

Федеральное бюджетное образовательное учреждение
высшего профессионального образования
«ВОЛЖСКАЯ ГОСУДАРСТВЕННАЯ АКАДЕМИЯ ВОДНОГО ТРАНСПОРТА»

На правах рукописи

Куимова Анастасия Сергеевна

**МОДЕЛИ И АЛГОРИТМЫ УПРАВЛЕНИЯ
ОБСЛУЖИВАНИЕМ БИНАРНЫХ ПОТОКОВ
ОБЪЕКТОВ В ОДНОПРОЦЕССОРНОЙ СИСТЕМЕ
С НАКОПИТЕЛЬНО-РАСХОДНЫМИ КОМПОНЕНТАМИ**

Специальность 05.13.01 – Системный анализ, управление
и обработка информации (в науке и промышленности)
по техническим наукам

Диссертация на соискание учёной степени
кандидата технических наук

Научный руководитель: д.т.н. профессор Федосенко Ю.С.

Научный консультант: д.т.н. профессор Коган Д.И.

Нижний Новгород – 2013

Оглавление

Введение	5
Глава 1. Модели обслуживания и методы	
решения задач дискретного программирования	14
§ 1.1 Элементы теории расписаний	14
§ 1.2 Модели обслуживания конечных множеств и детерминированных потоков объектов	16
§ 1.3 Подходы к решению дискретных задач синтеза стратегий обслуживания	19
1.3.1 Концепция динамического программирования	20
1.3.2 Концепция ветвей и границ	22
1.3.3 Метаэвристические концепции	24
1.3.4 Подходы к решению задач бикритериальной оптимизации .	26
Глава 2. Модели, однокритериальные задачи и алгоритмы синтеза стратегий обслуживания объектов в однопроцессорной системе с накопительно-расходным компонентом	29
§ 2.1 Задачи обслуживания бинарного множества объектов процессором с накопительно-расходным компонентом	29
2.1.1 Математическая модель	29
2.1.2 Постановки оптимизационных задач и исследование их вычислительной сложности	32
2.1.3 Решающие соотношения динамического программирования	35
2.1.4 Пример синтеза стратегии обслуживания	39
2.1.5 Результаты вычислительных экспериментов	42
§ 2.2 Задачи обслуживания бинарного потока объектов процессором с накопительно-расходным компонентом	44
2.2.1 Математическая модель	44
2.2.2 Постановки оптимизационных задач	44
2.2.3 Решающие соотношения динамического программирования	46
2.2.4 Результаты вычислительных экспериментов	49
§ 2.3 Решение задач обслуживания методом ветвей и границ	49
2.3.1 Описание алгоритма	49
2.3.2 Результаты вычислительных экспериментов	54
§ 2.4 Синтез субоптимальных решений задач обслуживания	55
2.4.1 Эволюционно-генетический алгоритм	57
2.4.2 Алгоритм имитации отжига	59

2.4.3 Алгоритм поиска с запретами	60
2.4.4 Результаты вычислительных экспериментов	62
Выводы по главе 2	63

Глава 3. Модели, бикритериальные задачи и алгоритмы синтеза стратегий обслуживания объектов в однопроцессорной системе с накопительно-расходным компонентом 65

§ 3.1 Задачи обслуживания бинарного множества объектов процессором с накопительно-расходным компонентом	65
3.1.1 Математическая модель	65
3.1.2 Постановки оптимизационных задач	66
3.1.3 Решающие соотношения динамического программирования	67
3.1.4 Результаты вычислительных экспериментов	73
§ 3.2 Задачи обслуживания бинарного потока объектов процессором с накопительно-расходным компонентом	74
3.2.1 Математическая модель	74
3.2.2 Постановки оптимизационных задач	74
3.2.3 Решающие соотношения динамического программирования	75
3.2.4 Пример синтеза стратегии обслуживания	79
3.2.5 Результаты вычислительных экспериментов	83
§ 3.3 Решение задач обслуживания методом ветвей и границ	84
3.3.1 Описание алгоритма	84
3.3.2 Результаты вычислительных экспериментов	88
§ 3.4 Полиномиально разрешимые подклассы задач обслуживания .	89
3.4.1 Концепция $2d$ -стратегий	90
3.4.2 Концепция $2q$ -стратегий	96
3.4.3 Результаты вычислительных экспериментов	99
§ 3.5 Синтез субоптимальных решений задач обслуживания	101
3.5.1 Эволюционно-генетический алгоритм	103
3.5.2 Алгоритм имитации отжига	106
3.5.3 Алгоритм поиска с запретами	107
3.5.4 Результаты вычислительных экспериментов	110
Выводы по главе 3	112

Глава 4. Модели, бикритериальные задачи и алгоритмы синтеза стратегий обслуживания потока объектов в однопроцессорной системе с двумя накопительно-расходными компонентами . . 113

§ 4.1 Математическая модель	113
§ 4.2 Постановки оптимизационных задач	116
§ 4.3 Решающие соотношения динамического программирования . .	117

§ 4.4 Пример синтеза стратегии обслуживания	121
§ 4.5 Алгоритм ветвей и границ	125
§ 4.6 Результаты вычислительных экспериментов	126
Выводы по главе 4	127
Глава 5. Архитектура программных средств синтеза оптимально-компромиссных стратегий обслуживания потока объектов . .	128
§ 5.1 Назначение и функциональные возможности	128
§ 5.2 Описание архитектуры программы	129
§ 5.3 Замечания о повышении эффективности программной реализации	130
Выводы по главе 5	131
Заключение	133
Литература	135
Приложение А. Условия проведения вычислительных экспериментов	148
Приложение Б. Акт внедрения результатов диссертационной работы в ОАО «Салехардский речной порт»	149
Приложение В. Свидетельство о государственной регистрации программы для ЭВМ	151
Приложение Г. Акт внедрения результатов диссертационной работы в учебный процесс Нижегородского государственного университета им. Н.И. Лобачевского	152
Приложение Д. Акт внедрения результатов диссертационной работы в учебный процесс Волжской государственной академии водного транспорта	153

Введение

Растущие требования к экономической эффективности функционирования целого ряда организационно-технических систем предопределяют необходимость повышения качества их транспортно-технологического обеспечения. Принимаемые для его реализации оперативные управленческие решения заключаются в планировании и диспетчеризации процессов обработки (обслуживания) материальных сущностей (объектов) приборами (процессорами). В частности, к такому классу обслуживания относятся рассматриваемые в данной работе процессы типа ГРП — переработки (погрузки, выгрузки) грузов на терминальных комплексах внутреннего водного транспорта. Такие процессы характеризуются высоким темпом изменения оперативной обстановки и, как следствие, достаточно жесткими требованиями, предъявляемыми не только к адекватности информационной среды принятия управляющих решений, но и к скорости формирования проектов таких решений.

Актуальным направлением повышения эффективности управления процессами обслуживания типа ГРП является формирование оперативных диспетчерских планов обслуживания на основе результатов адекватного математического моделирования и решения соответствующим образом поставленных оптимизационных задач за время, приемлемое в условиях конкретного производственного процесса.

Достаточно удобные средства для математического описания процессов обслуживания типа ГРП предоставляет аппарат теории расписаний. Точность такого моделирования определяется выбором шага дискретности пространственных и временных параметров, а синтез оптимальных решений — стратегий (расписаний) обслуживания принципиально осуществим методами дискретного программирования.

Фундаментальным научным исследованиям по теории распи-

саний посвящены труды В.С. Танаева, В.В. Шкурбы, В.С. Гордона, Я.М. Шафранского, Ю.Н. Сотскова, В.А. Струевича, М.Я. Ковалева, А.А. Лазарева, M. Garey, D. Johnson, E.G. Coffman, R.L. Graham, R.W. Conway, W.L. Maxwell, L.W. Miller, R.M. Karp, P. Brucker, M.L. Pinedo.

Применительно к различным задачам управления дискретными ресурсами и, в частности для моделей технологического обслуживания, задачи синтеза оптимальных решений исследовались в работах А.С. Беленького, В.Н. Буркова, Э.Х. Гимади, Д.И. Когана и Ю.С. Федосенко, А.А. Корбута, Д.А. Новикова, Т.П. Подчасовой, Ю.Ю. Финкельштейна, М.Х. Прилуцкого, И.Х. Сигала, М.В. Ульянова, А.В. Шеянова и ряда других авторов.

В контексте данной диссертационной работы отметим также многочисленные исследования, посвященные близким по формальному описанию моделям и оптимизационным задачам диспетчеризации использования ресурсов многоядерных вычислительных систем, выполненные в работах В.А. Костенко, М.Г. Курносова, А.В. Калашникова.

Основное внимание в данной работе уделено оптимизации управления обслуживанием потоков объектов в однопроцессорных моделях с накопительно-расходными компонентами.

Содержательный смысл такого рода моделей проиллюстрируем на примере описания процессов грузовой обработки танкерного флота в условиях Северного завоза [81] через речной порт г.Салехарда. В течение непродолжительного навигационного периода в этот порт крупнотоннажным танкерным флотом по магистральному руслу реки Обь доставляются нефтепродукты (дизельное топливо, авиационный керосин) с нефтеперегонных заводов Западной Сибири. Прибывающие суда в определенной очередности подаются к специализированному терминалу, техническими средствами которого нефтепродукты перекачиваются в соответствующие резервуары для временного хранения. Многочисленные пункты потребления нефтепродуктов располагаются, как правило, по берегам Обской губы и малых рек

прилегающего заполярного региона. Поэтому их доставка в данные пункты осуществляется водным путем из Салехардского порта малотоннажными танкерами ледового класса, загрузка которых также осуществляется на вышеупомянутом специализированном терминале.

В описанной схеме доставки нефтепродуктов конечным потребителям задействованы танкеры, характеризующиеся различными технико-экономическими параметрами. Проблема диспетчеризации рассматриваемых процессов заключается в выработке такой стратегии управления очередностью грузовой обработки поступающих единиц флота, которая обеспечивает сокращение суммарных расходов, обусловленных непроизводительными простоями танкеров в ожидании грузовой обработки, или (и) сокращение продолжительности таких простоев сверх установленных нормативов.

Специфика задач диспетчеризации на внутреннем водном транспорте заключается в том, что между моментом, когда полностью известны исходные данные задачи, и моментом, когда на текущем горизонте планирования следует начать грузовую обработку «первого» судна, проходит относительно небольшой промежуток времени, в течение которого план-график обслуживания должен быть составлен. Соответственно, в диссертационной работе построение и исследование алгоритмов синтеза стратегий обслуживания выполнено с учетом допустимых для них скоростных характеристик.

Целью работы является построение и исследование базовых математических моделей и вычислительных алгоритмов для компьютерных систем поддержки оперативного управления процессами однофазного обслуживания бинарных потоков объектов в однопроцессорных системах с накопительно-расходными компонентами.

Для достижения поставленной цели потребовалось решить следующие **задачи**:

— анализ научных публикаций по теме исследования и существующих

методов решения дискретных оптимизационных задач;

— построение математических моделей обслуживания бинарных потоков объектов стационарным процессором с одним и двумя накопительно-расходными компонентами;

— постановка экстремальных задач синтеза стратегий управления обслуживанием;

— конструирование и программная реализация алгоритмов синтеза оптимальных стратегий управления обслуживанием и оценка их трудоемкости;

— разработка и программная реализация алгоритмов синтеза субоптимальных стратегий управления обслуживанием с приемлемыми для решения практических задач временными характеристиками.

Научная новизна работы состоит в следующих выносимых на защиту основных результатах.

1. Построены базовые математические модели однофазного однократного обслуживания бинарных потоков объектов стационарным процессором с накопительно-расходными компонентами, адекватно описывающие в том числе и типовые схемы грузовой обработки танкерного флота в условиях Северного завоза.

2. Сформулированы однокритериальные и бикритериальные задачи синтеза оптимальных стратегий обслуживания бинарных потоков объектов.

3. Сконструированы решающие алгоритмы, реализующие схемы ветвей и границ и динамического программирования, в том числе в его бикритериальном расширении; получены оценки трудоемкости этих алгоритмов.

4. Для решения поставленных в п. 2 оптимизационных задач повышенной размерности разработаны алгоритмы синтеза субоптимальных решений, основанные на метаэвристических концепциях мягких вычислений.

5. Предложены подклассы моделей, учитывающие естественные с точки зрения приложений ограничения на структуру допустимых стратегий

обслуживания, и разработаны алгоритмы решения соответствующих оптимизационных задач.

Практическая значимость результатов диссертационной работы заключается в том, что разработанные математические модели, решающие алгоритмы и их программные реализации могут применяться в исследовательских, тренажерных и производственных компьютерных системах, предназначенных для решения оптимизационных задач диспетчерского управления обслуживанием бинарных потоков объектов стационарным процессором с накопительно-расходными компонентами.

Апробация работы. Основные результаты диссертационной работы докладывались и обсуждались на следующих научных форумах.

— Международная конференция «Проблемы теоретической кибернетики» (Нижний Новгород, 2011).

— 25th European Conference on Operational Research, EURO'2012 (Vilnius, Lithuania, 2012).

— Нижегородские сессии молодых ученых (Нижний Новгород, 2011, 2012¹).

— Международная научная конференция студентов, аспирантов и молодых ученых «Теоретические и прикладные аспекты кибернетики ТААС» (Киев, Украина, 2011).

— Конкурс «Молодые ученые транспортной отрасли» в рамках V Международного форума «Транспорт России» (Москва, 2011²).

— Международные научно-технические конференции «Информационные системы и технологии» (Нижний Новгород, 2010, 2011, 2012).

— Международная конференция «XI Белорусская математическая конференция» (Минск, Республика Беларусь, 2012).

— Научно-практическая конференция в рамках международного

¹ Доклад удостоен диплома II степени.

² Работа заняла второе призовое место.

научно-промышленного форума «Великие реки» (Нижний Новгород, 2010, 2012³).

— Научная конференция «Технологии Microsoft в теории и практике программирования» (Нижний Новгород, 2010).

— IX Международная молодежная научно-техническая конференция «Будущее технической науки» (Нижний Новгород, 2010).

— Межвузовские научно-практические конференции студентов и аспирантов «Современные тенденции и перспективы развития водного транспорта России» (Санкт-Петербург, 2010, 2011).

— Международная конференция «Водный транспорт России: инновационный путь развития» (Санкт-Петербург, 2010).

— Всероссийская конференция «Высокопроизводительные параллельные вычисления на кластерных системах» (Нижний Новгород, 2011).

Личный вклад автора. Выносимые на защиту научные результаты получены лично соискателем или при его непосредственном участии. Соавторам совместно опубликованных работ принадлежит общая постановка задачи по разработке математических моделей и алгоритмов (научный руководитель), а также реализация вычислительных экспериментов.

В *первой главе* (Модели обслуживания и методы решения задач дискретного программирования) проводится обзор: а) основных вопросов, изучаемых в теории расписаний (§ 1.1); б) моделей обслуживания конечных множеств и детерминированных потоков объектов, в том числе с учетом ограниченности используемых технологических ресурсов (§ 1.2); в) точных и приближенных подходов к решению одно- и многокритериальных задач дискретной оптимизации на основе концепции Парето (§ 1.3).

Вторая глава (Модели, однокритериальные задачи и алгоритмы синтеза стратегий обслуживания объектов в однопроцессорной системе с накопительно-расходным компонентом [41, 44, 46, 48, 49, 52, 53, 56–58, 64, 66,

³ Доклад удостоен диплома I степени.

67]) посвящена рассмотрению базовых математических моделей обслуживания совокупности объектов.

В § 2.1 выполняется построение математической модели обслуживания множества объектов на стационарном процессоре с накопительно-расходным компонентом. Ставятся однокритериальные оптимизационные задачи, для которых доказываются необходимые и достаточные условия существования стратегий обслуживания, описываются алгоритмы решения задач на основе метода динамического программирования. Приводятся оценки трудоёмкости алгоритмов, примеры их реализации и результаты вычислительных экспериментов.

