duplicate_Entries(List<flight> FlightList)
        {
            for (int i = 0; i < FlightList.Count - 1; i++)
                for (int j = i + 1; j < FlightList.Count; j++)
                    if (FlightList[i].REC_ID == FlightList[j].REC_ID)
                    {
                        FlightList[i].MTT_Dop = FlightList[j].MTT;
                        FlightList[j].Duplicate = true;
                    }
        }

        private void Find_Previous(int i, List<flight> FlightList)
        {
            for (int j = i - 1; j >= 0; j--)
                if (FlightList[i].BORT == FlightList[j].BORT
                    && FlightList[i].DEPPORT == FlightList[j].ARRPORT
                    && FlightList[i] > FlightList[j] && !FlightList[j].Duplicate)
                {
                    FlightList[i].Previous = j;
                    FlightList[j].Next = i;
                    break;
                }
        }

        private string Find_CurrAirPort(flight fl, List<flight> FlightList)
        {
            flight f = fl;
            string dp = f.DEPPORT;
            while (f.Previous != -1 && f.FlightStatus != flight.FS.Over && f.FlightStatus != flight.FS.In &&
f.FlightStatus != flight.FS.RA_Over && f.FlightStatus != flight.FS.RA_In)
            {
                f = FlightList[f.Previous];
                if (f.FlightStatus == flight.FS.Wait || f.FlightStatus == flight.FS.RA_Wait) dp = f.DEPPORT;
                else dp = f.ARRPORT;
            }
            return dp;
```

```
}

private void button4_Click(object sender, EventArgs e)
{
   if (textBox1.Text == "")
   {
      MessageBox.Show("Не задано имя файла");
      return;
   }

   if (textBox6.Text == "")
   {
      MessageBox.Show("Не указан номер листа с расписанием");
      return;
   }

   if (textBox7.Text == "")
   {
      MessageBox.Show("Не указан номер листа с расчетами");
      return;
   }

   if (textBox8.Text == "")
   {
      MessageBox.Show("Не указан номер строки");
      return;
   }

   int from = Convert.ToInt32(textBox6.Text);
   int res_list = Convert.ToInt32(textBox7.Text);
   int res_row = Convert.ToInt32(textBox8.Text);
   int column_count = 15;
   exapp = new ExcelApp(textBox1.Text, column_count);
   bool[][] matr;
   int row;

   if (exapp.IsSheetExist("Корректировка"))
   {
```

```
        MessageBox.Show("Лист Корректировка уже существует.\n Для повторения расчета удалите или переименуйте лист в файле");
        exapp.Quit();
        return;
    }

    exapp.SetActiveSheet(from);
    int last_row = exapp.GetLastRow();

    int to = exapp.AddNewPage("Корректировка");
    exapp.CopyCells("A1:Q" + last_row.ToString(), from, "A1", to);

    List<string> Bort = new List<string>();
    List<string> Aircraft = new List<string>();
    List<string> FlNum = new List<string>();

    exapp.ReadToList(res_list, 2, "B", FlNum, true);
    exapp.ReadToList(res_list, res_row, "A", Bort, false);
    for (int i = 0; i < Bort.Count; i++)
        Aircraft.Add(exapp.FindValueInOtherColumn(from, 2, "G", "F", (string)Bort[i]));

    matr = new bool[Bort.Count][];
    for (int i = 0; i < Bort.Count; i++)
        matr[i] = new bool[FlNum.Count];

    for (int i = 0; i < Bort.Count; i++)
        for (int j = 0; j < FlNum.Count; j++)
            matr[i][j] = false;

    exapp.ReadMatrix(res_list, res_row, "B", matr);

    int[] shed = new int[FlNum.Count];
    int[] betta = new int[FlNum.Count];

    exapp.ReadShedRow(shed, res_row - Bort.Count - 11, FlNum.Count);
    exapp.ReadShedRow(betta, res_row - Bort.Count - 7, FlNum.Count);

    for (int i = 0; i < Bort.Count; i++)
        for (int j = 0; j < FlNum.Count; j++)
```

```
            if (matr[i][j])
            {
               row = exapp.FindStringInColumn(to, 2, "C", (string)FlNum[j]);
               exapp.SetActiveCell("G" + row.ToString());
               if (exapp.GetValueFromActiveCell() != Bort[i])
               {
                  //MessageBox.Show("H" + row.ToString());
                  exapp.SetValue(Bort[i]);
                  exapp.SetActiveCell("F" + row.ToString());
                  exapp.SetValue(Aircraft[i]);
                  exapp.SetActiveCell("H" + row.ToString());
                  string std = exapp.GetValueFromActiveCell();
                  exapp.SetActiveCell("I" + row.ToString());
                  string sta = exapp.GetValueFromActiveCell();
                  exapp.SetActiveCell("M" + row.ToString());
                  int blk = Convert.ToInt32(exapp.GetValueFromActiveCell());
                  ChangeTime(ref std, ref sta, betta[j] - shed[j], blk);
                  exapp.SetActiveCell("H" + row.ToString());
                  exapp.SetValue(std);
                  exapp.SetActiveCell("I" + row.ToString());
                  exapp.SetValue(sta);
               }
               exapp.SetActiveCell("O" + row.ToString());
               exapp.SetValue("Переназначен/" + exapp.GetValueFromActiveCell());

            }

      exapp.SetVisible();
      this.TopMost = true;
   }

   private void ChangeTime(ref string s1, ref string s2, int b1, int b2)
   {
      DateTime std = new DateTime();

      if (!StringToDateTime(s1, out std)) return;
      std = std.AddMinutes(b1);
      s1 = std.ToString();
      std = std.AddMinutes(b2);
```