В § 2.2 строится модель обслуживания потока объектов, принципиальное отличие которой от рассмотренной в § 2.1 заключается в учете моментов поступления объектов в очередь на обслуживание. Сформулированы однокритериальные задачи минимизации, выводятся решающие рекуррентные соотношения динамического программирования. Приведены результаты соответствующих вычислительных экспериментов.

В § 2.3 описываются основанные на схеме ветвей и границ алгоритмы решения поставленных в § 2.1, § 2.2 задач и приводятся результаты сравнительных вычислительных экспериментов.

§ 2.4 посвящен алгоритмам синтеза субоптимальных решений поставленных в § 2.1, § 2.2 задач повышенной размерности с использованием следующих метаэвристических концепций мягких вычислений: эволюционно-генетическая парадигма, имитация отжига, поиск с запретами. Приведены результаты вычислительных экспериментов.

В *третьей главе* (Модели, бикритериальные задачи и алгоритмы синтеза стратегий обслуживания объектов в однопроцессорной системе с накопительно-расходным компонентом [40–43, 45, 47, 52, 53, 55, 56, 58–65, 67]) рассмотрено расширение введенных в главе 2 математических моделей на случай двух критериев оценки качества стратегий управления обслужива-

нием.

§ 3.1 посвящен модели обслуживания бинарного множества объектов, в которой каждому объекту ставится в соответствие не одна, а две монотонно возрастающие (в нестрогом смысле) функции индивидуального штрафа. В рамках модели сформулированы три бикритериальные задачи, для решения которых выводятся рекуррентные соотношения алгоритма динамического программирования, приводятся оценки его трудоемкости и результаты массовых вычислительных экспериментов. Реализация алгоритма продемонстрирована на примере.

В § 3.2 формулируется модель обслуживания бинарного потока объектов, отличие которой от рассмотренной в § 3.1 заключается в учете моментов поступления объектов в очередь на обслуживание. В рамках введенной модели сформулированы бикритериальные задачи; для их решения построены рекуррентные соотношения динамического программирования и приведены результаты массовых вычислительных экспериментов.

§ 3.3 посвящен конструированию и оценке быстродействия алгоритмов ветвей и границ для решения рассматриваемых бикритериальных задач.

В § 3.4 рассмотрены модификации модели обслуживания, учитывающие ограничения на допустимые величины опережений в очереди на обслуживание. Данный подход основан на введенных понятиях d - и q -стратегий обслуживания бинарного потока объектов [24]. Для соответствующих задач оптимизации построены алгоритмы решения и проведены массовые вычислительные эксперименты.

§ 3.5 посвящен описанию сконструированных в диссертации алгоритмов решения бикритериальных задач повышенной размерности на основе метаэвристических концепций мягких вычислений: эволюционно-генетической парадигмы, имитации отжига и поиска с запретами. Представлены результаты оценки качества аппроксимации парето-множества синтезируемым метаэвристикой квазипарето-множеством.

В *четвертой главе* (Модели, бикритериальные задачи и алгоритмы синтеза стратегий обслуживания потока объектов в однопроцессорной системе с двумя накопительно-расходными компонентами [52, 53, 64, 67]) построена модель обслуживания бинарного потока объектов, отличающаяся от рассмотренной в § 3.2 наличием двух накопительно-расходных компонентов. Для этой модели доказаны необходимые и достаточные условия непустоты множества допустимых стратегий обслуживания (§ 4.1), сформулированы бикритериальные задачи оптимизации (§ 4.2), сконструированы решающие алгоритмы (§ 4.3 и § 4.5), приведен пример реализации (§ 4.4) и описаны результаты массовых вычислительных экспериментов (§ 4.6).

Пятая глава (Архитектура программных средств синтеза оптимально-компромиссных стратегий обслуживания потока объектов) посвящена изложению назначения и функциональных возможностей программных средств (§ 5.1), а в § 5.2, § 5.3 описывается архитектура и замечания о повышении эффективности программной реализации.

В *заключении* изложены основные научные и практические результаты диссертационной работы.

Приложения содержат описание условий проведения вычислительных экспериментов, а также документы о внедрении и использовании результатов диссертационной работы.

Глава 1. Модели обслуживания и методы решения задач дискретного программирования

Проводится обзор вопросов, изучаемых в теории расписаний. Описываются модели обслуживания конечных множеств и детерминированных потоков объектов, в том числе с учетом использования ограниченных технологических ресурсов. Рассматриваются методы решения задач дискретной оптимизации.

§ 1.1. Элементы теории расписаний

Задачи теории расписаний связаны с построением стратегий (расписаний), т.е. с упорядочиванием некоторых объектов (работ, требований, заявок) по времени и/или по процессорам (приборам, машинам, исполнителям) при различных предположениях о характере обслуживания [86]. При этом зачастую необходимо учитывать ограничения на последовательность обслуживания объектов, ограничения, связанные с процессорами, и т.п. Цель решения таких задач — построение допустимых стратегий, при которых все ограничения соблюдены, или, что является более сложным, — нахождение оптимальной допустимой стратегии по тому или иному критерию оптимальности. Например, построение оптимальной стратегии по быстродействию (т.е. с минимизацией общего времени обслуживания всех объектов), стратегии с минимальными финансовыми затратами и т.п. [11, 120].

Считается, что теория расписания берет свое начало с известной работы Г.Л. Гантта 1903 г. [104], предложившего то, что сегодня называют диаграммами Гантта; такие диаграммы часто используются для представления календарного плана производственного процесса в схематичном виде. Сам термин «теория расписаний» предложил Р. Беллман в 1956 году [94]. С этого и началось активное теоретическое исследование задач в этой новой области. Одними из первых были работы W.E. Smith [122], S.M. Johnson [112] и J.R. Jackson [110], в которых изучались вопросы классификации задач. Наиболее устоявшаяся на нынешний день классификация задач теории рас-

писаний была предложена в работах Р.Л. Грэхема и др. [107].

С появлением статьи Р.М. Карпа [22] большое внимание в работах по теории расписаний стало уделяться не только поиску эффективных алгоритмов решения, но и вопросам вычислительной сложности задач теории расписаний. Достаточно полные обзоры по этой тематике представлены в трудах М. Garey и D. Johnson [17], E.G. Coffman [96, 97], R.L. Graham [107], R.W. Conway, W.L. Maxwell и L.W. Miller [99], M.L. Pinedo [120], P. Brucker [95], E.L. Lawler [117], J.K. Lenstra [117, 118], В.С. Танаева [85–88], В.С. Гордона [15, 16], М.Я. Ковалева [88], А.А. Лазарева [69, 70], Ю.Н. Сотскова [87], В.А. Струсевича [87], Я.М. Шафранского [86, 88].

Применительно к различным проблемам управления дискретными ресурсами, в частности для моделей технологического обслуживания, задачи синтеза оптимальных решений — стратегий (расписаний) обслуживания исследовались в работах Д.И. Батищева [1], А.С. Беленького и Е.В. Левнера [5, 6], В.Н. Буркова [10], Э.Х. Гимади [13], Р.В. Игудина [21], Д.И. Когана и Ю.С. Федосенко [26, 27, 31, 32, 34], А.А. Корбута [37, 38], Д.А. Новикова [36], Т.П. Подчасовой [78], М.Х. Прилуцкого [79], И.Х. Сигала [82, 83], М.В. Ульянова [89, 90], Ю.Ю. Финкельштейна [92] и ряда других авторов.

Содержательно многие задачи теории расписаний являются оптимизационными, т.е. состоят в выборе среди множества допустимых условиями задачи стратегий тех решений, на которых достигается в том или ином смысле оптимальное значение целевой функции. Обычно под оптимальностью понимается минимальное или максимальное значение некоторой функции. В данной диссертационной работе рассматриваются модели, отличающиеся от традиционных следующими двумя особенностями:

- 1) обслуживающий процессор оснащен ограниченными по объёму накопительно-расходными компонентами, которые используются для временного хранения продуктов в процессе обслуживания объектов;
- 2) качество упорядочения оценивается по значениям как одного, так и

двух независимых критериев.

Первая особенность объясняется тем фактом, что в целом ряде транспортно-технологических приложений необходимо учитывать ограниченность резервуара обслуживающего процессора, который служит буфером для промежуточного хранения перевозимого объектами продукта. Близкими по постановке проблемами являются задачи календарного планирования с ограниченными ресурсами, в которых необходимо найти расписание выполнения работ с учетом ограничений предшествования, директивных сроков и ограничений на ресурсы. Исследованиям в области планирования с ограниченными ресурсами посвящены работы Э.Х. Гимади и С.В. Севастьянова [12], А.А. Лазарева, Е.Р. Гафарова и Ф. Вернера [103], A. Janiak и M.-C. Portmann [111] и других авторов.

Вторая особенность обусловлена тем, что в практических задачах одного критерия оценки качества обслуживания зачастую бывает недостаточно и лицу, принимающему решения (ЛПР), приходится учитывать ряд независимых показателей, нередко противоречащих друг другу. С математической точки зрения в данном случае вместо единственной оптимальной стратегии в распоряжение ЛПР должна быть предоставлена полная или достаточно представительная совокупность оптимально-компромиссных стратегий. Фундаментальными исследованиями в области многокритериальной теории расписаний и многокритериальной оптимизации занимались V. T'kindt и J.-Ch. Billaut [123], J.L. Cohon [98], B. Villareal и M. Karwan [125], Д.И. Коган [29, 33], В.В. Подиновский и В.Д. Ногин [73, 77], Д.И. Батищев и Д.Е. Шапошников [2] и ряд других авторов.

§ 1.2. Модели обслуживания конечных множеств и детерминированных потоков объектов

При формализации процессов управления функционированием транспортных, технологических или производственных систем различного на-

значения адекватное математическое описание в ряде случаев может быть выполнено в рамках дискретных моделей обслуживания.

В базовых дискретных динамических моделях, используемых для решения задач управления обслуживанием, основными элементами являются: обслуживающий процессор и конечный детерминированный поток объектов $\{o_1, o_2, \dots, o_n\}$, подлежащих обслуживанию. В каждой модели предполагается, что реализация искомой стратегии начинается в момент времени $t = 0$. Время считается дискретным, а все временные характеристики объектов целочисленными. Обслуживание объекта процессором является однофазным и происходит без прерываний. В каждый момент времени процессор может обслуживать не более одного объекта, а один объект не может обслуживаться одновременно двумя процессорами.

Рассматриваемые модели бывают двух основных типов. В моделях первого типа считается, что по состоянию на момент времени $t = 0$ все объекты готовы к обслуживанию. В моделях второго типа считается, что по состоянию на момент $t = 0$ в систему обслуживания поступила лишь часть объектов, для остальных объектов известен момент времени их поступления в систему. Объект считается готовым к обслуживанию с момента поступления в очередь. Модели первого типа обычно называют моделями обслуживания *множеств* объектов, а второго типа — моделями обслуживания *потоков* объектов. В диссертации рассматриваются модели как первого, так и второго типов.

Базовая модель обслуживания множества объектов

Имеется множество объектов $\{o_1, o_2, \dots, o_n\}$ и процессор, который осуществляет обслуживание этих объектов. Обслуживание каждого объекта реализуется без прерываний от начала до конца; переход от обслуживания одного объекта к обслуживанию следующего за ним объекта затрат времени не требует. Для каждого объекта задается продолжительность обслуживания

процессором τ_i и штраф за единицу времени пребывания в системе обслуживания a_i .

Под стратегией обслуживания подразумевается перестановка $S = (i_1, i_2, \dots, i_n)$ элементов множества $\{1, 2, \dots, n\}$, задающая очередность обслуживания объектов.

При реализации стратегии S величина штрафа по произвольному объекту i оказывается равной $a_{i_k} \cdot \bar{t}(S, i_k)$, где $\bar{t}(S, i_k)$ — момент завершения обслуживания объекта с индексом i_k при реализации стратегии S .

Суммарный штраф определяется выражением

$$\sum_{i=1}^n a_{i_k} \cdot \bar{t}(S, i_k). \quad (1.1)$$

Проблему, заключающуюся в синтезе стратегии, обеспечивающей минимальное значение суммарного штрафа (1.1), обычно называют задачей мастера [35, 80, 85].

Базовая модель обслуживания потока объектов

Система с одним процессором должна обслужить поток объектов $\{o_1, o_2, \dots, o_n\}$, поступающих в дискретном времени. Каждый объект o_i характеризуется тремя целочисленными параметрами: t_i — момент поступления в систему обслуживания; τ_i — число тактов дискретного времени обслуживания процессором; a_i — штраф за такт пребывания объекта в системе обслуживания, $i = \overline{1, n}$. Если обслуживание некоторого произвольного объекта o_i завершается в момент времени $\bar{t}(i)$, то величина индивидуального штрафа по этому объекту считается равной $a_i(\bar{t}(i) - t_i)$. Обслуживание каждого объекта может выполняться без прерываний с любого момента, начиная с его поступления в систему.

Обозначим через $t^*(S, i_k)$ и $\bar{t}(S, i_k)$ моменты начала и завершения обслуживания объекта i_k при реализации стратегии S , $k = \overline{1, n}$. Считается, что реализация стратегии компактна. Суммарный штраф по всем объектам

потока, обслуживаемым согласно расписанию S , определяется выражением

$$\sum_{i=1}^n a_{i_k} \cdot (\bar{t}(S, i_k) - t_{i_k}). \quad (1.2)$$

Проблему, заключающуюся в синтезе стратегии, обеспечивающей минимальное значение суммарного штрафа (1.2), будем называть канонической задачей транспортного типа.

Наличие других, дополнительных, параметров, характеризующих объекты множества или потока, а также процессор, условия функционирования обслуживающей системы и условия принятия управленческих решений, определяется частной спецификой прикладных задач.

Так, в диссертационной работе рассматриваются расширения базовой модели на случай, когда обслуживающий процессор обладает одним или двумя накопительно-расходными компонентами ограниченной ёмкости. Эти компоненты служат буфером для временного хранения перевозимого объектами продукта: каждый объект либо заполняет компонент продуктом, либо забирает из него часть продукта. Таким образом, множество объектов, подлежащих обслуживанию, состоит из двух непересекающихся подмножеств. При этом в силу ограниченности компонента не любая стратегия допустима, поскольку может возникнуть ситуация обслуживания, при которой накопительно-расходный компонент либо переполнится, либо в нем будет недостаточное количество продукта для обслуживания очередного объекта, предусмотренное этой стратегией. Ниже в работе рассматриваются как однокритериальные, так и бикритериальные оптимизационные задачи синтеза допустимых стратегий обслуживания.

§ 1.3. Подходы к решению дискретных задач синтеза стратегий обслуживания

Для решения дискретных задач синтеза стратегий обслуживания наряду с эвристическими приемами используются следующие концепции:

— динамическое программирование [9, 29, 39, 125],

- ветви и границы [28, 115, 119, 124],
- метаэвристики [18, 102, 106, 109, 113].

1.3.1. Концепция динамического программирования

Динамическое программирование — раздел математического программирования, посвященный исследованию многошаговых задач принятия оптимальных решений [7–9, 29]. При этом многошаговость задачи отражает реальное протекание процесса принятия решений во времени, т.е. одновременный выбор значений большого числа переменных решаемой экстремальной задачи заменяется поочередным определением каждой из них в зависимости от возможных обстоятельств. Цель такого представления состоит в сведении исходной задачи высокой размерности к решению на каждом шаге задачи меньшей размерности. Общая схема многошагового процесса принятия оптимальных решений в дискретном времени состоит в следующем.

Пусть имеется некоторая система Ψ , характеризующаяся конечным множеством ее возможных состояний D , известным состоянием x_0 в начальный момент времени и множеством финальных состояний F . В каждом состоянии, кроме финальных $x \in D \setminus F$, делается некоторый выбор (управление) $v \in V(x)$, в результате чего система изменяет свое состояние в соответствии с заданной функцией переходов $f(x, v)$, где $V(x)$ — конечное множество возможных управлений в состоянии x . При этом каждое управление v должно удовлетворять как исходным ограничениям системы Ψ , так и ограничениям, возникающим за счет ранее сделанных выборов. Функция переходов ставит в соответствие каждому состоянию x и возможному в данном состоянии управлению $v \in V(x)$ то состояние $f(x, v) \in D$, в которое перейдет система из x под воздействием v . Каждое управление связано с определенным «штрафом» («платежем»), который оценивается функцией $s(x, v)$. Для применимости схемы динамического программирования необ-

ходимо, чтобы итоговый штраф за все выполненные шаги был равен сумме штрафов на отдельных шагах.