```
            s2 = std.ToString();
            return;
        }

        public static bool StringToDateTime(string s, out DateTime dt)
        {
            string pattern = "dd.MM.yyyy HH:mm:ss";
            string pattern1 = "dd.MM.yyyy H:mm:ss";
            if (DateTime.TryParseExact(s, pattern, null,
                                DateTimeStyles.None, out dt)) return true;
            if (DateTime.TryParseExact(s, pattern1, null,
                                DateTimeStyles.None, out dt)) return true;
            return false;
        }

        private void button5_Click(object sender, EventArgs e)
        {
            if (textBox1.Text == "")
            {
                MessageBox.Show("Не задано имя файла");
                return;
            }

            if (textBox10.Text == "")
            {
                MessageBox.Show("Не указан номер листа для сортировки");
                return;
            }

             int list = Convert.ToInt32(textBox10.Text);
             /*int column_count = 15;
             exapp = new ExcelApp(textBox1.Text, column_count);
             exapp.SortByColumn(list); */
            Microsoft.Office.Interop.Excel.Application ObjExcel = new
Microsoft.Office.Interop.Excel.Application();
            //Открываем книгу
            Microsoft.Office.Interop.Excel.Workbook ObjWorkBook = ObjExcel.Workbooks.Open(textBox1.Text,
/*0, false, 5, "", "", false, Microsoft.Office.Interop.Excel.XlPlatform.xlWindows, "", true, false, 0, true, false,
false);*/
                Type.Missing, Type.Missing, Type.Missing, Type.Missing,
```

```
            Type.Missing, Type.Missing, Type.Missing, Type.Missing,
            Type.Missing, Type.Missing, Type.Missing, Type.Missing,
            Type.Missing, Type.Missing);
        //Выбираем таблицу(лист)
        Microsoft.Office.Interop.Excel.Worksheet ObjWorkSheet;
        ObjWorkSheet = (Microsoft.Office.Interop.Excel.Worksheet)ObjWorkBook.Sheets[list];

        //Область сортировки
        Microsoft.Office.Interop.Excel.Range range = ObjWorkSheet.UsedRange;
        //По какому столбцу сортировать
        Microsoft.Office.Interop.Excel.Range rangeKey = ObjWorkSheet.get_Range("H2");
        //Добавляем параметры сортировки
        ObjWorkSheet.Sort.SortFields.Add(rangeKey);
        ObjWorkSheet.Sort.SetRange(range);
        ObjWorkSheet.Sort.Header = Microsoft.Office.Interop.Excel.XlYesNoGuess.xlYes;
        ObjWorkSheet.Sort.Orientation = Microsoft.Office.Interop.Excel.XlSortOrientation.xlSortColumns;
        ObjWorkSheet.Sort.SortMethod = Microsoft.Office.Interop.Excel.XlSortMethod.xlPinYin;
        ObjWorkSheet.Sort.Apply();
        ObjExcel.Visible = true;
    }

    private void button6_Click(object sender, EventArgs e)
    {
        if (textBox1.Text == "")
        {
            MessageBox.Show("Не задано имя файла");
            return;
        }

        Summary f = new Summary(textBox1.Text);
        f.ShowDialog();
    }
  }
}
```

Summary.cs – форма объединения

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
```

```
namespace s7
{
   public partial class Summary : Form
   {
      ExcelApp exapp;
      public Summary(string file)
      {
         InitializeComponent();
         int column_count = 15;
         exapp = new ExcelApp(file, column_count);
         int count = exapp.GetSheetsCount();

         for (int i = 1; i <= count; i++)
         {
            string name = exapp.GetSheetName(i);
            clb.Items.Add(name);
            if(name.StartsWith("Задержки")) clb.SetItemChecked(i - 1, true);
         }
      }

      private void button1_Click(object sender, EventArgs e)
      {

         if (textBox1.Text == "")
         {
            MessageBox.Show("Не указан лист для свода");
            return;
         }

         if (exapp.IsSheetExist(textBox1.Text))
         {
            MessageBox.Show("Лист " + textBox1.Text + " уже существует");
            return;
         }

         if (textBox2.Text == "")
         {
            MessageBox.Show("Не указан номер строки");
            return;
         }

         int from;
         int res_row = Convert.ToInt32(textBox2.Text);
         bool[][] matr;
         int svod = exapp.AddNewPage(textBox1.Text);

         List<string> Bort = new List<string>();