Последовательность допустимых управлений на отдельных шагах назовем траекторией и обозначим $T = \{x^0, x^1, \dots, x^m\}$, при этом $x^j = f(x^{j-1}, v^j)$, где $v^j \in V(x^{j-1})$, $j = \overline{1, m}$. Траекторию $T = \{x^0, x^1, \dots, x^m\}$ назовем *полной*, если её начальным состоянием является начальное состояние системы Ψ , а конечным — одно из финальных состояний F , т.е. $x^0 = x_0$ и $x^m \in F$. Траекторию $T = \{x^0, x^1, \dots, x^m\}$ назовем *заключительной x -траекторией*, если она имеет своим начальным состоянием произвольное состояние x системы, т.е. $x^0 = x$ и $x^m \in F$.

Задача оптимизации заключается в поиске полной траектории системы Ψ , дающей минимальный суммарный платеж. Для её решения могут быть построены рекуррентные соотношения динамического программирования, реализующие принцип оптимальности в формулировке Р. Беллмана [7]: «оптимальное поведение обладает тем свойством, что каковы бы ни были первоначальное состояние и решение в начальный момент, последующие решения должны составлять оптимальное поведение относительно состояния, являющегося результатом первого решения».

В применении к задачам построения оптимальных стратегий обслуживания принцип Беллмана выражается следующим образом [29].

Если полная траектория $T = \{x^0, x^1, x^2, \dots, x_m\}$ оптимальна, то любая ее заключительная часть $T_k = \{x^k, x^{k+1}, \dots, x^m\}$ также оптимальна.

Рекуррентные соотношения, позволяющие определить полную оптимальную траекторию в системе Ψ , представляют собой соотношения для вычисления значений функции $B(x)$, называемой функцией Беллмана. Очевидно, что

$$B(x) = 0, \quad x \in F. \quad (1.3)$$

Пусть x — произвольное нефинальное состояние системы. В этом со-

стоянии можно реализовать любое принадлежащее множеству $V(x)$ управление. Предположим, что выбрано управление v , $v \in V(x)$. Данное управление связано со штрафом $s(x, v)$, а следующим состоянием системы оказывается $f(x, v)$. Если далее реализуется оптимальная заключительная $f(x, v)$ -траектория, то суммарная величина последующих штрафов оказывается равной $B(f(x, v))$. Из принципа Беллмана получаем

$$B(x) = \min_{v \in V(x)} \{s(x, v) + B(f(x, v))\}, \quad x \in D \setminus F. \quad (1.4)$$

Вычисление значений функции Беллмана по соотношениям (1.3), (1.4) выполняется поэтапно в следующем порядке.

На первом этапе фиксируются значения $B(x) = 0$ для всех $x \in F$.

Далее на каждом следующем этапе вычисление очередного значения функции Беллмана выполняется для произвольного состояния x такого, что $B(x)$ неизвестно, но значения $B(y)$ для всех непосредственно следующих за x состояний y уже найдены.

Последним в процессе счета определяется значение $B(x_0)$.

Для синтеза самой оптимальной полной траектории системы при выполнении определяемой соотношениями (1.3), (1.4) процедуры счета дополнительно требуется для каждого состояния x фиксировать запись вида (x, x^*) , где x^* — состояние, в которое переходит система, дающее минимум сумме в правой части (1.4). После вычисления значения функции $B(x_0)$ оптимальная траектория может быть очевидным образом построена по полученному списку пар (x, x^*) .

1.3.2. Концепция ветвей и границ

Метод ветвей и границ основан на идее «разумного» перечисления всех допустимых решений оптимизационной задачи и, по существу, является сокращенным перебором с отсеком подмножеств допустимых решений, заведомо не содержащих оптимальных значений [38, 83, 115]. Метод ветвей и границ был впервые предложен в 1960 году Ленд и Дойг [116] для

решения задач целочисленного программирования.

Реализация схемы ветвей и границ заключается в построении дерева вариантов решений с анализом вершин, получаемых в процессе построения. С целью сокращения количества перебираемых вариантов на каждом шаге вычисляются нижние и верхние оценки целевой функции, определение которых выполняется с учетом специфики решаемой задачи.

К основным понятиям метода ветвей и границ относятся [119]:

- способ ветвления, т.е. способ разбиения всего множества допустимых решений на непересекающиеся подмножества;
- способ получения верхней и нижней оценок в вершинах дерева;
- правило выбора вершины, в которой будет выполняться очередное ветвление;
- правило отсева заведомо бесперспективных ветвлений;
- рекордное значение критерия, соответствующее наилучшему решению, полученному к данному этапу вычислений.

В процессе ветвления поддерживается список задач-кандидатов, на начальном этапе содержащий только исходную задачу G_0 . После выполнения процедуры ветвления список модифицируется: в него вместо выбранной на этапе ветвления задачи G записывается непересекающееся множество подзадач G_i , $i = \overline{1, s}$, на которое разбивается G . Такая модификация выполняется на каждом шаге процесса ветвления. Задача решена, если список задач-кандидатов пуст, т.е. если все они отсеяны по правилам отсева.

Общая схема метода ветвей и границ включает в себя следующие шаги [38, 83].

1. В список задач-кандидатов записывается исходная задача, вычисляются нижняя и верхняя оценки, находится текущий рекорд.
2. Если список пуст, то задача решена, если не пуст — переход к п. 3.
3. Осуществляется выбор подзадачи (вершины) для ветвления.
4. Выполняется ветвление и модифицируется список задач-

кандидатов.

5. Вычисляются верхние и нижние оценки в вершинах, порожденных в результате ветвления, и находится текущий рекорд.

6. Отсеиваются вершины, не содержащие оптимальных решений, и проверяется признак оптимальности.

7. Выполняется переход к п. 2, если оптимальное решение не найдено.

Полученное в результате данной схемы значение текущего рекорда совпадает с оптимальным значением критерия исходной задачи.

1.3.3. Метаэвристические концепции

Подавляющее большинство исследованных задач синтеза стратегий обслуживания множеств и потоков объектов являются *NP*-трудными [17]. Несмотря на это, в приложениях часто встречаются именно такие задачи и для их приближенного решения за приемлемое время широко используются метаэвристические концепции.

К такого рода концепциям относятся: поиск с запретами [106], имитация поведения муравьиных колоний [100], принципы эволюционной генетики [30, 102] и другие. Совокупно алгоритмы, реализующие указанные концепции, принято называть мягкими вычислениями [127].

Эволюционно-генетический алгоритм (Genetic Algorithm) основан на идее моделирования биологической эволюции. Метод получил всеобщее признание после выхода книги «Адаптация в естественных и искусственных системах» Джона Холланда (Jon Holland) [109]. Алгоритм работает не с одним, а с целой популяцией решений. Каждое решение кодируется подобно хромосоме, отдельные гены которой представляют собой отдельные параметры изучаемой задачи. Некоторым, обычно случайным, способом создается множество хромосом особей начальной популяции. Каждая особь оценивается мерой ее «приспособленности», которая является значением целевой функции. Наиболее приспособленные особи получают большую

вероятность «воспроизвести» потомство с помощью скрещивания с другими. Это приводит к появлению новых особей, имеющих характеристики, присущие обоим родителям. Иногда, на этапе формирования нового поколения, генетическим алгоритмом вносится некоторая случайная информация в хромосомы каждой новой особи. Это необходимо для внесения свежего материала, что помогает избежать вырождения, т.е. нахождения локального оптимума вместо глобального. Таким образом, из поколения в поколение хорошие характеристики распространяются на всю популяцию. В итоге алгоритм сходится к оптимальному решению задачи. Процесс эволюции продолжается итеративно, пока не будет выполнен критерий остановки алгоритма.

В основе *алгоритма муравья* (Ant Colony) лежит моделирование поведения муравьев, связанное с их способностью быстро находить кратчайший путь от муравейника к источнику пищи и адаптироваться к изменяющимся условиям. Алгоритмы муравья работают также, как сами муравьи. Смоделированные муравьи-агенты совместно решают проблему и помогают другим муравьям-агентам в дальнейшей оптимизации решения. Само решение проблемы представляет собой кратчайший путь до «источника пищи» — оптимального значения целевой функции. Впервые идея была представлена М. Dorigo в тезисах к его докторской диссертации [100].

Интересный подход, основанный на физических процессах *симуляции восстановления* (Simulated Annealing) был предложен для решения комбинаторных задач [113]. Он основывается на имитации физического процесса восстановления металла после нагревания. В результате получается прочная кристаллическая структура, которая отличается от структуры с дефектами при быстром беспорядочном охлаждении. Структура здесь представляет собой закодированное решение, а температура используется для указания момента принятия решения. Фундаментальная идея заключается в том, что во время работы моделирующего данный процесс алгоритма разрешается

выбирать решения худшего качества, по сравнению с текущим. Это помогает избежать получения локального оптимума в качестве решения, вместо глобального. Вероятность выбора худшего решения уменьшается во время поиска и зависит от текущей температуры, которая также уменьшается.

Очень часто используемым и много цитируемым методом решения задач комбинаторной оптимизации является алгоритм *поиска с запретами* (Tabu Search). Базовая идея этого алгоритма была представлена F. Glover в 1986 году [105]. Описание метода и его концепция была дана F. Glover, M. Laguna в 1997 году [106]. Алгоритм поиска с запретами итеративный, он эффективно использует историю поиска как для того, чтобы избежать локального оптимума, так и для реализации стратегии исследования множества допустимых решений. Концепция поиска с запретами заключается в том, что исследованные на предыдущих итерациях решения запоминаются в особом списке, списке запретов. Учитывая специфику задачи алгоритм запрещает использование на текущем шаге тех решений, которые занесены в этот список. В результате выполнения очередной итерации выбирается лучшее из смежных решений, даже если это решение оказывается хуже текущего.

Все рассмотренные метаэвристические концепции при своей алгоритмической реализации не гарантируют получение оптимального решения, но они вполне применимы для нахождения «хорошего», рационального, решения за приемлемое время.

1.3.4. Подходы к решению задач бикритериальной оптимизации

При решении бикритериальных задач, как правило, невозможно достичь оптимума одновременно по обоим критериям, вследствие этого невозможно найти единственное целесообразное решение. Поэтому в процессе решения таких задач важная роль отводится ЛПР. Именно он определяет выбор решающей процедуры и, если найдено многоэлементное множество

оптимально-компромиссных решений, принимает наиболее подходящее в данной ситуации решение [71].

Для поиска решения задач дискретной многокритериальной оптимизации возможны два подхода.

Первый подход заключается в привлечении имеющейся у ЛПР дополнительной информации и в назначении одной из типовых схем компромисса; реализация каждой из таких схем сводит решение исходной многокритериальной задачи к решению одной или нескольких специальным образом построенных однокритериальных задач. Основное преимущество данного подхода заключается в возможности использования большого набора хорошо изученных методов однокритериальной оптимизации. Среди наиболее часто применяемых схем компромисса отметим метод последовательных уступок, аддитивную свертку критериев, метод главного критерия и метод идеальной точки [20]. Каждая из схем имеет свои особенности и специфику применения. Данный подход не лишен недостатков, поскольку у ЛПР не всегда хватает дополнительной информации, чтобы выбрать наиболее целесообразную схему компромисса и определить необходимые для ее реализации параметры. Кроме того, некоторые схемы компромисса достаточно сложны для реализации.

Второй подход базируется на принятии концепции эффективной оценки и парето-оптимального решения [77]. В данном случае технология решения бикритериальной задачи заключается в построении для нее полной совокупности эффективных оценок с одновременным обеспечением возможности восстановления по любой эффективной оценке парето-оптимального решения ее порождающего. Данная совокупность позволяет ЛПР составить полное представление о задаче и выбрать любое целесообразное для него решение. Вместе с тем, полная совокупность может быть достаточно велика, а ее синтез, как правило, требует больших вычислительных затрат.

В диссертации для построения множества целесообразных решений

использовался именно этот, наиболее общий, подход, основанный на концепции парето-оптимальности [77].

Суть концепции применительно к рассматриваемым в работе бикритериальным задачам обслуживания состоит в следующем.

Критерии оценки $K_1(S)$ и $K_2(S)$ каждой стратегии S образуют вектор-функцию $\mathbf{K}(S)$. Две допустимые стратегии обслуживания считаются эквивалентными, если соответствующие им значения критериев $K_1(S)$ и $K_2(S)$ одинаковы. Обозначим через M произвольное множество двумерных оценок. Оценка $\mathbf{m} = (m_1, m_2)$ из множества M называется недоминируемой, если в множестве M не найдется ни одной оценки $\mathbf{m}' = (m'_1, m'_2)$ такой, что $m'_1 \leq m_1$ и $m'_2 \leq m_2$, причем одно из неравенств является строгим. Недоминируемые оценки именуются эффективными по Парето. Множество всех эффективных оценок — это область компромиссов задачи. Множество всех стратегий, характеризующихся оценками из области компромисса, представляет собой множество парето-оптимальных решений. Очевидно, что любое целесообразное решение многокритериальной задачи должно быть парето-оптимальным.

Процедуру решения бикритериальной задачи можно представить как последовательность выполнения следующих трех этапов.

- 1) Построение полной совокупности эффективных оценок.
- 2) Выбор ЛПР конкретной эффективной оценки.
- 3) Построение стратегии, соответствующей эффективной оценке, выбранной на втором этапе.

Глава 2. Модели, однокритериальные задачи и алгоритмы синтеза стратегий обслуживания объектов в однопроцессорной системе с накопительно-расходным компонентом

Выполняется построение базовых математических моделей обслуживания множества и потока объектов в однопроцессорной системе с накопительно-расходным компонентом. Формулируются однокритериальные задачи оптимизации. Конструируются точные и приближенные решающие алгоритмы синтеза стратегий управления обслуживанием. Приводятся оценки трудоемкости алгоритмов и результаты массовых вычислительных экспериментов.

§ 2.1. Задачи обслуживания бинарного множества объектов процессором с накопительно-расходным компонентом

2.1.1. Математическая модель

Рассматривается множество объектов $\mathcal{O}_n = \{o_1, o_2, \dots, o_n\}$, подлежащих однократному однофазному обслуживанию стационарным процессором с накопительно-расходным компонентом. Для каждого объекта o_i , $i = \overline{1, n}$ определены целочисленные параметры: τ_i – норма длительности обслуживания, d_i – мягкий директивный срок завершения обслуживания, $d_i \geq \tau_i$, v_i – объемная характеристика (вместимость) объекта. С каждым объектом o_i ассоциируется монотонно возрастающая (в нестрогом смысле) функция индивидуального штрафа $\varphi_i(t)$, выражающая величину потерь, зависящую от времени завершения обслуживания объекта.

Множество \mathcal{O}_n обладает свойством бинарности, т.е. состоит из двух подмножеств. Назовем одно из них входящим подмножеством и обозначим его \mathcal{O}^+ , а второе – исходящим и обозначим \mathcal{O}^- . Подмножества обладают следующими свойствами: $\mathcal{O}^+ \cup \mathcal{O}^- = \mathcal{O}_n$ и $\mathcal{O}^+ \cap \mathcal{O}^- = \emptyset$. Принадлежность объекта o_i , $i = \overline{1, n}$ тому или иному подмножеству определяется значением параметра w_i ($w_i = +1$, если $o_i \in \mathcal{O}^+$; $w_i = -1$, если $o_i \in \mathcal{O}^-$).

Процессор готов к обслуживанию множества объектов в начальный момент времени $t = 0$. Обслуживание объекта o_i , $i = \overline{1, n}$ осуществляется процессором без прерываний; необслуженный объект не может покинуть очередь; одновременное обслуживание процессором двух и более объектов и его непроизводительные простои запрещены.