         List<string> FlNum = new List<string>();
         List<string> FlName = new List<string>();

         from = clb.CheckedIndices[0] + 1;
         exapp.ReadToList(from, 1, "B", FlName, true);
         exapp.ReadToList(from, 2, "B", FlNum, true);
         exapp.ReadToList(from, res_row, "A", Bort, false);

         matr = new bool[Bort.Count][];
         for (int i = 0; i < Bort.Count; i++)
```

```
    matr[i] = new bool[FlNum.Count];

for (int i = 0; i < Bort.Count; i++)
    for (int j = 0; j < FlNum.Count; j++)
        matr[i][j] = false;

exapp.ReadMatrix(from, res_row, "B", matr);

int[] shed = new int[FlNum.Count];
int[] betta = new int[FlNum.Count];

exapp.ReadShedRow(shed, res_row - Bort.Count - 11, FlNum.Count);
exapp.ReadShedRow(betta, res_row - Bort.Count - 7, FlNum.Count);

foreach (int indexChecked in clb.CheckedIndices)
{
    if (from == indexChecked + 1) continue;
    from = indexChecked + 1;
    exapp.SetActiveSheet(from);

    List<string> FlNum1 = new List<string>();
    exapp.ReadToList(from, 2, "B", FlNum1, true);

    bool[][] matr1 = new bool[Bort.Count][];
    for (int i = 0; i < Bort.Count; i++)
        matr1[i] = new bool[FlNum1.Count];

    for (int i = 0; i < Bort.Count; i++)
        for (int j = 0; j < FlNum1.Count; j++)
            matr1[i][j] = false;

    exapp.ReadMatrix(from, res_row, "B", matr1);

    int[] shed1 = new int[FlNum1.Count];
    int[] betta1 = new int[FlNum1.Count];

    exapp.ReadShedRow(shed1, res_row - Bort.Count - 11, FlNum1.Count);
    exapp.ReadShedRow(betta1, res_row - Bort.Count - 7, FlNum1.Count);

    for (int i = 0; i < Bort.Count; i++)
        for (int j = 0; j < FlNum1.Count; j++)
            if (matr1[i][j])
            {
                int col = -1;
                for (int k = 0; k < FlNum.Count; k++)
                    if (FlNum[k] == FlNum1[j])
                    {
                        col = k;
                        break;
                    }
                if (col > -1)
                {
                    matr[i][col] = true;
                    shed[col] = shed1[j];
                    betta[col] = betta1[j];
                }
            }
```

```
}

exapp.SetActiveSheet(svod);
exapp.SetActiveCell("B1");
foreach(string s in FlName)
{
   exapp.SetValue(s);
   exapp.MoveRight(1);
}

exapp.SetActiveCell("B2");
foreach (string s in FlNum)
{
   exapp.SetValue(s);
   exapp.MoveRight(1);
}

exapp.SetActiveCell("A4");
exapp.SetValue("SHED");
foreach (int i in shed)
{
   exapp.MoveRight(1);
   exapp.SetValue(i.ToString());
}

int sum = 0;

exapp.SetActiveCell("A6");
exapp.SetValue("objective");
for (int j = 0; j < FlNum.Count; j++)
{
   exapp.MoveRight(1);
   int diff = betta[j] - shed[j];
   if (betta[j] != 0)
   {
      exapp.SetValue(diff.ToString());
      sum += diff;
   }
   else
      exapp.SetValue("0");
}
exapp.MoveRight(2);
exapp.SetValue(sum.ToString());

exapp.SetActiveCell("A8");
exapp.SetValue("korr");
foreach (int i in betta)
{
   exapp.MoveRight(1);
   exapp.SetValue(i.ToString());
}

exapp.SetActiveCell("A10");
exapp.SetValue("objective");
sum = 0;
for (int j = 0; j < FlNum.Count; j++)
{
```

```
            int sum1 = 0;
            for (int i = 0; i < Bort.Count; i++)
               if (matr[i][j]) sum1++;
            exapp.MoveRight(1);
            exapp.SetValue(sum1.ToString());
            sum += sum1;
         }
         exapp.MoveRight(2);
         exapp.SetValue(sum.ToString());

         exapp.SetActiveCell("A13");
         foreach (string s in Bort)
         {
            exapp.SetValue(s);
            exapp.MoveDown(1);
         }

         exapp.SetActiveCell("B13");
         for (int i = 0; i < Bort.Count; i++)
         {
            sum = 0;
            for (int j = 0; j < FlNum.Count; j++)
            {
               if (matr[i][j])
               {
                  exapp.SetValue("1");
                  sum++;
               }
               else exapp.SetValue("0");
               exapp.MoveRight(1);
            }
            exapp.MoveRight(1);
            exapp.SetValue(sum.ToString());
            exapp.MoveLeft(FlNum.Count + 1);
            exapp.MoveDown(1);
         }

         exapp.SetVisible();
         this.TopMost = true;
      }
   }
}
```