Накопительно-расходный компонент процессора представляет собой резервуар объемом V^* . В момент времени t объемная величина его заполнения характеризуется значением переменной V_t (V_0 – величина заполнения в начальный момент времени $t = 0$). В результате обслуживания объекта o_i из подмножества \mathcal{O}^+ (\mathcal{O}^-) заполнение резервуара увеличивается (уменьшается) на величину v_i . Как очевидно, обслуживание объекта o_i из входящего подмножества \mathcal{O}^+ возможно при наличии необходимого свободного объема в резервуаре, т.е. при выполнении условия $V_t + v_i \leq V^*$; обслуживание любого объекта o_i из исходящего подмножества \mathcal{O}^- возможно при наличии достаточного количества продукта в резервуаре, т.е. при выполнении условия $V_t - v_i \geq 0$. Назовем эти условия *объёмными ограничениями* на обслуживание объектов.

Стратегия обслуживания объектов S представляет собой произвольную перестановку $S = \{i_1, i_2, \dots, i_n\}$ совокупности индексов $N = \{1, 2, \dots, n\}$; при её реализации объект с индексом i_k обслуживается k -м по очереди, $k = \overline{1, n}$. Стратегию S именуем допустимой, если выполняются отмеченные выше объёмные ограничения на обслуживание объектов. Совокупность всех допустимых стратегий обозначим как множество Ω .

Здесь и ниже будем считать истинным всегда выполняющиеся в практических приложениях неравенства

$$2 \cdot v_i \leq V^*, i = \overline{1, n}. \quad (2.1)$$

При выполнении этих неравенств, имеет место теорема.

Теорема 2.1. *Необходимым и достаточным условием непустоты*

множества Ω является выполнение следующих неравенств:

$$0 \leq V_0 + \sum_{i=1}^n w_i \cdot v_i \leq V^*. \quad (2.2)$$

Доказательство. Условия теоремы являются необходимыми. Действительно, если $V_0 + \sum_{i=1}^n w_i \cdot v_i < 0$, то при любой стратегии система окажется в состоянии, при котором для обслуживания очередного объекта из исходящего подмножества \mathcal{O}^- недостаточно продукта в резервуаре и больше не осталось объектов, способных пополнить резервуар. Аналогично, если $V_0 + \sum_{i=1}^n w_i \cdot v_i > V^*$, то при любой стратегии система окажется в состоянии, при котором обслуживание следующего объекта из входящего подмножества \mathcal{O}^+ невозможно, так как в резервуаре недостаточно места и не осталось объектов, способных забрать часть продукта.

Перейдем к доказательству достаточности. Допустим, при соблюдении введенных условий множество допустимых решений задачи пусто. Это означает, что в процессе обслуживания возникла ситуация, когда невозможно обслужить любой из оставшихся объектов.

Обозначим через M множество индексов оставшихся необслуженными объектов. В этом случае реализуется одна из следующих трех ситуаций.

1) В множестве M присутствуют только индексы объектов входящего подмножества \mathcal{O}^+ , и в результате обслуживания любого из оставшихся объектов резервуар переполнится, т.е. $V_t + v_j > V^*$ для каждого $j \in M$. Но это противоречит выполнению неравенств (2.2).

2) В множестве M присутствуют только индексы объектов исходящего подмножества \mathcal{O}^- , и в результате обслуживания любого из оставшихся объектов резервуар переполнится, т.е. $V_t - v_j < 0$ для каждого $j \in M$. Но это также противоречит выполнению неравенств (2.2).

3) В множестве M присутствуют как индексы входящего \mathcal{O}^+ , так и исходящего \mathcal{O}^- подмножеств. В результате обслуживания любого из объек-

тов множества \mathbf{M} резервуар либо переполнится, либо обслуживание будет невозможно из-за отсутствия нужного количества продукта в резервуаре. С учетом истинности условий (2.1) в первом случае будет иметь место соотношение $V^* - V_t < v_j \leq V^*/2$, $j \in \mathbf{M}$ и, следовательно, $V_t > V^*/2$. А во втором — соотношение $V_t < v_j \leq V^*/2$, $j \in \mathbf{M}$ и, следовательно, $V_t < V^*/2$. Получаем противоречие.

Таким образом, предположение о пустоте множества допустимых решений при соблюдении условий (2.2) ложно. Теорема доказана. \square

2.1.2. Постановки оптимизационных задач и исследование их вычислительной сложности

Для дальнейшего введем обозначения $t^*(i_k, S)$ и $\bar{t}(i_k, S)$ — соответственно моменты начала и завершения обслуживания объекта с индексом i_k при реализации стратегии S . Считаем, что момент начала обслуживания первого объекта в стратегии S равен 0. Поскольку обслуживание каждого последующего объекта возможно лишь после завершения обслуживания предыдущего, то между введенными временными характеристиками имеют место соотношения

$$\begin{aligned} t^*(i_1, S) &= 0; \quad t^*(i_k, S) = \bar{t}(i_{k-1}, S), k = \overline{2, n}; \\ \bar{t}(i_k, S) &= t^*(i_k, S) + \tau_{i_k}, k = \overline{1, n}. \end{aligned}$$

Наибольший интерес с позиций повышения эффективности управления обслуживанием множества объектов \mathcal{O}_n для рассматриваемой в данном параграфе математической модели представляют следующие задачи.

Задача Set1R(\sum). Построить стратегию обслуживания S , минимизирующую значение суммы индивидуальных штрафов $\varphi_i(t)$ по всем объектам множества \mathcal{O}_n :

$$\min_{S \in \Omega} \left(\sum_{k=1}^n \varphi_{i_k}(\bar{t}(i_k, S)) \right). \quad (2.3)$$

Задача Set1R(max). Построить стратегию обслуживания S , минимизирующую значение максимального из индивидуальных штрафов $\varphi_i(t)$

по всем объектам множества \mathcal{O}_n :

$$\min_{S \in \Omega} (\max_{1 \leq k \leq n} (\varphi_{i_k}(\bar{t}(i_k, S))))). \quad (2.4)$$

Замечание 2.1. Отметим две важные для прикладных приложений интерпретации задач $\text{Set1R}(\sum)$ и $\text{Set1R}(\max)$.

1) При линейных функциях индивидуального штрафа

$$\varphi_i(t) = a_i \cdot (t - \tau_i), i = \overline{1, n} \quad (2.5)$$

и заданном для каждого объекта коэффициенте a_i стратегия обслуживания будет оцениваться по суммарному штрафу за простои объектов множества \mathcal{O}_n в ожидании обслуживания. Возникающая при этом задача $\text{Set1R}(\sum)$ минимизации максимального из индивидуальных штрафов

$$\min_{S \in \Omega} (\sum_{k=1}^n a_{i_k} \cdot (\bar{t}(i_k, S) - \tau_{i_k}))$$

является частным случаем задачи (2.3).

2) В моделях с предписанными объектам директивными сроками функции индивидуального штрафа имеют вид

$$\varphi_i(t) = \max\{t - d_i, 0\}, i = \overline{1, n}, \quad (2.6)$$

а стратегия обслуживания оценивается максимальным по продолжительности нарушением директивного срока завершения обслуживания среди всех объектов множества \mathcal{O}_n . Тогда задачу $\text{Set1R}(\max)$, являющуюся частным случаем задачи (2.4), можно записать в виде

$$\min_{S \in \Omega} \{ \max_{1 \leq k \leq n} [\max(\bar{t}(i_k, S) - d_{i_k}, 0)] \}.$$

Заметим, что задача $\text{Set1R}(\sum)$ с линейными функциями штрафа вида (2.5) является NP -трудной, этот результат был доказан в работе [25].

Теорема 2.2. Задача $\text{Set1R}(\max)$ с функциями штрафа вида (2.6) является NP -трудной.

Доказательство. В доказательстве этой теоремы используем NP -полноту задачи Разбиение [17]: известны натуральные числа p_1, p_2, \dots, p_n ,

причем сумма этих чисел должна быть четным числом, т.е. $\sum_{i=1}^n p_i = 2U$. Спрашивается: можно ли в совокупности индексов H выделить подмножество J такое, что $\sum_{i \in J} p_i = \sum_{i \in H \setminus J} p_i = U$?

Построим задачу Set1R(max) следующим образом.

Начальное заполнение ёмкости резервуара V_0 полагаем равным $2U$. Считаем объём резервуара V^* равным $2U$. Обслуживанию подлежат объекты o_1, o_2, \dots, o_n , принадлежащие исходящему подмножеству \mathcal{O}^- . Назовем эти объекты обыкновенными; для них полагаем $w_i = -1$, $\tau_i = v_i = p_i$, $d_i = T$, ($i = \overline{1, n}$), где d_i — мягкий директивный срок завершения обслуживания, а $T = 2U + A$.

Считаем, что во входящем подмножестве \mathcal{O}^+ имеется также объект o_{n+1} . Назовем этот объект особым; для него полагаем $w_{n+1} = +1$, $v_{n+1} = U$, $d_{n+1} = 0$, $\tau_{n+1} = A$, где A — константа.

Очевидно, что для минимизации значения критерия необходимо обслужить особый объект как можно раньше. Следовательно, оптимальная стратегия обслуживания S будет состоять из трех частей.

В первой части будут обслужены обыкновенные объекты; соответствующую совокупность индексов объектов обозначим через Λ . Очевидно, что $\sum_{i \in \Lambda} \tau_i = \sum_{i \in \Lambda} p_i = U + \varepsilon$, где ε — целая неотрицательная константа. Штраф $K^I(S)$ по всем объектам в первой части стратегии равен нулю.

Во второй части стратегии будет обслужен особый объект, и соответствующий штраф $K^{II}(S)$ определяется суммой $A + U + \varepsilon$.

В третьей части стратегии будут обслужены оставшиеся обыкновенные объекты и штраф $K^{III}(S)$ в этой части равен нулю.

В результате максимальный из индивидуальных штрафов $K(S) = \max(0, A + U + \varepsilon, 0)$ определяется как сумма $A + U + \varepsilon$.

Как следует из предыдущего, величина $K(S)$ не превышает значения $A + U$ тогда и только тогда, когда $\varepsilon = 0$, а это возможно, если первая часть

стратегии длится ровно U единиц времени, т.е. $\sum_{i \in \Lambda} \tau_i = U$. Данное равенство реализуемо в том и только в том случае, когда исходная задача Разбиение имеет положительное решение. Построенное сведение выполняется за полиномиальное время. Теорема доказана. \square

2.1.3. Решающие соотношения динамического программирования

Выполним решение задач $\text{Set1R}(\sum)$ и $\text{Set1R}(\max)$ алгоритмами, основанными на концепции динамического программирования.

Алгоритм решения задачи $\text{Set1R}(\sum)$

При обслуживании объектов множества \mathcal{O}_n решения принимаются в те моменты времени, когда процессор свободен и необходимо выбрать следующий для обслуживания объект из входящего или исходящего подмножества, учитывая текущее заполнение резервуара. Соответственно тройка значений (t, Q, V_t) определяет текущее состояние системы в момент t принятия решения, где Q — множество индексов объектов, еще не обслуженных в этот момент времени, V_t — текущее заполнение накопительно-расходного компонента.

Пусть $E(t, Q, V_t)$ — минимальная величина суммарного штрафа, полученная за период времени от момента t и до окончания процесса обслуживания всех объектов множества \mathcal{O}_n . Тогда минимальное значение суммы индивидуальных штрафов в решаемой задаче $\text{Set1R}(\sum)$ будет представлять собой величину $E(0, \mathcal{O}_n, V_0)$.

В текущем состоянии (t, Q, V_t) системы обслуживания объект с индексом из входящего подмножества \mathcal{O}^+ может быть допущен к обслуживанию только при условии выполнения неравенства $V_t + v_\alpha \leq V^*$, а объект с индексом из исходящего подмножества \mathcal{O}^- может быть обслужен, если $V_t - v_\beta \geq 0$. Обозначим через Q^* совокупность индексов объектов множества \mathcal{O}_n , допустимых к обслуживанию в состоянии (t, Q, V_t) , $Q^* \subseteq Q$.

Очевидно, что для любого $\theta \geq 0$ и $Q = \{\alpha\}$, где α — индекс допустимого к обслуживанию объекта множества \mathcal{O}_n , имеет место соотношение

$$E(t_n + \theta, \{\alpha\}, V_{t_n + \theta}) = \varphi_\alpha(t_n + \theta + \tau_\alpha). \quad (2.7)$$

Если в состоянии (t, Q, V_t) в качестве очередного обслуживаемого объекта выбран объект с индексом $\alpha \in Q^*$, то следующим моментом принятия решения будет $t + \tau_\alpha$; выбор индекса следующего объекта для обслуживания будет осуществляться из множества $Q \setminus \{\alpha\}$; при этом характеристика заполнения компонента определяется как $V_t + w_\alpha \cdot v_\alpha$. Суммируя приведенные рассуждения, запишем рекуррентное выражение

$$E(t, Q, V_t) = \min_{\alpha \in Q^*} [\varphi_\alpha(t + \tau_\alpha) + E(t + \tau_\alpha, Q \setminus \{\alpha\}, V_t + w_\alpha \cdot v_\alpha)], \quad (2.8)$$

образующее совместно с (2.7) замкнутую систему решающих задачу Set1R(\sum) соотношений динамического программирования.

Для пояснения процесса решения задачи введем оператор \mathcal{R} раскрытия состояния: если (t, Q, V_t) является произвольным состоянием системы, то $\mathcal{R}(t, Q, V_t)$ — множество состояний, непосредственно следующих за состоянием (t, Q, V_t) . Последовательно раскрывая все состояния системы обслуживания от начального, в конечном итоге получим совокупность финальных состояний системы. Этот процесс можно представить как построение конечного взвешенного ориентированного графа, вершины которого взаимно однозначно соответствуют состояниям системы (t, Q, V_t) , дуги — объектам, выбранным на обслуживание из множества Q , веса дуг — значениям величины $E(t, Q, V_t)$. Пронумеруем вершины графа неотрицательными целыми числами и присвоим начальному состоянию номер 0. Будем последовательно нумеровать все получаемые уникальные вершины-состояния в порядке их порождения оператором \mathcal{R} . При этом заметим, что одно и то же состояние может быть получено путем раскрытия разных состояний, а порядковый номер соответствует первому раскрытию.

Процесс решения задачи на основе построенных соотношений (2.7) и

(2.8) состоит из трех последовательных этапов.

На первом этапе выполняется разметка для определения достижимых состояний системы. Фиксируются финальные состояния, соответствующие завершению процесса обслуживания всех объектов потока \mathcal{O}_n . Таким состояниям соответствует величина $E(t, Q, V_t) = 0$.

На втором этапе выполняется расчет очередных значений критерия для произвольных достижимых состояний. Эти состояния характеризуются тем, что значения критерия для них неизвестны, но для всех непосредственно следующих за ними состояний значения критерия уже известны; состояние является непосредственно следующим за некоторым текущим состоянием, если в него можно перейти из текущего, выбрав на обслуживание любой объект с индексом из совокупности Q . Так, в начале второго этапа известны только величины критерия для финальных состояний системы обслуживания. Последним в процессе решения задачи $\text{Set1R}(\Sigma)$ определяется минимальное значение суммы индивидуальных штрафов для начального состояния системы $E(0, \mathcal{O}_n, V_0)$.

Для синтеза оптимальной стратегии на втором этапе дополнительно строится список переходов. В этот список для каждой рассчитываемой величины $E(t, Q, V_t)$ вносится запись (m, α, z, z^*) , где:

m — значение суммарного штрафа;

z — номер текущего состояния, получаемый в результате применения оператора \mathcal{R} ;

$\alpha \in Q$ — индекс объекта, обслуживаемого в текущем состоянии (t, Q, V_t) при выборе которого величина $E(t, Q, V_t)$ будет равна m ;

z^* — номер непосредственно следующего состояния (полагаем его равным \emptyset , если состояние (t, Q, V_t) финальное), получаемый в результате применения оператора \mathcal{R} .

На третьем, последнем этапе работы алгоритма с помощью построенного списка переходов последовательно синтезируется стратегия обслу-

живания, соответствующая минимальной величине суммарного штрафа m_0 . Для этого в списке отыскивается запись вида $(m_0, \alpha_0, 0, z_0^*)$. Второй элемент этой записи α_0 представляет собой индекс объекта, который при реализации искомой стратегии обслуживается первым. Следующим за состоянием с номером 0 будет состояние с номером z_0^* . В списке переходов отыскивается запись с третьим элементом равным $z_1 = z_0^*$: $(m_1, \alpha_1, z_1, z_1^*)$; при этом второй элемент записи α_1 представляет собой индекс объекта, который при реализации искомой стратегии выбирается вторым.

Действуя описанным выше образом, последовательно получаем соответствующую значению штрафа m_0 оптимальную стратегию. Процесс синтеза заканчивается отысканием записи финального состояния системы с четвертой компонентой равной \varnothing .

Дадим верхнюю оценку числа элементарных операций, выполняемых при вычислениях по соотношениям (2.7) и (2.8) в процессе решения задачи $\text{Set1R}(\sum)$. Эта оценка зависит от числа состояний, на которых подсчитываются величины $E(t, Q, V_t)$, и сложности вычисления значения каждой этой функции.

В рассматриваемых состояниях (t, Q, V_t) число значений аргумента t сверху ограничено величиной $H = \sum_{i=0}^n \tau_i$, а Q является произвольным подмножеством индексов объектов. Таким образом, число состояний системы не превышает $H2^n$.

Число элементарных операций, выполняемых для нахождения каждого очередного значения функции $E(t, Q, V_t)$, в свою очередь, определяется двумя компонентами: числом операций, выполняемых при выборе из множества Q подмножества допустимых в данном состоянии объектов Q^* , и сложностью подсчета минимального значения. Первая компонента не превышает n элементарных операций, а вторая мажорируется линейной функцией от n .

Таким образом, верхней оценкой числа реализуемых алгоритмом опе-

раций является величина $O(H2^n n^2)$. Полученная оценка трудоемкости процедуры синтеза эффективной оценки для задачи $\text{Set1R}(\sum)$ экспоненциальна.

Алгоритм решения задачи $\text{Set1R}(\max)$.

Рассуждая аналогично, как при построении рекуррентных соотношений для решения задачи $\text{Set1R}(\max)$, получим следующие рекуррентные соотношения динамического программирования:

$$E(t_n + \theta, \{\alpha\}, V_{t_n + \theta}) = \varphi_\alpha(t_n + \theta + \tau_\alpha), \quad (2.9)$$

$$E(t, Q, V_t) = \min_{\alpha \in Q^*} \{ \max[\varphi_\alpha(t + \tau_\alpha), E(t + \tau_\alpha, Q \setminus \{\alpha\}, V_t + w_\alpha \cdot v_\alpha)] \}. \quad (2.10)$$

Как очевидно, число реализуемых алгоритмом операций для решения задачи $\text{Set1R}(\max)$ оценивается такой же величиной, как и при решении задачи $\text{Set1R}(\sum)$, $O(H2^n n^2)$.

2.1.4. Пример синтеза стратегии обслуживания

Иллюстрацию технологии расчета оптимальной стратегии обслуживания по рекуррентным соотношениям динамического программирования приведем на примере решения задачи $\text{Set1R}(\sum)$.

Пример 2.1. Требуется построить стратегию обслуживания S , минимизирующую значение суммы индивидуальных штрафов вида (2.5) по всем объектам множества \mathcal{O}_4 при характеристиках объектов, приведенных в таблице 2.1. Максимальная вместимость резервуара $V^* = 19$ с заполнением $V_0 = 10$ в начальный момент времени.

Таблица 2.1

Исходные данные

i	τ_i	a_i	v_i	w_i
1	3	2	5	1
2	4	3	6	1
3	2	5	2	-1
4	4	6	7	-1

По ходу решения будем составлять список переходов, в который вносятся записи описанного выше типа (Таблица 2.2).

Полная совокупность эффективных оценок в этой задаче будет:
 $E(0, \{1, 2, 3, 4\}, 10) = \min[0 + E(3, \{2, 3, 4\}, 15), 0 + E(4, \{1, 3, 4\}, 16), 0 + E(2, \{1, 2, 4\}, 8), 0 + E(4, \{1, 2, 3\}, 3)]$.

Присвоим начальному состоянию $(0, \{1, 2, 3, 4\}, 10)$ номер 0, а остальным состояниям $(3, \{2, 3, 4\}, 15)$, $(4, \{1, 3, 4\}, 16)$, $(2, \{1, 2, 4\}, 8)$ и $(4, \{1, 2, 3\}, 3)$ номера 1, 2, 3 и 4 соответственно.

Таблица 2.2

Список переходов

№	(m, α, z, z^*)	№	(m, α, z, z^*)
1	$(54, 4, 7, \varnothing)$	8	$(56, 4, 10, 12)$
2	$(27, 2, 8, \varnothing)$	9	$(60, 3, 11, 12)$
3	$(57, 4, 5, 8)$	10	$(76, 3, 2, 10)$
4	$(55, 3, 9, \varnothing)$	11	$(38, 2, 13, 12)$
5	$(62, 3, 6, 10)$	12	$(50, 4, 3, 13)$
6	$(72, 3, 1, 5)$	13	$(58, 3, 4, 13)$
7	$(20, 1, 12, \varnothing)$	14	$(50, 3, 0, 3)$

Далее по тексту каждое новое состояние системы, порождаемое оператором \mathcal{R} , будет получать следующий порядковый номер. В таблице 2.3 приведены все рассмотренные при решении примера 2.1 состояния.

Продолжение работы алгоритма заключается в последовательном нахождении значений $E(3, \{2, 3, 4\}, 15)$, $E(4, \{1, 3, 4\}, 16)$, $E(2, \{1, 2, 4\}, 8)$ и $E(4, \{1, 2, 3\}, 3)$ с помощью соотношений (2.7) и (2.8).

Если в состоянии $E(3, \{2, 3, 4\}, 15)$ обслужить объект с индексом 2, то резервуар переполнится и текущее заполнение будет равно 21. Следовательно, $E(3, \{2, 3, 4\}, 15) = \min[15 + E(5, \{2, 4\}, 13), 18 + E(7, \{2, 3\}, 8)]$.

Используя формулу (2.8), получаем $E(5, \{2, 4\}, 13) = \min[15 + E(9, \{4\}, 19), 30 + E(9, \{2\}, 6)]$, а из формулы (2.7) следует $E(9, \{4\}, 19) = 54$, $E(9, \{2\}, 6) = 27$. Поместим записи $(54, 4, 7, \varnothing)$ и $(27, 2, 8, \varnothing)$ в список переходов.

Теперь находим $E(5, \{2, 4\}, 13) = \min[15 + 54, 30 + 27] = 57$. Поместим запись $(57, 4, 5, 8)$ в список переходов. Далее определяем $E(7, \{2, 3\}, 8) = \min[21 + E(11, \{3\}, 14), 35 + E(9, \{2\}, 6)]$. Используя формулу (2.7), вычислим $E(11, \{3\}, 14) = 55$ и поместим запись $(55, 3, 9, \varnothing)$ в список переходов. Состояние $(9, \{2\}, 6)$ уже было ранее рассмотрено и имеет номер 8, при этом $E(9, \{2\}, 6) = 27$.

Таблица 2.3

Список состояний

№	(t, Q, V_t)	№	(t, Q, V_t)
0	$(0, \{0, 1, 2, 3\}, 10)$	7	$(9, \{4\}, 19)$
1	$(3, \{2, 3, 4\}, 15)$	8	$(9, \{2\}, 6)$
2	$(4, \{1, 3, 4\}, 16)$	9	$(11, \{3\}, 14)$
3	$(2, \{1, 2, 4\}, 8)$	10	$(6, \{1, 4\}, 14)$
4	$(4, \{1, 2, 3\}, 3)$	11	$(8, \{1, 3\}, 9)$
5	$(5, \{2, 4\}, 13)$	12	$(10, \{1\}, 7)$
6	$(7, \{2, 3\}, 8)$	13	$(6, \{1, 2\}, 1)$

Таким образом, $E(7, \{2, 3\}, 8) = \min[21 + 55, 35 + 27] = 62$, и, следовательно, запись $(62, 3, 6, 10)$ вносится в список переходов. Далее получаем $E(3, \{2, 3, 4\}, 15) = \min[15 + 57, 18 + 62] = 72$ и вносим запись $(72, 3, 1, 5)$ в список переходов.

Действуя аналогичным образом, находим значения $E(4, \{1, 3, 4\}, 16) = 76$, $E(2, \{1, 2, 4\}, 8) = 50$ и $E(4, \{1, 2, 3\}, 3) = 58$. В итоге получаем оптимальное значение критерия $E(0, \{1, 2, 3, 4\}, 10) = 50$.

Рис. 2.1 содержит взвешенный ориентированный граф, построенный в процессе решения задачи. Вершины графа взаимно однозначно соответствуют номерам состояний системы, полученных в результате выполнения оператора \mathcal{R} ; дуги графа соответствуют объектам, выбранным на обслуживание из множества Q в каждом конкретном состоянии (на рис. 2.1 — числа в скобках); веса дуг графа соответствуют значениям величины $E(t, Q, V_t)$.

Переходя к построению стратегии обслуживания, соответствующей полученной оценке 50, заметим, что она находится в записи $(50, 3, 0, 3)$,

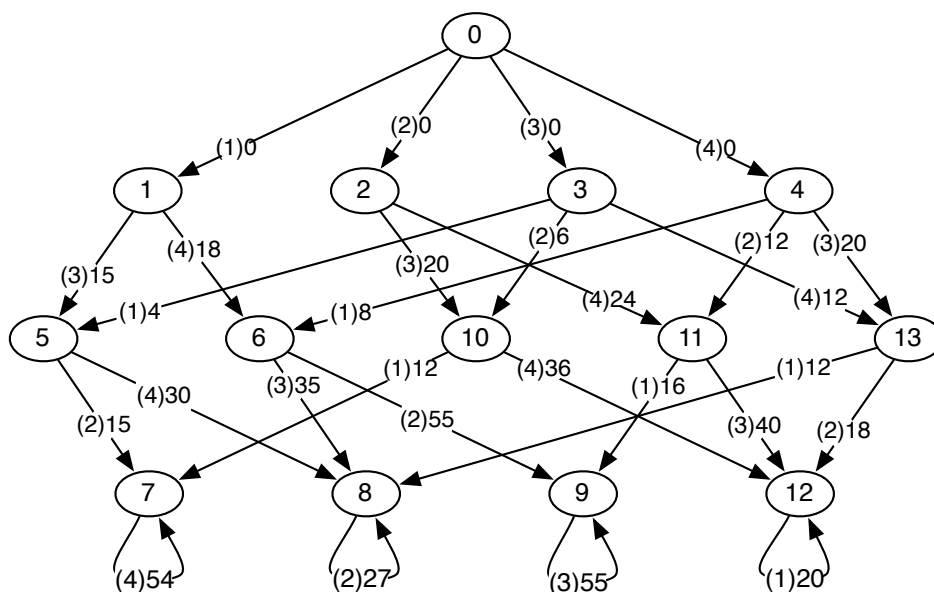


Рис. 2.1. Граф решения задачи.

соответствующей начальному состоянию с номером 0. Второй элемент записи равен 3, и, следовательно, объект с индексом 3 будет обслуживаться первым в стратегии. Следующим состоянием будет состояние с номером 3. Находим в списке переходов соответствующую запись (50, 4, 3, 13), второй элемент которой равен 4, и, следовательно, вторым будет обслуживаться объект с индексом 4. Текущая оценка 50 в состоянии с номером 3 получена из состояния с номером 13. Соответствующая запись в списке переходов будет (38, 2, 13, 12), и, следовательно, третьим будет обслуживаться объект с индексом 2. Следующая запись соответствует оценке в состоянии с номером 12, в списке переходов это финальное состояние (20, 1, 12, \varnothing), т.е. последним обслуживается объектом с индексом 1. В итоге получаем следующую стратегию обслуживания объектов:

$$\{3, 4, 2, 1\}.$$

2.1.5. Результаты вычислительных экспериментов

В целях определения временных характеристик алгоритмов и границ целесообразности их применения была проведена серия вычислительных

экспериментов в условиях, описанных в приложении А.

Для рассматриваемого класса задач существенным показателем качества решающего алгоритма является скорость его отработки, которая в выполненных вычислительных экспериментах измерялась с точностью до тысячной доли секунды. Результаты экспериментов представлены в таблице 2.4, где приняты следующие обозначения: «EX» — среднее время решения задачи алгоритмом полного перебора; «DP (Set1R(\sum))» и «DP (Set1R(max))» — временные характеристики алгоритмов решения задач Set1R(\sum) Set1R(max) соответственно; t_{\min} — минимальное время решения; t_{\max} — максимальное время решения; t_{avg} — среднее время решения задачи.

Таблица 2.4

Зависимости длительностей отработки алгоритмов
решения задач от размерности потока объектов \mathcal{O}_n , с

n	EX	DP (Set1R(\sum))			DP (Set1R(max))		
	t_{avg}	t_{avg}	t_{\min}	t_{\max}	t_{avg}	t_{\min}	t_{\max}
8	0.031	0.002	0.000	0.016	0.002	0.000	0.016
9	0.250	0.017	0.000	0.063	0.017	0.000	0.063
10	3.015	0.138	0.015	0.500	0.155	0.031	0.594
11	36.610	1.296	0.234	5.265	1.310	0.132	6.077
12	430.734	11.197	2.20	32.281	13.063	2.078	39.074
13	5321.698	128.068	8.281	294.196	134.152	9.719	386.719
14	-	836.514	12.658	2221.092	963.534	16.845	2684.452

Разработанные алгоритмы позволяют выполнить синтез стратегий обслуживания множества объектов в количестве до 14 единиц за промежуток времени в среднем от нескольких секунд до 15 минут. Такие временные интервалы приемлемы для синтеза стратегий обслуживания объектов в системах транспортного типа.

§ 2.2. Задачи обслуживания бинарного потока объектов процессором с накопительно-расходным компонентом

2.2.1. Математическая модель

Рассматривается n -элементный поток независимых объектов $\mathcal{O}_n = \{o_1, o_2, \dots, o_n\}$, подлежащих однократному однофазному обслуживанию стационарным процессором с накопительно-расходным компонентом. При этом для каждого объекта определен целочисленный параметр t_i — момент поступления в очередь на обслуживание; объекты пронумерованы в порядке поступления в систему обслуживания, т.е. $0 \leq t_1 \leq \dots \leq t_n$. Остальные характеристики и параметры потока, процессора и накопительно-расходного компонента принимаются такими же, как и в модели п. 2.1.1. Напомним эти характеристики и параметры:

★ поток \mathcal{O}_n обладает свойством бинарности (т.е. состоит из двух подпотоков — «входящего» \mathcal{O}^+ и «исходящего» \mathcal{O}^- таких, что $\mathcal{O}^+ \cup \mathcal{O}^- = \mathcal{O}_n$ и $\mathcal{O}^+ \cap \mathcal{O}^- = \emptyset$);

★ τ_i — норма длительности обслуживания;

★ d_i — директивный срок завершения обслуживания, $d_i \geq \tau_i + t_i$;

★ v_i — объемная характеристика (вместимость) объекта;

★ w_i — параметр принадлежности объекта o_i тому или иному подпотоку ($w_i = +1$, если $o_i \in \mathcal{O}^+$ и $w_i = -1$, если $o_i \in \mathcal{O}^-$);

★ $t^*(i_k, S)$ и $\bar{t}(i_k, S)$ — соответственно моменты начала и завершения обслуживания объекта с индексом i_k при реализации стратегии S .