ExcelApp.cs – класс для работы с исходным файлом

```
using System;
using System.Collections.Generic;
using System.Reflection;

namespace s7
{
   public class ExcelApp
   {
      object app, range;
      object WorkBooks, WorkBook, WorkSheets, WorkSheet;
      object[] args, args2, args4;
```

```
    int column_count, current_row;

    public ExcelApp(string excel_file, int count)
    {
       args = new object[1];
       args2 = new object[2];
       args4 = new object[4];

       Type tExcelObj = Type.GetTypeFromProgID("Excel.Application");
       app = Activator.CreateInstance(tExcelObj);
       WorkBooks = app.GetType().InvokeMember("Workbooks", BindingFlags.GetProperty, null, app, null);
       args[0] = excel_file;
       WorkBook = WorkBooks.GetType().InvokeMember("Open", BindingFlags.InvokeMethod, null,
WorkBooks, args);
       WorkSheets = WorkBook.GetType().InvokeMember("Worksheets",
              BindingFlags.GetProperty, null, WorkBook, null);
       column_count = count;
    }

    public int GetSheetsCount()
    {
       return (int) WorkSheets.GetType().InvokeMember("Count",
              BindingFlags.GetProperty, null, WorkSheets, null);
    }
    public void PrepareReadSheet(int sheet, int row)
    {
       SetActiveSheet(sheet);
       current_row = row;
    }

    public string FindValueInOtherColumn(int sheet, int row, string column1, string column2, string pattern)
    {
       row = FindStringInColumn(sheet, row, column1, pattern);
       if (row >= 0)
       {
          SetActiveCell(column2 + row.ToString());
          return GetValueFromActiveCell();
       }
       return "";
    }

    public int FindStringInColumn(int sheet, int row, string column, string pattern)
    {
       SetActiveSheet(sheet);
       SetActiveCell(column + row.ToString());
       string s = GetValueFromActiveCell();
       while (s != "")
       {
          if (s == pattern) return row;
          MoveDown(1);
          row++;
          s = GetValueFromActiveCell();
       }
       return -1;
    }

    public void ReadToList(int sheet, int row, string column, List<string> rlist, bool horiz_direction)
```

```
{
   SetActiveSheet(sheet);
   SetActiveCell(column + row.ToString());
   string s = GetValueFromActiveCell();
   while (s != "")
   {
      rlist.Add(s);
      if (horiz_direction) MoveRight(1);
      else MoveDown(1);
      s = GetValueFromActiveCell();
   }
}

public void ReadMatrix(int sheet, int row, string column, bool[][] matr)
{
   int row_count = 0, col_count = 0;
   SetActiveSheet(sheet);
   SetActiveCell(column + row.ToString());
   string s = GetValueFromActiveCell();
   while (s != "")
   {
      col_count++;
      MoveRight(1);
      s = GetValueFromActiveCell();
   }

   SetActiveCell(column + row.ToString());
   s = GetValueFromActiveCell();
   while (s != "")
   {
      row_count++;
      MoveDown(1);
      s = GetValueFromActiveCell();
   }

   SetActiveCell(column + row.ToString());
   for (int i = 0; i < row_count; i++)
   {
      for (int j = 0; j < col_count; j++)
      {
         s = GetValueFromActiveCell();
         if (s == "1") matr[i][j] = true;
         MoveRight(1);
      }
      MoveLeft(col_count);
      MoveDown(1);
   }
}

public void SetActiveSheet(int sheet)
{
   args[0] = sheet;
   WorkSheet = WorkSheets.GetType().InvokeMember("Item",
            BindingFlags.GetProperty, null, WorkSheets, args);
}

public void PrepareWriteSheet(int sheet, int row)
```

```
{
   SetActiveSheet(sheet);
   SetActiveCell("A" + row);
}

public void WriteShedRow(string[] s)
{
   current_row++;
   for (int i = 0; i < column_count; i++)
   {
      SetValue(s[i]);
      MoveRight(1);
   }
}

public void WriteChanges(bool a, string s)
{
   SetActiveCell("N" + current_row);
   SetValue("NULL");
   if (a)
   {
      SetActiveCell("O" + current_row);
      SetValue(s);
   }
}

public bool ReadNextShedRow(string[] s)
{
   current_row++;
   if (IsEmptyRow()) return false;
   for (int i = 0; i < column_count; i++)
   {
      s[i] = GetValueFromActiveCell();
      MoveRight(1);
   }
   return true;
}

public void ReadShedRow(int[] shed, int row, int count)
{
   SetActiveCell("B" + row.ToString());
   for (int i = 0; i < count; i++)
   {
      string s = GetValueFromActiveCell();
      if(s == "") shed[i] = 0;
      else shed[i] = Convert.ToInt32(s);
      MoveRight(1);
   }
   return;
}

public int GetLastRow()
{
   current_row = 2;
   while (!IsEmptyRow()) current_row++;
   return current_row - 1;
}
```

```
        public bool ReadNextTimeRow(string[] s)
        {
            current_row++;
            if (IsEmptyRow()) return false;

            s[0] = GetValueFromActiveCell();
            MoveRight(1);
            s[1] = GetValueFromActiveCell();
            MoveRight(2);
            s[2] = GetValueFromActiveCell();
            MoveRight(7);
            s[3] = GetValueFromActiveCell();
            MoveRight(1);
            s[4] = GetValueFromActiveCell();
            return true;
        }

        public string GetValueFromActiveCell()
        {
            try
            {
                return range.GetType().InvokeMember("Value", BindingFlags.GetProperty, null, range,
null).ToString();
            }
            catch
            {
                return "";
            }

        }

        public void SetValue(string val)
        {
            args[0] = val;
            range.GetType().InvokeMember("Value", BindingFlags.SetProperty, null, range, args);
        }

        public void MoveRight(int step)
        {
            args2[0] = 0;
            args2[1] = step;
            range = range.GetType().InvokeMember("Offset", BindingFlags.GetProperty, null, range, args2);
        }