2.2.2. Постановки оптимизационных задач

Считаем, что момент начала обслуживания первого объекта в стратегии S равен моменту его поступления. Так же, как и в модели п. 2.1.1, обслуживание каждого последующего объекта возможно лишь после завершения обслуживания предыдущего, но в рассматриваемой модели не может начаться раньше момента его поступления в систему. Соответственно, меж-

ду введенными временными характеристиками имеют место соотношения

$$\begin{aligned} t^*(i_1, S) &= t_{i_1}; \quad t^*(i_k, S) = \max(\bar{t}(i_{k-1}, S), t_{i_k}), \quad k = \overline{2, n}; \\ \bar{t}(i_k, S) &= t^*(i_k, S) + \tau_{i_k}, \quad k = \overline{1, n}. \end{aligned} \quad (2.11)$$

Оптимизационные задачи, по смыслу аналогичные (2.3), (2.4), записываются в следующем виде.

Задача Flow1R(\sum). Построить стратегию обслуживания S , минимизирующую значение суммы индивидуальных штрафов $\varphi_i(t)$ по всем объектам потока \mathcal{O}_n :

$$\min_{S \in \Omega} \left(\sum_{k=1}^n \varphi_{i_k}(\bar{t}(i_k, S)) \right). \quad (2.12)$$

Задача Flow1R(max). Построить стратегию обслуживания S , минимизирующую значение максимального из индивидуальных штрафов $\varphi_i(t)$ по всем объектам потока \mathcal{O}_n :

$$\min_{S \in \Omega} \left(\max_{1 \leq k \leq n} (\varphi_{i_k}(\bar{t}(i_k, S))) \right). \quad (2.13)$$

Замечание 2.2. Важными для прикладных приложений частными видами функций индивидуального штрафа в задачах Flow1R(\sum) и Flow1R(max) являются следующие.

1) **Линейные функции**

$$\varphi_i(t) = a_i \cdot (t - \tau_i - t_i), \quad i = \overline{1, n} \quad (2.14)$$

при заданном для каждого объекта коэффициенте a_i ; в этом случае стратегия обслуживания будет оцениваться по суммарному штрафу за простои объектов множества \mathcal{O}_n в ожидании обслуживания.

2) **Кусочно-линейные функции, учитывающие возможность нарушения предписанных директивных сроков обслуживания**

$$\varphi_i(t) = \max\{t - d_i, 0\}, \quad i = \overline{1, n}; \quad (2.15)$$

в этом случае стратегия обслуживания будет оцениваться максимальным по продолжительности нарушением директивного срока завершения обслуживания среди всех объектов множества \mathcal{O}_n .

Очевидно, что задачи $\text{Flow1R}(\sum)$ и $\text{Flow1R}(\max)$ являются NP -трудными даже при линейных функциях штрафа вида (2.14) и (2.15), так как они являются обобщениями задач $\text{Set1R}(\sum)$ и $\text{Set1R}(\max)$.

2.2.3. Решающие соотношения динамического программирования

Выполним решение задач $\text{Flow1R}(\sum)$ и $\text{Flow1R}(\max)$ алгоритмами, основанными на идеологии динамического программирования.

Алгоритм решения задачи $\text{Flow1R}(\sum)$

Введем следующие обозначения:

★ $F(t)$ — определяемое исходными данными подмножество индексов объектов потока \mathcal{O}_n , поступивших в систему обслуживания в момент времени t ;

★ $D(t, \delta)$ — совокупность индексов объектов потока \mathcal{O}_n , поступающих в систему в дискретные моменты времени $[t + 1, \dots, t + \delta]$, $\delta \geq 1$; очевидно, что $D(t, \delta) = \bigcup_{i=1}^{\delta} F(t + i)$.

При обслуживании объектов потока \mathcal{O}_n решения принимаются в те моменты времени, когда процессор свободен и необходимо выбрать следующий объект из входящего \mathcal{O}^+ или исходящего \mathcal{O}^- подпотока, учитывая текущее заполнение резервуара. Следовательно, тройка значений (t, Q, V_t) определяет текущее состояние системы в момент t принятия решения, где Q — множество индексов объектов потока \mathcal{O}_n , ожидающих обслуживания в этот момент времени.

В множество Q считается включенным индекс 0 фиктивного объекта с параметрами $t_0 = t$, $\tau_0 = \min\{\delta | D(t, \delta) \neq \emptyset\}$, $a_0 = 0$, $d_0 = \infty$. Выбор фиктивного объекта на обслуживание означает, что процессор будет простаивать, начиная от момента времени t принятия решения до момента поступления в обслуживающую систему очередного объекта. Фиктивный объект исключается из множества Q , если $t \geq t_n$, т.е. когда все объекты потока \mathcal{O}_n поступили в систему и ожидают своего обслуживания.

Пусть $E(t, Q, V_t)$ — минимальная величина суммарного штрафа, полученная за период времени от момента t и до окончания процесса обслуживания всех объектов потока \mathcal{O}_n ; $E(0, F(0), V_0)$ — минимальное значение суммарного штрафа в исходной задаче Flow1R(Σ).

Обозначим через Q^* множество индексов объектов, допустимых к обслуживанию с учетом объемных ограничений в состоянии (t, Q, V_t) , $Q^* \subseteq Q$.

Очевидно, что для любого $\theta \geq 0$ и $Q = \{\alpha\}$, где α — индекс ожидающего обслуживания объекта, имеет место соотношение

$$E(t_n + \theta, \{\alpha\}, V_{t_n + \theta}) = \varphi_\alpha(t_n + \theta + \tau_\alpha). \quad (2.16)$$

Если в состоянии (t, Q, V_t) в качестве очередного обслуживаемого объекта выбран объект с индексом $\alpha \in Q^*$, то следующим моментом принятия решения будет $t + \tau_\alpha$; множество индексов объектов, ожидающих обслуживания в этот момент времени, определяется как совокупность $(Q \setminus \{\alpha\}) \cup D(t, \tau_\alpha)$. Поэтому

$$E(t, Q, V_t) = \min_{\alpha \in Q^*} \{ \varphi_\alpha(t + \tau_\alpha) + E(t + \tau_\alpha, (Q \setminus \{\alpha\}) \cup D(t, \tau_\alpha), V_t + w_\alpha \cdot v_\alpha) \}. \quad (2.17)$$

Соотношения (2.16) и (2.17) суть решающие задачу Flow1R(Σ) рекуррентные выражения.

Процесс решения задачи на основе построенных соотношений аналогичен описанному в п. 2.1.3.

Дадим верхнюю оценку числа элементарных операций, выполняемых при вычислениях по соотношениям (2.16) и (2.17) в процессе решения задачи Flow1R(Σ). Эта оценка зависит от числа состояний, на которых подсчитываются значения функции $E(t, Q, V_t)$, и сложности вычисления каждого отдельного значения этой функции. В рассматриваемых состояниях (t, Q, V_t) число значений аргумента t сверху ограничено величиной $L = t_n + \sum_{i=1}^n \tau_i$, а Q является произвольным подмножеством индексов из

числа прибывших на момент времени t объектов с учетом наличия фиктивного объекта. Таким образом, число состояний системы не превышает $L2^{n+1}$. Число элементарных операций, выполняемых для нахождения каждого очередного значения функции $E(t, Q, V_t)$, в свою очередь определяется двумя факторами:

- не превышающим n числом операций, выполняемых при выборе из множества Q подмножества допустимых в данном состоянии объектов Q^* ;
- мажорируемой линейной функцией от n сложностью подсчета минимального значения.

Таким образом, верхней оценкой числа реализуемых алгоритмом операций является $O(L2^{n+1}n^2)$. Полученная оценка трудоемкости процедуры синтеза эффективной оценки для задачи $\text{Flow1R}(\sum)$ экспоненциальна.

З а м е ч а н и е 2.3. Верхняя оценка числа элементарных операций, выполняемых для решения задачи $\text{Flow1R}(\sum)$, больше, чем для решения задачи $\text{Set1R}(\sum)$, так как $L > H$ ($L = t_n + \sum_{i=1}^n \tau_i$, $H = \sum_{i=1}^n \tau_i$) и при решении задачи $\text{Set1R}(\sum)$ нет необходимости учитывать фиктивный объект при синтезе стратегии обслуживания.

Алгоритм решения задачи $\text{Flow1R}(\max)$

Рассуждая аналогично, как и при построении рекуррентных соотношений для задачи $\text{Flow1R}(\sum)$, получим следующие решающие соотношения динамического программирования:

$$E(t_n + \theta, \{\alpha\}, V_{t_n + \theta}) = \varphi_\alpha(t_n + \theta + \tau_\alpha), \quad (2.18)$$

$$E(t, Q, V_t) = \min_{\alpha \in Q^*} \{ \max[\varphi_\alpha(t + \tau_\alpha), E(t + \tau_\alpha, (Q \setminus \{\alpha\}) \cup D(t, \tau_\alpha), V_t + w_\alpha \cdot v_\alpha)] \}. \quad (2.19)$$

Верхняя оценка числа реализуемых алгоритмом операций при решении задачи $\text{Flow1R}(\max)$ по соотношениям (2.18), (2.19) выражается такой же величиной, как для задачи $\text{Flow1R}(\sum)$.

2.2.4. Результаты вычислительных экспериментов

Для оценки быстродействия отработки алгоритмов $\text{Flow1R}(\sum)$ и $\text{Flow1R}(\max)$ была проведена серия массовых вычислительных экспериментов, результаты которых представлены в таблице 2.5.

Таблица 2.5

Зависимости длительностей отработки алгоритмов решения задач от размерности потока объектов \mathcal{O}_n , с

n	DP ($\text{Flow1R}(\sum)$)			DP ($\text{Flow1R}(\max)$)		
	t_{avg}	t_{min}	t_{max}	t_{avg}	t_{min}	t_{max}
8	0.001	0.000	0.016	0.001	0.000	0.016
9	0.002	0.000	0.031	0.003	0.000	0.031
10	0.013	0.000	0.234	0.019	0.000	0.250
11	0.074	0.000	0.937	0.107	0.000	1.177
12	0.484	0.000	5.328	0.664	0.000	8.903
13	3.851	0.000	44.234	4.806	0.000	63.488
14	29.219	0.016	263.199	53.514	0.016	321.092
15	263.964	0.016	1506.349	313.476	0.016	2178.237

Разработанные алгоритмы позволяют выполнить синтез оптимальных стратегий обслуживания потока объектов в количестве до 15 единиц за промежутки времени в среднем от нескольких секунд до 10 минут.

§ 2.3. Решение задач обслуживания методом ветвей и границ

2.3.1. Описание алгоритма

Не уменьшая общности, конкретизацию схемы алгоритма ветвей и границ рассмотрим на примере решения задачи $\text{Flow1R}(\sum)$. Для задачи $\text{Flow1R}(\max)$ рассуждения можно повторить с учётом специфики вычисления оптимизируемого критерия.

Обозначим через \mathcal{Z} однокритериальную задачу вида (2.3). Так как задача синтеза стратегии обслуживания сводится к нахождению перестановки совокупности индексов n объектов, то дерево вариантов решений для задачи \mathcal{Z} — это дерево, каждое ребро которого фиксирует индекс очередного обслуживаемого объекта.

Для каждой вершины N дерева вариантов однозначно определен путь, ведущий из корня в эту вершину. Число составляющих этот путь ребер будем именовать *рангом вершины* и обозначать $r(N)$. При этом корень дерева решений — это вершина с рангом 0. Вершине N соответствует частная задача $\mathcal{Z}(N)$, отличающаяся от исходной задачи \mathcal{Z} тем, что очередность обслуживания для первых $r(N)$ объектов уже определена. Вершину с рангом $r(N) = n - 1$ назовём *конечной* вершиной дерева, так как в этом случае остается единственный объект на обслуживание и нет необходимости в дальнейшем ветвлении. Путь от корня к конечной вершине дерева полностью определяет допустимое решение задачи.

Процесс ветвления состоит в выборе на первом уровне дерева ветвления первого объекта на обслуживание, на следующем уровне — объекта обслуживаемого вторым и т.д. При ветвлении дополнительно необходимо учитывать объемные ограничения на обслуживание объектов. Обозначим через $P(N) = \{p(1), p(2), \dots, p(r(N))\}$ стратегию, которая фиксируется путем из корня дерева в вершину N . Тогда величина заполнения емкости резервуара $V(N)$ в вершине N однозначно определяется выражением
$$V(N) = V_0 + \sum_{j=1}^{r(N)} w_{p(j)} \cdot v_{p(j)}.$$
 При ветвлении из вершины N ребро для объекта с индексом α , $o_\alpha \in O^+$ строится только в том случае, если $V(N) + v_\alpha \leq V^*$; для объекта с индексом β , $o_\beta \in O^-$ ребро строится только для случая $V(N) \geq v_\beta$.

При выполнении ветвления из вершины N строится не более, чем $(n - r(N))$ рёбер, порождающих не более $(n - r(N))$ новых вершины, ранг которых $r(N) + 1$.

При анализе каждой получаемой вершины N рассчитываются две оценки: *верхняя* (примем для нее обозначение $H(N)$) и *нижняя* (примем для нее обозначение $L(N)$). Верхняя оценка порождается допустимым решением задачи, а нижняя характеризует оптимальное решение в задаче $\mathcal{Z}(N)$ как

некую «надежду».

В процессе построения дерева решения определяющую роль играет текущий рекорд R , которой формируется из наименьшей верхней оценки. Наличие рекорда позволяет исключать заведомо неперспективные направления поиска оптимальной стратегии обслуживания.

Обозначим через $\mathbf{M}(N)$ множество индексов объектов, для которых еще не определена очередность обслуживания: $|\mathbf{M}(N)| = n - r(N)$. Пусть \bar{T} — момент времени окончания обслуживания объектов согласно стратегии $P(N)$. *Облегченной* назовем задачу, отличие которой от частной задачи $\mathcal{Z}(N)$ состоит в том, что одновременное обслуживание двух и более объектов разрешено и процессор может начать обслуживание объекта с индексом $l \in \mathbf{M}(N)$ в любой момент времени t , $t \geq t_l \geq \bar{T}$. Тогда нижняя оценка рассчитывается по формуле

$$L(N) = \sum_{j \notin \mathbf{M}(N)} \varphi_j(\bar{t}(j, P(N))) + \sum_{l \in \mathbf{M}(N)} \varphi_l(\max(\bar{T}, t_l) + \tau_l). \quad (2.20)$$

Отметим, что в общем случае нижняя оценка является недостижимой.

Обозначим через $P'(N) = \{p'(1), p'(2), \dots, p'(n - r(N))\}$ стратегию обслуживания объектов с индексами входящими в множество $\mathbf{M}(N)$, эта стратегия такая, что объекты в ней обслуживаются в порядке поступления, но при этом учитываются объемные ограничения. При условии, что объекты множества $\mathbf{M}(N)$ обслуживаются в соответствии со стратегией $P'(N)$ и обслуживание может быть начато в момент \bar{T} , верхняя оценка рассчитывается по формуле

$$H(N) = \sum_{j \notin \mathbf{M}(N)} \varphi_j(\bar{t}(j, P(N))) + \sum_{l \in \mathbf{M}(N)} \varphi_l(\bar{t}(l, P'(N))). \quad (2.21)$$

На любой стадии построения дерева решений необходимо *выбирать вершину, в которой будет выполняться ветвление*. При этом совокупность всех вершин дерева делится на три группы.

Первая группа — закрытые вершины, это вершины, в которых ветвле-

ние уже выполнено.

Вторая группа — терминальные вершины, отброшенные по описанным ниже правилам отсева; к терминальным также относятся конечные вершины.

Не входящие в список закрытых или терминальных вершин входят в третью группу и именуются открытыми вершинами.

Для каждого очередного ветвления выбирается открытая вершина N с наименьшим значением верхней оценки $H(N)$; если таких вершин несколько, из них выбирается вершина наибольшего ранга — поиск в глубину.

Для *исключения бесперспективных для дальнейшего ветвления вершин* в дереве решений будем пользоваться следующими правилами.

1) Если для приписанной данной вершине нижней оценки выполняется условие $L(N) \geq R$, то вершину можно поместить в группу терминальных вершин.