        public void MoveLeft(int step)
        {
            MoveRight(-step);
        }

        public void MoveDown(int step)
        {
            args2[0] = step;
            args2[1] = 0;
            range = range.GetType().InvokeMember("Offset", BindingFlags.GetProperty, null, range, args2);
        }
```

```
        public void MoveUp(int step)
        {
            MoveDown(-step);
        }

        private bool IsEmptyRow()
        {
            SetActiveCell("A" + current_row);
            if (range.GetType().InvokeMember("Value", BindingFlags.GetProperty, null, range, null) == null)
return true;
            return false;
        }
        public void SetActiveCell(string cell)
        {
            args[0] = cell;
            range = WorkSheet.GetType().InvokeMember("Range", BindingFlags.GetProperty, null, WorkSheet,
args);
        }

        public void CopyCells(string cells, int from, string cell, int to)
        {
            SetActiveSheet(from);
            args[0] = cells;
            range = WorkSheet.GetType().InvokeMember("Range", BindingFlags.GetProperty, null, WorkSheet,
args);
            range.GetType().InvokeMember("Copy",
                BindingFlags.InvokeMethod, null, range, null);
            SetActiveSheet(to);
            SetActiveCell(cell);
            args4[0] = 8;  //xlPasteColumnWidths - сохранить ширину столбцов
            args4[1] = Missing.Value;
            args4[2] = Missing.Value;
            args4[3] = Missing.Value;
            range.GetType().InvokeMember("PasteSpecial",
                BindingFlags.InvokeMethod, null, range, args4);
            WorkSheet.GetType().InvokeMember("Paste",
                BindingFlags.InvokeMethod, null, WorkSheet, null);
        }

        public bool IsSheetExist(string name)
        {
            int count = GetSheetsCount();

            for (int i = 1; i <= count; i++)
            {
                if (name == GetSheetName(i)) return true;
            }
            return false;
        }

        public string GetSheetName(int number)
        {
            args[0] = number;
            WorkSheet = WorkSheets.GetType().InvokeMember("Item",
                    BindingFlags.GetProperty, null, WorkSheets, args);
            return (string)WorkSheet.GetType().InvokeMember("Name", BindingFlags.GetProperty, null,
WorkSheet, null);
```

```
    }

    public int AddNewPage(string Name)
    {
        int count = (int)WorkSheets.GetType().InvokeMember("Count",
                BindingFlags.GetProperty, null, WorkSheets, null);
        SetActiveSheet(count);

        args4[0] = Missing.Value;
        args4[1] = WorkSheet;
        args4[2] = Missing.Value;
        args4[3] = Missing.Value;
        WorkSheet = WorkSheets.GetType().InvokeMember("Add",
            BindingFlags.InvokeMethod, null, WorkSheets, args4);

        WorkSheet.GetType().InvokeMember("Name", BindingFlags.SetProperty, null, WorkSheet, new
object[] { Name });
        return count + 1;
    }

    public void SortByColumn(int list)
    {
        object Sort, SortFields;
        SetActiveSheet(list);
        int row = GetLastRow();

        Sort = WorkSheet.GetType().InvokeMember("Sort",
            BindingFlags.InvokeMethod, null, WorkSheet, null);
        SortFields = Sort.GetType().InvokeMember("SortFields",
                BindingFlags.GetProperty, null, Sort, null);
        SortFields.GetType().InvokeMember("Clear",
            BindingFlags.InvokeMethod, null, SortFields, null);

        SetActiveCell("H2");
        args4[0] = range;
        args4[1] = 0; // SortOn=SortOnValues - сортировка по значению
        args4[2] = 1; // Order=xlAscending - сортировка по возрастанию
        args4[3] = 0; // DataOption=xlSortNormal - числовые и текстовые данные сортируются отдельно
        SortFields.GetType().InvokeMember("Add",
            BindingFlags.InvokeMethod, null, SortFields, args4);

        SetActiveCell("A2:Q" + row.ToString());
        args[0] = range;
        Sort.GetType().InvokeMember("SetRange",
            BindingFlags.InvokeMethod, null, Sort, args);
        Sort.GetType().InvokeMember("Apply",
            BindingFlags.InvokeMethod, null, Sort, null);
        /*Microsoft.Office.Interop.Excel.Application ObjExcel = new
Microsoft.Office.Interop.Excel.Application();
        //Открываем книгу
        Microsoft.Office.Interop.Excel.Workbook ObjWorkBook = ObjExcel.Workbooks.Open("test.xls", 0,
false, 5, "", "", false, Microsoft.Office.Interop.Excel.XlPlatform.xlWindows, "", true, false, 0, true, false, false);
        //Выбираем таблицу(лист)
        Microsoft.Office.Interop.Excel.Worksheet ObjWorkSheet;
        ObjWorkSheet = (Microsoft.Office.Interop.Excel.Worksheet)ObjWorkBook.Sheets[1];

        //Область сортировки
```

```
            Microsoft.Office.Interop.Excel.Range range = ObjWorkSheet.get_Range("A9", "M100");
            //По какому столбцу сортировать
            Microsoft.Office.Interop.Excel.Range rangeKey = ObjWorkSheet.get_Range("A" + (i - 1));
            //Добавляем параметры сортировки
            ObjWorkSheet.Sort.SortFields.Add(rangeKey);
            ObjWorkSheet.Sort.SetRange(range);
            ObjWorkSheet.Sort.Orientation = Microsoft.Office.Interop.Excel.XlSortOrientation.xlSortColumns;
            ObjWorkSheet.Sort.SortMethod = Microsoft.Office.Interop.Excel.XlSortMethod.xlPinYin;
            ObjWorkSheet.Sort.Apply(); */
        }

        public void SetVisible()
        {
            args[0] = true;
            app.GetType().InvokeMember("Visible", BindingFlags.SetProperty, null, app, args);
        }

        public void Quit()
        {
            app.GetType().InvokeMember("Quit", BindingFlags.InvokeMethod, null, app, null);
        }
    }
}
```

flight_time.cs – класс для работы с рейсами

```
using System;

namespace s7
{
    public class flight_time
    {
        public string STN1;
        public string STN2;
        public string AC;
        public int FromTo;
        public int ToFrom;

        public flight_time(string[] s)
        {
            STN1 = s[0];
            STN2 = s[1];
            AC = s[2];
            FromTo = Convert.ToInt32(s[3]);
            ToFrom = Convert.ToInt32(s[4]);
        }
    }
}
```

flight.cs – класс для работы со справочником полетного времени

```
using System;
using System.Globalization;
namespace s7
{
    public class flight
    {
```

```
public enum FS
{
   Wait,
   In,
   Over,
   RA_Wait,
   RA_In,
   RA_Over,
   Other
}
public string REC_ID;
public string Carrier;
public int FlightNo;
public string DEPPORT;
public string ARRPORT;
public string AIRCRAFT;
public string BORT;
public string STD;
public string STA;
public string ATD;
public string ATA;
public int MTT;
public int BLK_Plan;
public string CurrentDelay;
public FS FlightStatus;
public string ETD;
public string ETA;
public int Delay;
public int MTT_Dop;
public int Next;
public int Previous;
public bool Duplicate;
public flight(string[] s)
{
   REC_ID = s[0];
   Carrier = s[1];
   FlightNo = Convert.ToInt32(s[2]);
   DEPPORT = s[3];
   ARRPORT = s[4];
   AIRCRAFT = s[5];
   BORT = s[6];
   STD = s[7];
   STA = s[8];
   ATD = s[9];
   ATA = s[10];
   MTT = Convert.ToInt32(s[11]);
   BLK_Plan = Convert.ToInt32(s[12]);
   CurrentDelay = s[13];
   FlightStatus = StringToStatus(s[14]);
}

public bool Change(string for_time)
{
   //if (FlightStatus == FS.Reassigned) return false;
   DateTime std = new DateTime();
   DateTime sta = new DateTime();
   DateTime atd = new DateTime();
```

```
    DateTime ata = new DateTime();
    DateTime cur_time = new DateTime();
    string ct;
    int flag = 2; // есть фактич. время вылета и прилета
    if (!StringToDateTime(STD, out std)) return false;
    if (!StringToDateTime(STA, out sta)) return false;
    if (!StringToDateTime(ATA, out ata)) flag = 1; // есть фактич. время вылета
    if (!StringToDateTime(ATD, out atd)) flag = 0; // нет фактич. время вылета

    ct = IntWithZero(std.Day) + "." + IntWithZero(std.Month)
              + "." + std.Year + " " + for_time + ":" + "00";
    if (!StringToDateTime(ct, out cur_time)) return false;

    if (flag == 0 && cur_time < std || flag > 0 && cur_time < atd) // еще не вылетел
       if (FlightStatus == FS.Wait || FlightStatus == FS.RA_Wait) return false;
       else
       {
          if (FlightStatus == FS.RA_Over || FlightStatus == FS.RA_In) FlightStatus = FS.RA_Wait;
          else FlightStatus = FS.Wait;
          return true;
       }
    if (flag == 0 && cur_time >= sta ||
       flag == 2 && cur_time >= ata ||
       flag == 1 && cur_time > atd.AddMinutes(BLK_Plan)) // уже прилетел
       if (FlightStatus == FS.Over || FlightStatus == FS.RA_Over) return false;
       else
       {
          if (FlightStatus == FS.RA_Wait || FlightStatus == FS.RA_In) FlightStatus = FS.RA_Over;
          else FlightStatus = FS.Over;
          return true;
       }

    if (flag == 0 && cur_time > std && cur_time < sta ||
      flag == 2 && cur_time > atd && cur_time < ata ||
      flag == 1 && cur_time > atd && cur_time < atd.AddMinutes(BLK_Plan)) // в полете
       if (FlightStatus == FS.In  || FlightStatus == FS.RA_In) return false;
       else
       {
          if (FlightStatus == FS.RA_Wait || FlightStatus == FS.RA_Over) FlightStatus = FS.RA_In;
          else FlightStatus = FS.In;
          return true;
       }
    return false;
}
private string IntWithZero(int i)
{
    string s = i.ToString();
    if (s.Length == 1) s = "0" + s;
    return s;
}
public FS StringToStatus(string s)
{
    FS fs = FS.Other;
    if (s == "Завершен") fs = FS.Over;
    if (s == "В полете") fs = FS.In;
    if (s == "Ожидает вылета") fs = FS.Wait;
    if (s == "Переназначен/Завершен") fs = FS.RA_Over;
```

```
    if (s == "Переназначен/В полете") fs = FS.RA_In;
    if (s == "Переназначен/Ожидает вылета") fs = FS.RA_Wait;
    return fs;
  }
  public string StatusToString()
  {
    string s = "";
    if (FlightStatus == FS.Over) s = "Завершен";
    if (FlightStatus == FS.In) s = "В полете";
    if (FlightStatus == FS.Wait) s = "Ожидает вылета";
    if (FlightStatus == FS.RA_Over) s = "Переназначен/Завершен";
    if (FlightStatus == FS.RA_In) s = "Переназначен/В полете";
    if (FlightStatus == FS.RA_Wait) s = "Переназначен/Ожидает вылета";
    return s;
  }
  public void ToStrings(string[] s)
  {
    s[0] = REC_ID;
    s[1] = Carrier;
    s[2] = FlightNo.ToString();
    s[3] = DEPPORT;
    s[4] = ARRPORT;
    s[5] = AIRCRAFT;
    s[6] = BORT;
    s[7] = STD;
    s[8] = STA;
    s[9] = ATD;
    s[10] = ATA;
    s[11] = MTT.ToString();
    s[12] = BLK_Plan.ToString();
    s[13] = CurrentDelay;
    s[14] = StatusToString();
  }
  public void Diff_Actual_Planned_Departure_Time()
  {
    Diff_Time(ATD, STD);
  }
  public void Diff_Estimated_Planned_Departure_Time()
  {
    Diff_Time(ETD, STD);
  }
 public void Calc_Estimated_Departure_Time(string s)
  {
    DateTime eta = new DateTime();
    DateTime etd = new DateTime();
    DateTime std = new DateTime();

    if (!StringToDateTime(s, out eta)) return;
    eta = eta.AddMinutes(MTT);
    if (!StringToDateTime(STD, out std)) return;
    if (eta <= std) etd = std;
    else etd = eta;
    ETD = etd.ToString();
  }
  public void Calc_Estimated_Arrival_Time(string s)
  {
    DateTime dt = new DateTime();
```

```
            if (!StringToDateTime(s, out dt)) return;
            ETA = dt.AddMinutes(BLK_Plan).ToString();
        }
        public static bool StringToDateTime(string s, out DateTime dt)
        {
            string pattern = "dd.MM.yyyy HH:mm:ss";
            string pattern1 = "dd.MM.yyyy H:mm:ss";
            if (DateTime.TryParseExact(s, pattern, null,
                                DateTimeStyles.None, out dt)) return true;
            if (DateTime.TryParseExact(s, pattern1, null,
                                DateTimeStyles.None, out dt)) return true;
            return false;
        }
        private void Diff_Time(string s1, string s2)
        {
            DateTime dt1 = new DateTime();
            DateTime dt2 = new DateTime();

            if (!StringToDateTime(s1, out dt1)) return;
            if (!StringToDateTime(s2, out dt2)) return;
            TimeSpan diff = dt1.Subtract(dt2);
            Delay = diff.Days * 24 * 60 + diff.Hours * 60 + diff.Minutes;
            if (Delay < 0) Delay = 0;
        }
        public static bool operator >(flight a, flight b)
        {
            DateTime dt1 = new DateTime();
            DateTime dt2 = new DateTime();

            if (!StringToDateTime(a.STD, out dt1)) return false;
            if (!StringToDateTime(b.STA, out dt2)) return false;
            return dt1 > dt2;
        }

        public static bool operator <(flight a, flight b)
        {
            DateTime dt1 = new DateTime();
            DateTime dt2 = new DateTime();

            if (!StringToDateTime(a.STD, out dt1)) return false;
            if (!StringToDateTime(b.STA, out dt2)) return false;
            return dt1 < dt2;
        }
    }
}
```