2) В качестве дополнительного условия отсева открытых вершин воспользуемся следующим определением: если в некоторый момент можно показать, что наилучший потомок вершины Y не хуже наилучшего потомка вершины X , то будем говорить, что Y *доминирует* над X и X можно исключить из дальнейшего рассмотрения [119].

Предположим, что в результате решения мы получили две вершины Y и X такие, что множества индексов объектов, уже включенных в стратегии обслуживания, одинаковы, а стратегии $P(Y)$ и $P(X)$ различны. Следовательно, для формирования оставшейся части стратегии необходимо обслужить одно и то же множество объектов. Если в вершине Y значение текущего момента времени не больше, чем в X , и значение критерия не больше, чем в вершине X , то нет смысла продолжать решение из X ; лучшие результаты будут достигнуты, если продолжать решение из Y .

Следовательно, вершина X может быть перенесена из группы открытых в группу терминальных раньше, чем будут получены какие-нибудь пол-

ные решения.

Опишем алгоритм BnB решения задачи \mathcal{Z} методом ветвей и границ, используя введенные выше построения.

Шаг 0. Вершине-корню N_0 ставится в соответствие задача $\mathcal{Z}(N_0)$, которая совпадает с исходной; в ней же вычисляются верхняя $H(N_0)$ и нижняя $L(N_0)$ оценки; текущему рекорду R присваивается значение $H(N_0)$.

Шаг 1. Если в построенном фрагменте дерева вариантов нет открытых вершин, то задача решена, иначе переходим к шагу 2.

Шаг 2. Осуществляется выбор очередной вершины N для ветвления.

Шаг 3. Выполняется ветвление в вершине N , в результате которого получаем $(n - r(N))$ новых открытых вершин; вершина N переходит в группу закрытых.

Шаг 4. В каждой новой вершине определяются нижняя $L(N)$ и верхняя $H(N)$ оценки, вычисляемые по формулам (2.20) и (2.21) соответственно; каждая вновь полученная верхняя оценка сравнивается с R : из двух значений выбирается меньшее для текущего значения рекорда.

Шаг 5. Выполняется отсев заранее бесперспективных направлений поиска оптимальной стратегии, соответствующие вершины переходят в группу терминальных.

Шаг 6. Осуществляется переход к шагу 1.

Полученное по окончанию реализации данной алгоритмической схемы значение текущего рекорда R совпадает с оптимальным значением критерия исходной задачи \mathcal{Z} .

Максимальное число вершин в дереве ветвления равно $n!$, но, как показывают массовые вычислительные эксперименты, число просматриваемых вершин при решении прикладных задач обычно существенно меньше.

З а м е ч а н и е 2.4. Схема алгоритма ветвей и границ для решения задачи $\text{Flow1R}(\max)$ аналогична описанной выше за исключением способа расчета верхней и нижней оценок, которые рассчитываются по следующим

формулам соответственно:

$$L(N) = \max\left\{\max_{j \notin M(N)} \varphi_j(\bar{t}(j, P(N))), \max_{l \in M(N)} \varphi_l(\max(\bar{T}, t_l) + \tau_l)\right\},$$

$$H(N) = \max\left\{\max_{j \notin M(N)} \varphi_j(\bar{t}(j, P(N))), \max_{l \in M(N)} \varphi_l(\bar{t}(l, P'(N)))\right\}.$$

З а м е ч а н и е 2.5. *Описанная схема алгоритма ветвей и границ для решения задач Flow1R(\sum) и Flow1R(max) может быть использована также для решения задач Set1R(\sum) и Set1R(max) с учетом специфики расчета нижней и верхней оценок.*

2.3.2. Результаты вычислительных экспериментов

С целью выяснения эффективности применения алгоритма ветвей и границ и сравнения его с алгоритмом динамического программирования была выполнена для задачи Flow1R(\sum) серия массовых вычислительных экспериментов в условиях, описанных в приложении А.

Эксперименты для алгоритмов VnB и DP выполнялись на одних и тех же наборах исходных данных. При этом в дополнение к параметрам t_{\min} , t_{\max} , t_{avg} определялись такие показатели, как q_{avg} — среднее количество вершин в дереве решений и g_{avg} — среднее количество вершин, в которых выполнялось ветвление. В таблице 2.6 представлены характеристики каждого из алгоритмов DP и VnB.

Полученные результаты сравнительных экспериментов позволяют сделать следующий вывод.

Метод ветвей и границ в среднем является более эффективным по сравнению с методом динамического программирования при любом значении величины размерности потока n ; эффективность тем выше, чем больше число объектов в решаемой задаче.

Также представленные экспериментальные данные позволяют сделать вывод о том, что число вершин дерева ветвления обычно существенно мень-

Таблица 2.6

Значения показателей быстродействия алгоритма BnB
при решении задачи Flow1R(Σ)

n	DP	BnB				
	t_{avg}, c	t_{avg}, c	t_{min}, c	t_{max}, c	q_{avg}	g_{avg}
8	0.001	0.000	0.000	0.016	178	37
9	0.002	0.000	0.000	0.016	346	65
10	0.013	0.000	0.000	0.032	695	120
11	0.074	0.001	0.000	0.032	1281	203
12	0.484	0.005	0.000	0.094	2328	341
13	3.851	0.023	0.000	0.625	4822	655
14	29.219	0.099	0.000	3.922	9172	1165
15	263.964	0.499	0.016	16.781	19189	2298
16	-	1.353	0.000	31.671	33009	3688
17	-	7.568	0.000	96.885	71104	7506
18	-	32.589	0.000	1090.998	107122	10804
19	-	449.359	0.000	8122.050	246244	23383

ше максимально возможного, что свидетельствует о достаточной эффективности в среднем применения алгоритма BnB.

З а м е ч а н и е 2.6. Не представленные здесь результаты массовых вычислительных экспериментов для задачи Flow1R(max) имеют аналогичный порядок.

§ 2.4. Синтез субоптимальных решений задач обслуживания

Решение задач синтеза стратегий обслуживания путем реализации алгоритмов, сконструированных в параграфах § 2.1–§ 2.3, требует не всегда допустимых в производственных условиях временных затрат. Поэтому существенный интерес представляет разработка таких алгоритмов, которые, не гарантируя построения точных решений, все же позволяют за допустимое производственным регламентом время синтезировать стратегии обслуживания приемлемого для практического использования качества [93]. Будем называть такие стратегии субоптимальными.

Данный параграф посвящен алгоритмам решения задач Set1R(Σ),

Set1R(max), Flow1R(\sum) и Flow1R(max) повышенной размерности с использованием следующих метаэвристических концепций мягких вычислений: эволюционно-генетическая парадигма, имитация отжига, поиск с запретами. Для их описания введем следующие обозначения.

В качестве особи для эволюционно-генетического алгоритма или текущего решения в алгоритмах имитации отжига и поиска с запретами будем считать произвольную перестановку $\{i_1, i_2, \dots, i_n\}$ совокупности индексов объектов потока \mathcal{O}_n , которая определяет стратегию S обслуживания. Данная стратегия оценивается следующими критериями:

- ★ $\sum_{k=1}^n \varphi_{i_k}(\bar{t}(i_k, S))$ для задач Set1R(\sum) и Flow1R(\sum);
- ★ $\max_{1 \leq k \leq n} (\varphi_{i_k}(\bar{t}(i_k, S)))$ для задач Set1R(max) и Flow1R(max).

При описании алгоритмов синтеза субоптимальных стратегий будем обозначать как $K(S)$ критерий вне зависимости от рассматриваемой задачи, но имея в виду при этом, что способ его вычисления специфичен для каждой из них.

Под операцией *восстановления допустимости стратегии* S будем понимать выполнение следующей процедуры (при этом через S^* будем обозначать стратегию, в которой восстановлена допустимость решения).

1. Положить $l = 1$ — индекс текущего рассматриваемого элемента.
2. Проверить по порядку для всех еще нерассмотренных в $S = \{i_1, i_2, \dots, i_n\}$ индексов объектов α выполнение объёмных ограничений: если ограничения выполняются, то перед переходом к п. 3 индекс α поместить на позицию l в стратегии S^* , исключить индекс α из дальнейшего рассмотрения, увеличить l на единицу.
3. Если рассмотрены все индексы перестановки S , закончить восстановление допустимости стратегии, иначе вернуться к п. 2.

2.4.1. Эволюционно-генетический алгоритм

Эволюционно-генетический алгоритм GA (Genetic Algorithm) — это алгоритм поисковой оптимизации, который начинается с начальной популяции особей P^0 (совокупности кодировок решений $\{S_1, S_2, \dots, S_{N_p}\}$), и итеративно выполняет следующий цикл операций¹:

- ★ вычисление функций приспособленности для всех особей $S_j \in P^t$;
- ★ выбор из популяции P^t , $t = 0, 1, \dots$ репродукционного множества R^t ;
- ★ генерация из репродукционного множества R^t новых особей с помощью комбинаций операций копирования, скрещивания и мутации;
- ★ формирование на очередном шаге нового поколения P^{t+1} .

Применительно к задачам Set1R(\sum), Set1R(max), Flow1R(\sum) и Flow1R(max) структура генетического алгоритма может быть описана следующим образом.

1. Инициализация и оценка начальной популяции
 $P^0 = \{S_1, S_2, \dots, S_{N_p}\}$ численностью N_p .

Сгенерировать случайным образом хромосомы N_p особей фиксированной длины n , которые представляют собой допустимые стратегии обслуживания совокупности объектов, $S_j \in \Omega$, $j = \overline{1, N_p}$.

Оценить каждую особь популяции с помощью функции приспособленности, в качестве которой принимается значение оценочного критерия $K(S_j)$, $j = \overline{1, N_p}$.

2. Выбор лучшей хромосомы.

В следующую популяцию всегда переходит хромосома с наименьшим значением функции приспособленности.

3. Воспроизведение $k_1(N_p - 1)$ потомков согласно жадному оператору скрещивания.

¹Используемая здесь и ниже терминология соответствует принятой в этой области мягких вычислений [3, 4, 18, 109].

Случайным образом выбирается «разрезающая» точка. Далее происходит копирование левого сегмента одного родителя в потомка. Остальные позиции потомка берутся от другого родителя по правилу жадного алгоритма: на каждую следующую позицию выбирается индекс объекта, обслуживание которого приводит к наименьшему значению критерия, исключая те индексы, которые уже присутствуют в потомке. Выполняется восстановление допустимости решения.

Второй потомок генерируется подобным образом.

В следующую популяцию перейдет потомок с меньшим значением функции приспособленности [14].

4. Воспроизводство $k_2(N_p - 1)$ потомков согласно оператору мутации.

Случайно выбирается позиция в родительской хромосоме, значение гена в этой позиции обменивается на значение рядом расположенного гена, выполняется восстановление допустимости решения. Потомок перейдет в следующую популяцию, если значение его функции приспособленности меньше, чем у родителя; иначе в следующую популяцию копируется родительская хромосома.

5. Воспроизводство $k_3(N_p - 1)$ потомков согласно оператору инверсии.

Случайным образом выбирается «разрезающая» точка в родительской хромосоме. Новая хромосома формируется из родителя путем инверсии сегмента, который расположен справа после «разрезающей» точки. Выполняется восстановление допустимости решения. Потомок перейдет в следующую популяцию, если значение его функции приспособленности меньше, чем у родителя; иначе в следующую популяцию копируется родительская хромосома [14].

6. Замена текущей популяции P^t новой популяцией P^{t+1} .

В новую популяцию переходят:

★ выбранная в п. 2 особь,

- ★ $k_1(N_p - 1)$ особей, которые появились в результате скрещивания;
- ★ $k_2(N_p - 1)$ особей, которые появились в результате мутации;
- ★ $k_3(N_p - 1)$ особей, появившихся в результате инверсии.

Считаем константы k_1 , k_2 и k_3 заданными и при этом $k_1 + k_2 + k_3 = 1$ и $0 \leq k_l \leq 1, l = \{1, 2, 3\}$.

Таким образом, каждая популяция P^t имеет постоянную численность N_p .

7. Сменить номер текущего поколения $t = t + 1$ и проверить условие останова алгоритма.

Критерием останова является заданное число генераций или неизменность хромосомы, выбранной на этапе 2 селекции на протяжении заданного числа генераций.

Если критерий останова выполнен, то хромосома с наименьшим значением функции приспособленности будет соответствовать субоптимальному решению задачи.

2.4.2. Алгоритм имитации отжига

Алгоритм отжига или симуляции восстановления SA (Simulated Annealing) построен на основе моделирования физического процесса нагрева и последующего охлаждения некоторой субстанции с целью получения прочной кристаллической структуры [18, 113]. Алгоритм отжига включает в себя пять следующих последовательно выполняемых этапов.

1 этап. В качестве начального решения S_0 выбирается случайно сгенерированное допустимое решение задачи оптимизации. Для этого стратегия сначала инициализируется последовательностью чисел $\{1, 2, \dots, n\}$, а затем выполняется n случайных перестановок; восстанавливается допустимость решения. Задается начальное значение температуры $T = T_0$.

2 этап. Оценивается решение S_j , заключающееся в декодировании стратегии и расчете значения критерия $K(S_j)$.

3 этап. Выполняется случайный поиск решения. Для этого текущее решение S_j копируется в рабочее S'_j , затем рабочее решение произвольно модифицируется. Чтобы сохранить его целостность после модификации, два элемента в рабочей стратегии произвольно переставляются и выполняется восстановление допустимости решения. Полученное рабочее решение оценивается.

4 этап. Осуществляется выбор текущего решения с учетом значения критерия допуска, вычисляемого по формуле $P(K(S'_j)) = e^{-K(S'_j)/T}$. На данном этапе имеется два решения: текущее S_j и модифицированное рабочее S'_j . С каждым из них связана определенная энергия — значения критериев $K(S_j)$ и $K(S'_j)$ соответственно. Если рабочее решение имеет меньшую энергию, чем текущее, то рабочее решение копируется в текущее и алгоритм переходит к этапу снижения температуры. В противном случае проверяется выполнение неравенства $P(K(S'_j)) > R$, где R — случайно выбранное число из диапазона $(0, 1)$. Если неравенство выполняется, то перед переходом к этапу 5 рабочее решение копируется в текущее.

5 этап. Осуществляется снижение температуры по закону $T_{i+1} = \alpha T_i$, где $\alpha < 1$. При одном значении температуры выполняется заранее установленное количество итераций; при этом на каждой итерации сохраняется лучшее решение задачи. Процесс продолжается до тех пор, пока температура не достигнет нуля.

По завершению работы алгоритма текущее решение S_j будет содержать стратегию с достигнутым качеством обслуживания бинарного потока объектов.

2.4.3. Алгоритм поиска с запретами

В основе метода поиска с запретами TS (Taboo Search) лежит схема, позволяющая осуществлять поиск глобального оптимума, не останавливаясь в локальном оптимуме; при этом основной идеей является формирова-

ние в процессе поиска так называемого списка запретов, который блокирует исследование части или всей окрестности текущего решения [106].

Под окрестностью $\mathcal{N}(S_j)$ решения S_j будем понимать $\mathcal{N}(S_j) = \{(i'_1, i'_2, \dots, i'_n) : \exists f, k \in \{1, \dots, n\} : i'_f = i_k, i'_k = i_f, i'_m = i_m \text{ для } \forall m \in \{1, \dots, n\}, m \neq f, k\}$ — множество стратегий, полученных из текущего решения S_j перестановкой двух соседних элементов.

На j -м шаге алгоритма осуществляется переход из текущего решения S_j в соседнее S'_j , которому соответствует минимальное среди всех соседних значений критерия $K(S'_t)$, т.е. $K(S_{j+1}) = \min\{K(S'_j) \mid S'_j \in \mathcal{N}(S_j)\}$. Для того чтобы алгоритм не остановился в точке локального оптимума, решение S_j удаляется из дальнейшего рассмотрения. Алгоритм хранит такие решения за последние l шагов и на каждом следующем шаге запрещает движение в этих направлениях. Упорядоченный список $\Phi_j = \{S_j, S_{j-1}, \dots, S_{j-l+1}\}$ будем называть *списком запретов*. Список запретов удаляет из окрестности $\mathcal{N}(S_j)$ не более l точек; оставшаяся часть этой окрестности представляет собой множество $\mathcal{N}(S_j, \Phi_j)$ незапрещенных точек.