ПРИЛОЖЕНИЕ Д

Копии свидетельств о государственной регистрации программы для ЭВМ

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2020663452

Программа для расчета задержек и корректировки расписания движения воздушных судов авиакомпании

Правообладатель: ***ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ» (RU)***

Авторы: ***Эстрайх Игорь Викторович (RU), Короткова Юлия Леонидовна (RU), Мезенцев Юрий Анатольевич (RU)***

Заявка № **2020662848**
Дата поступления **28 октября 2020 г.**
Дата государственной регистрации
в Реестре программ для ЭВМ ***28 октября 2020 г.***

Руководитель Федеральной службы по интеллектуальной собственности

Г.П. Ивлиев

Рисунок Д.1 – Свидетельство о государственной регистрации программы для ЭВМ «Программы для расчета задержек и корректировки расписания»

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2021666271

Программа для решения задачи оптимизации расписаний параллельно-последовательной системы специального типа

Правообладатель: ***ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ» (RU)***

Авторы: ***Эстрайх Игорь Викторович (RU), Короткова Юлия Леонидовна (RU), Мезенцев Юрий Анатольевич (RU)***

Заявка № **2021665658**

Дата поступления **12 октября 2021 г.**

Дата государственной регистрации в Реестре программ для ЭВМ ***12 октября 2021 г.***

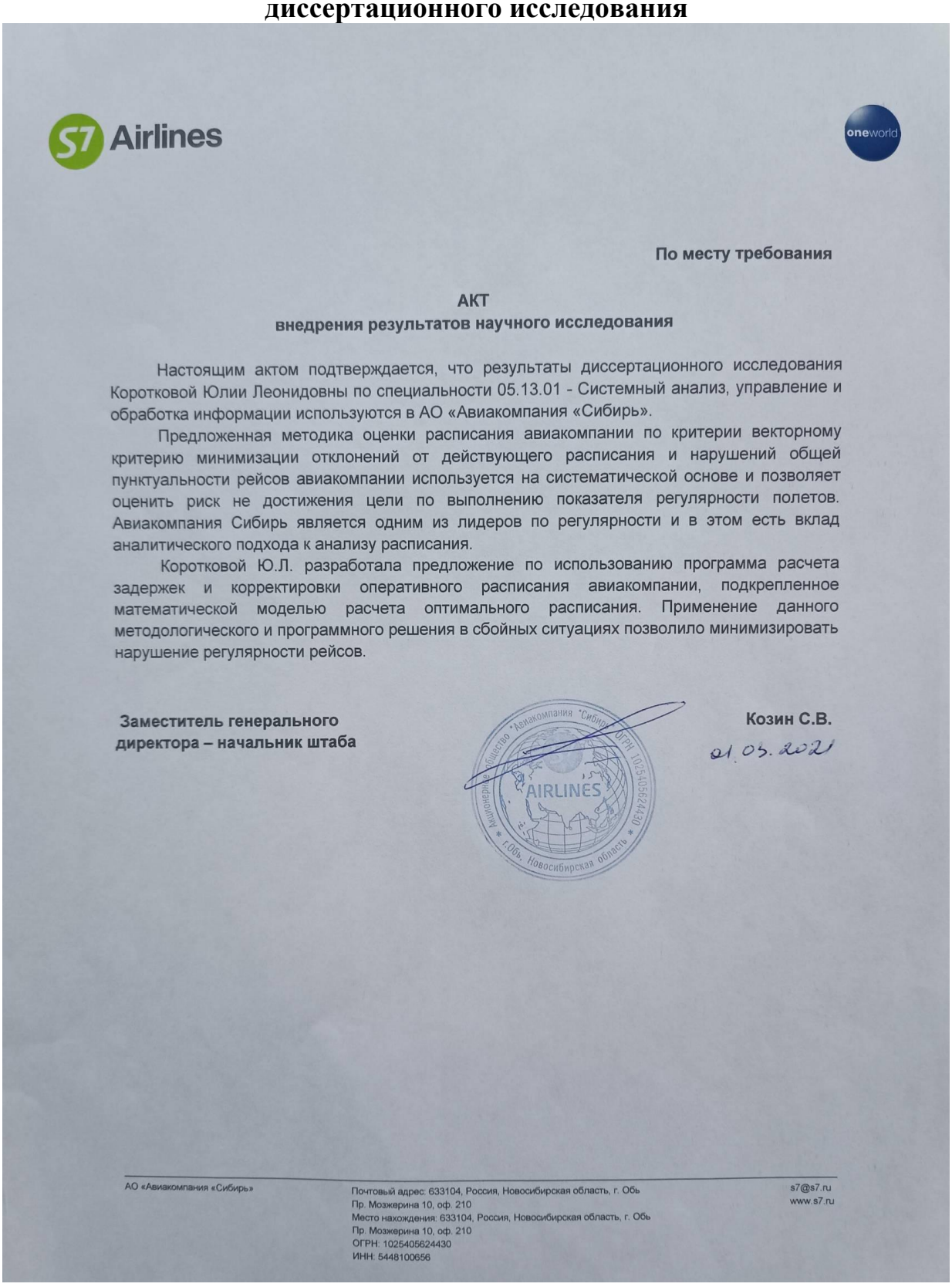
Руководитель Федеральной службы по интеллектуальной собственности

Г.П. Ивлиев

Рисунок Д.2 – Свидетельство о государственной регистрации программы для ЭВМ «Программа для решения задачи оптимизации расписаний параллельно-последовательной системы специального типа»

ПРИЛОЖЕНИЕ Е

Копии документов, подтверждающих использование результатов диссертационного исследования



S7 Airlines

oneworld

По месту требования

АКТ
внедрения результатов научного исследования

Настоящим актом подтверждается, что результаты диссертационного исследования Коротковой Юлии Леонидовны по специальности 05.13.01 - Системный анализ, управление и обработка информации используются в АО «Авиакомпания «Сибирь».

Предложенная методика оценки расписания авиакомпании по критерии векторному критерию минимизации отклонений от действующего расписания и нарушений общей пунктуальности рейсов авиакомпании используется на систематической основе и позволяет оценить риск не достижения цели по выполнению показателя регулярности полетов. Авиакомпания Сибирь является одним из лидеров по регулярности и в этом есть вклад аналитического подхода к анализу расписания.

Коротковой Ю.Л. разработала предложение по использованию программа расчета задержек и корректировки оперативного расписания авиакомпании, подкрепленное математической моделью расчета оптимального расписания. Применение данного методологического и программного решения в сбойных ситуациях позволило минимизировать нарушение регулярности рейсов.

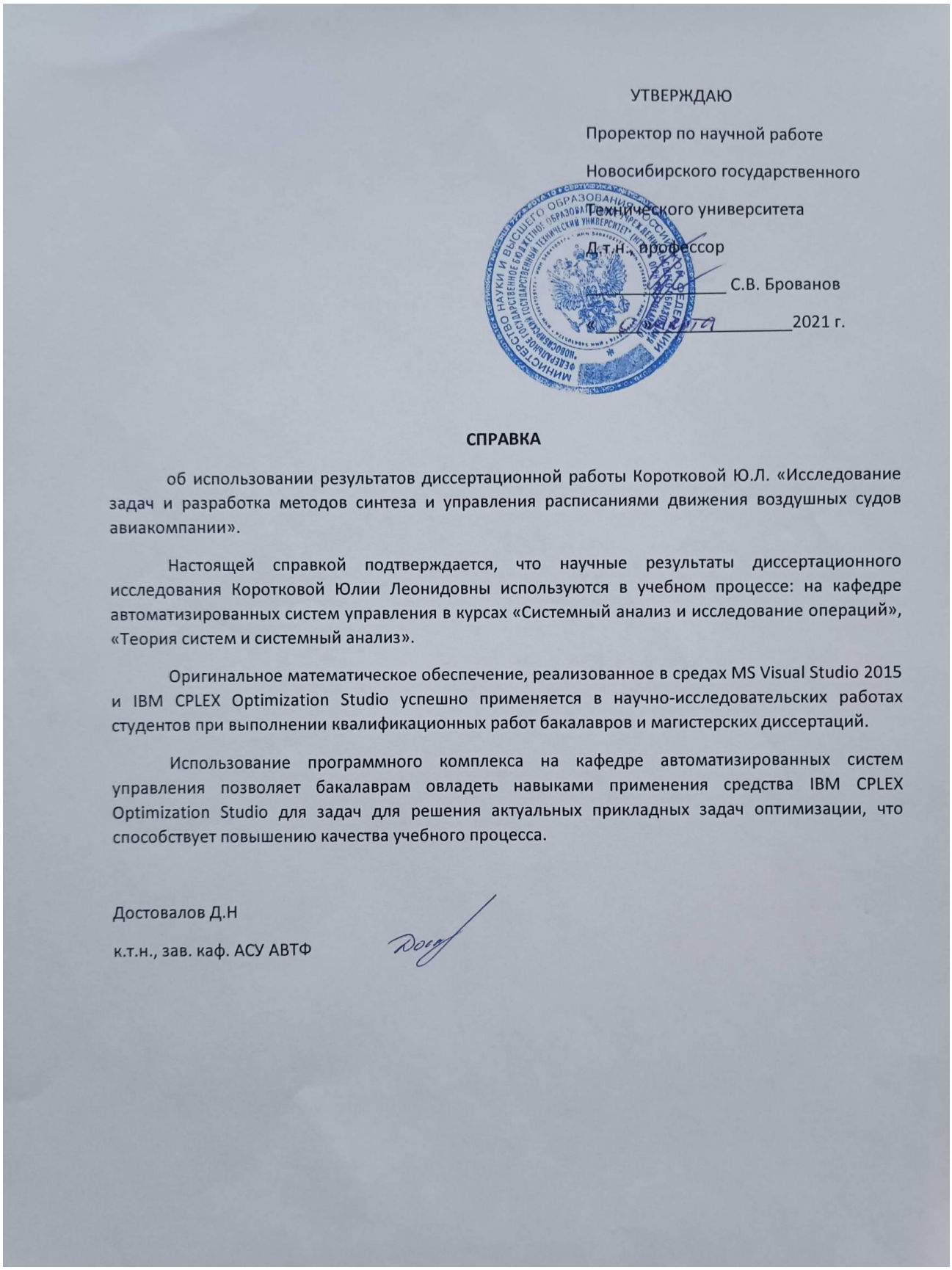
Заместитель генерального директора – начальник штаба — **Козин С.В.**

01.05.2021

Акционерное общество "Авиакомпания "Сибирь" ОГРН 1025405624430 AIRLINES г.Обь, Новосибирская область

АО «Авиакомпания «Сибирь»
Почтовый адрес: 633104, Россия, Новосибирская область, г. Обь
Пр. Мозжерина 10, оф. 210
Место нахождения: 633104, Россия, Новосибирская область, г. Обь
Пр. Мозжерина 10, оф. 210
ОГРН: 1025405624430
ИНН: 5448100656
s7@s7.ru
www.s7.ru

Рисунок Е.1 – Акт о внедрении результатов кандидатской диссертации



УТВЕРЖДАЮ

Проректор по научной работе

Новосибирского государственного

Технического университета

Д.т.н., профессор

_____ С.В. Брованов

«___» _____ 2021 г.

СПРАВКА

об использовании результатов диссертационной работы Коротковой Ю.Л. «Исследование задач и разработка методов синтеза и управления расписаниями движения воздушных судов авиакомпании».

Настоящей справкой подтверждается, что научные результаты диссертационного исследования Коротковой Юлии Леонидовны используются в учебном процессе: на кафедре автоматизированных систем управления в курсах «Системный анализ и исследование операций», «Теория систем и системный анализ».

Оригинальное математическое обеспечение, реализованное в средах MS Visual Studio 2015 и IBM CPLEX Optimization Studio успешно применяется в научно-исследовательских работах студентов при выполнении квалификационных работ бакалавров и магистерских диссертаций.

Использование программного комплекса на кафедре автоматизированных систем управления позволяет бакалаврам овладеть навыками применения средства IBM CPLEX Optimization Studio для задач для решения актуальных прикладных задач оптимизации, что способствует повышению качества учебного процесса.

Достовалов Д.Н

к.т.н., зав. каф. АСУ АВТФ

Рисунок Е.2 – Справка об использовании результатов кандидатской диссертации в учебном процессе