Пусть K — минимальное найденное в процессе решения значение критерия. Тогда с учетом введенных выше пояснений схема алгоритма поиска с запретами может быть представлена следующим образом.

1. Выбрать в качестве начального решения стратегию $S_0 = \{1, \dots, n\}$ обслуживания объектов в порядке их поступления. Если полученное решение недопустимо, восстановить допустимость решения. Положить $K = K(S_0)$, $\Phi_0 = \emptyset$, $j = 0$.

2. Выполнять, пока не сработал критерий остановки, последовательность действий 2.1–2.4:

- 2.1. Сформировать окрестность текущего решения $\mathcal{N}(S_j, \Phi_j)$;

- 2.2. Найти такое решение в окрестности $\mathcal{N}(S_j, \Phi_j)$, что $K(S_{j+1}) = \min\{K(S'_j) \mid S'_j \in \mathcal{N}(S_j, \Phi_j)\}$;

- 2.3. Если $K(S_{j+1}) < K$, то изменить значение $K = K(S_{j+1})$;

2.4. Обновить список запретов Φ_j и счетчик $j = j + 1$.

3. Предъявить наилучшее найденное решение.

В качестве критерия завершения работы алгоритма используется остановка по общему числу шагов, либо по числу шагов, в ходе которых не меняется значение K . Величина l является управляющим параметром алгоритма, ее значение подбирается экспериментально.

2.4.4. Результаты вычислительных экспериментов

С целью выяснения эффективности применения вышеописанных алгоритмов мягких вычислений была выполнена серия массовых вычислительных экспериментов в условиях, описанных в приложении А.

В таблице 2.7 приведены значения средних длительностей отработки алгоритмов динамического программирования DP, ветвей и границ BnB, имитации отжига SA, эволюционно-генетического GA и поиска с запретами TS для решения задачи Flow1R(Σ).

Таблица 2.7

Зависимости средних длительностей отработки алгоритмов решения задачи Flow1R(Σ) от размерности потока объектов \mathcal{O}_n , с

n	DP	BnB	SA	GA	TS
8	0.001	0.000	0.121	0.015	0.023
9	0.002	0.000	0.145	0.078	0.086
10	0.013	0.000	0.170	0.133	0.114
11	0.074	0.001	0.198	0.178	0.199
12	0.484	0.005	0.228	0.262	0.318
13	3.851	0.023	0.263	0.337	0.530
14	29.219	0.099	0.298	0.457	0.773
15	263.964	0.499	2.496	7.898	1.623
16	-	1.353	4.254	12.266	2.923
17	-	7.568	7.150	26.120	6.143
18	-	32.589	10.234	32.266	9.453
19	-	449.359	26.254	47.523	17.123
30	-	-	89.316	114.985	67.103
40	-	-	186.019	316.019	139.316

Качество каждого алгоритма оценивалось по таким показателям, как продолжительность синтеза стратегии обслуживания и среднее значение количества точных совпадений полученных оценок с оптимальными.

На рис. 2.2 представлены результаты сравнения субоптимальных решений с оптимальными, которые оценивались по количеству χ (в процентном выражении) точных совпадений с оптимальными.

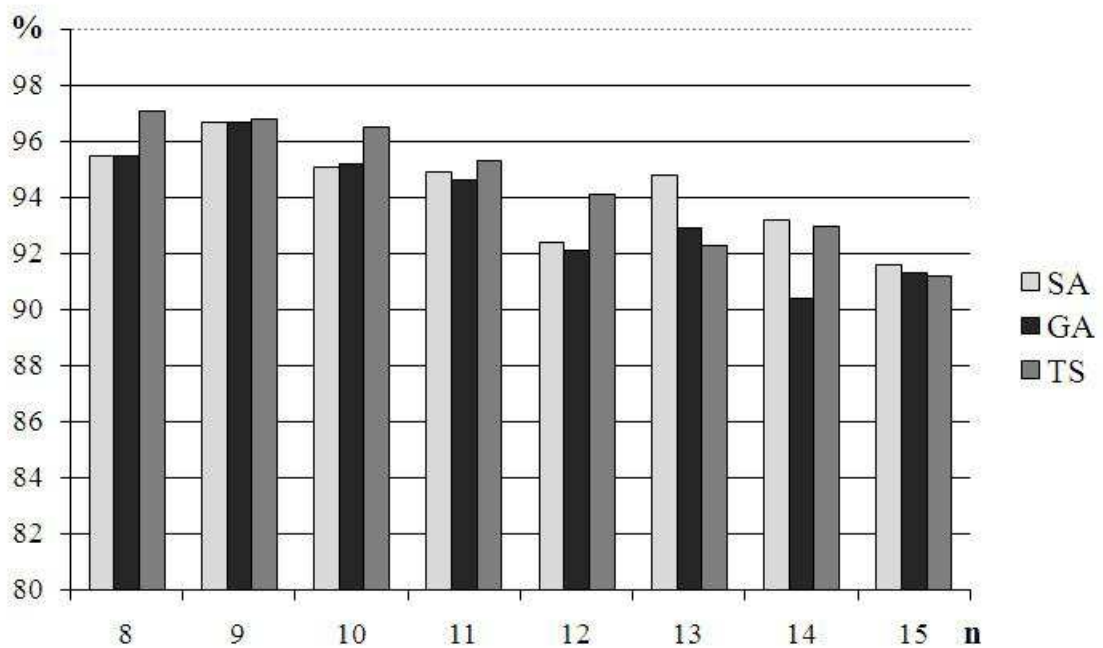


Рис. 2.2. Диаграмма статистики отклонения от точных решений приближенных, полученных алгоритмами GA, SA и TS.

Для задачи Flow1R(max) результаты аналогичны.

Выводы по главе 2

1) Построены однокритериальные модели обслуживания множества и потока объектов стационарным процессором с накопительно-расходным компонентом.

2) Для каждой модели п. 1 поставлены однокритериальные задачи синтеза оптимальных стратегий обслуживания.

3) Для поставленных в п. 2 задач синтеза выведены необходимые и достаточные условия существования решения; доказана *NP*-трудность задачи

синтеза оптимальной стратегии обслуживания по критерию минимизации максимального из них.

4) Сконструированы решающие алгоритмы для задач п. 2, основанные на идеологии динамического программирования и метода ветвей и границ, получены оценки их трудоёмкости, приведены результаты массовых вычислительных экспериментов.

5) Для синтеза субоптимальных стратегий обслуживания разработаны метаэвристические алгоритмы, основанные на эволюционно-генетическом подходе, имитации отжига и поиске с запретами.

Глава 3. Модели, бикритериальные задачи и алгоритмы синтеза стратегий обслуживания объектов в однопроцессорной системе с накопительно-расходным компонентом

Рассмотрено расширение введенных в главе 2 базовых математических моделей на случай двух критериев оценки качества стратегий управления обслуживанием. Описываются модификации модели обслуживания, учитывающие ограничения на допустимые величины опережений в очереди на обслуживание. Сформулированы бикритериальные задачи синтеза оптимальных стратегий обслуживания и сконструированы алгоритмы их точного, а также приближенного решения в рамках концепции Парето. Приведены примеры реализации алгоритмов и результаты массовых вычислительных экспериментов.

§ 3.1. Задачи обслуживания бинарного множества объектов процессором с накопительно-расходным компонентом

3.1.1. Математическая модель

Напомним описание математической модели обслуживания.

Задано конечное множество $\mathcal{O}_n = \{o_1, o_2, \dots, o_n\}$ независимых объектов, которые подлежат однократному однофазному обслуживанию стационарным процессором с накопительно-расходным компонентом. Для каждого объекта o_i , $i = \overline{1, n}$ определены следующие целочисленные параметры: τ_i — норма длительности обслуживания, d_i — мягкий директивный срок завершения обслуживания ($d_i \geq \tau_i$), v_i — вместимость объекта (объемная характеристика).

Множество \mathcal{O}_n обладает свойством бинарности и состоит из двух подмножеств: «входящего» \mathcal{O}^+ и «исходящего» \mathcal{O}^- таких, что $\mathcal{O}^+ \cup \mathcal{O}^- = \mathcal{O}_n$ и $\mathcal{O}^+ \cap \mathcal{O}^- = \emptyset$. Принадлежность объекта o_i , $i = \overline{1, n}$ тому или иному подмножеству определяется значением параметра w_i ($w_i = +1$, если $o_i \in \mathcal{O}^+$ и $w_i = -1$, если $o_i \in \mathcal{O}^-$).

Накопительный компонент процессора представляет собой резервуар

объемом V^* . Обозначим через V_t текущее заполнение резервуара в момент времени t (равное V_0 в начальный момент времени $t = 0$). В результате обслуживания объекта o_i из подмножества \mathcal{O}^+ (\mathcal{O}^-) значение характеристики V_t увеличивается (уменьшается) на величину v_i , $i = \overline{1, n}$. Процессор может начать обслуживание объекта $o_i \in \mathcal{O}_n$ при условии выполнения *объемных ограничений*

$$0 \leq V_t + w_i \cdot v_i \leq V^*. \quad (3.1)$$

Полагаем, что процессор готов к обслуживанию множества объектов \mathcal{O}_n в начальный момент времени $t = 0$. Обслуживание объекта o_i , $i = \overline{1, n}$ может быть начато свободным процессором в любой момент времени t и осуществляется без прерываний; необслуженный объект не может покинуть обслуживающую систему; одновременное обслуживание процессором двух и более объектов и его непроизводительные простои запрещены.

С каждым объектом o_i в рассматриваемой модели ассоциируются две монотонно возрастающие (в нестрогом смысле) функции индивидуального штрафа $\varphi_i(t)$ и $\psi_i(t)$. Совокупно функции $\varphi_i(t)$, $i = \overline{1, n}$ выражают величины потерь по первому показателю эффективности, а $\psi_i(t)$, $i = \overline{1, n}$ — по второму.

Стратегия обслуживания объектов S представляет собой произвольную перестановку $S = \{i_1, i_2, \dots, i_n\}$ совокупности индексов объектов $N = \{1, 2, \dots, n\}$. Стратегию S именуем допустимой, если удовлетворяются отмеченные выше объёмные ограничения на обслуживание каждого объекта. Множество всех допустимых стратегий обслуживания обозначим через Ω .

3.1.2. Постановки оптимизационных задач

С учетом введенных ранее обозначений $t^*(i_k, S)$ и $\bar{t}(i_k, S)$ и соотношений (2.11) формализация подхода парето-оптимальности для рассматриваемой в этом параграфе модели обслуживания приводит к следующим ак-

туальным для практического применения бикритериальным задачам.

Задача Set1R(\sum , max). Найти полную совокупность парето-оптимальных оценок в проблеме минимизации суммы индивидуальных штрафов $\varphi_i(t)$ по всем объектам входящего подмножества \mathcal{O}^+ и величины максимального из индивидуальных штрафов $\psi_i(t)$ по всем объектам исходящего подмножества \mathcal{O}^- :

$$\left\{ \min_{S \in \Omega} \left(\sum_{k: o_{i_k} \in \mathcal{O}^+} \varphi_{i_k}(\bar{t}(i_k, S)) \right), \min_{S \in \Omega} \left(\max_{l: o_{i_l} \in \mathcal{O}^-} \psi_{i_l}(\bar{t}(i_l, S)) \right) \right\}. \quad (3.2)$$

Задача Set1R(\sum , \sum). Найти полную совокупность парето-оптимальных оценок в проблеме минимизации суммы индивидуальных штрафов $\varphi_i(t)$ по всем объектам подмножества \mathcal{O}^+ и суммы индивидуальных штрафов $\psi_i(t)$ по всем объектам подмножества \mathcal{O}^- :

$$\left\{ \min_{S \in \Omega} \left(\sum_{k: o_{i_k} \in \mathcal{O}^+} \varphi_{i_k}(\bar{t}(i_k, S)) \right), \min_{S \in \Omega} \left(\sum_{l: o_{i_l} \in \mathcal{O}^-} \psi_{i_l}(\bar{t}(i_l, S)) \right) \right\}. \quad (3.3)$$

Задача Set1R(max, max). Найти полную совокупность парето-оптимальных оценок в проблеме минимизации величины максимального из индивидуальных штрафов $\varphi_i(t)$ по всем объектам подмножества \mathcal{O}^+ и величины максимального из индивидуальных штрафов $\psi_i(t)$ по всем объектам подмножества \mathcal{O}^- :

$$\left\{ \min_{S \in \Omega} \left(\max_{k: o_{i_k} \in \mathcal{O}^+} \varphi_{i_k}(\bar{t}(i_k, S)) \right), \min_{S \in \Omega} \left(\max_{l: o_{i_l} \in \mathcal{O}^-} \psi_{i_l}(\bar{t}(i_l, S)) \right) \right\}. \quad (3.4)$$

3.1.3. Решающие соотношения динамического программирования

Выполним решение задач Set1R(\sum , max), Set1R(\sum , \sum) и Set1R(max, max) алгоритмами, основанными на идеологии динамического программирования в его бикритериальном расширении.

Для этого обозначим через $\text{eff}(\mathbf{M})$ максимальное по включению подмножество недоминируемых по Парето двумерных векторов из \mathbf{M} , где \mathbf{M} — произвольное множество двумерных векторов-оценок. Введём двуместную операцию \otimes объединения двумерного вектора-оценки $\mathbf{x} = (x_1, x_2)$ и

множества \mathbf{Y} векторов $\mathbf{y} = (y_1, y_2)$ той же размерности. Определение этой операции для задач $\text{Set1R}(\sum, \max)$, $\text{Set1R}(\sum, \sum)$ и $\text{Set1R}(\max, \max)$ соответственно приведено в таблице 3.1.

Таблица 3.1

Операция \otimes объединения вектора и множества векторов

Задача	Определение операции
(\sum, \max)	$\mathbf{Y} \otimes \mathbf{x}$ — совокупность векторов $\mathbf{z} = (z_1, z_2)$, где $z_1 = y_1 + x_1$, а $z_2 = \max(y_2, x_2)$.
(\sum, \sum)	$\mathbf{Y} \otimes \mathbf{x}$ — совокупность векторов $\mathbf{z} = (z_1, z_2)$, где $z_1 = y_1 + x_1$ и $z_2 = x_2 + y_2$.
(\max, \max)	$\mathbf{Y} \otimes \mathbf{x}$ — совокупность векторов $\mathbf{z} = (z_1, z_2)$, где $z_1 = \max(y_1, x_1)$, а $z_2 = \max(y_2, x_2)$.

С учетом специфики операции \otimes объединения вектора и множества векторов решающие рекуррентные соотношения динамического программирования для задач $\text{Set1R}(\sum, \max)$, $\text{Set1R}(\sum, \sum)$ и $\text{Set1R}(\max, \max)$ формально можно записать в едином виде.

Алгоритм решения задач

$\text{Set1R}(\sum, \max)$, $\text{Set1R}(\max, \max)$, $\text{Set1R}(\max, \max)$

При обслуживании объектов множества \mathcal{O}_n решения принимаются в те моменты времени, когда процессор свободен и необходимо выбрать следующий для обслуживания объект из входящего или исходящего подмножества, учитывая текущее заполнение резервуара. Соответственно тройка значений (t, Q, V_t) определяет текущее состояние системы в момент t принятия решения, где Q — множество индексов объектов множества \mathcal{O}_n , еще не обслуженных в этот момент времени.

Пусть $E(t, Q, V_t)$ обозначает совокупность эффективных по Парето двумерных оценок, сгенерированную за период времени от момента t и до окончания процесса обслуживания всех объектов множества \mathcal{O}_n . Тогда полная совокупность эффективных по Парето оценок будет представлять собой множество $E(0, \mathcal{O}_n, V_0)$.

Обозначим через Q^* множество индексов объектов потока \mathcal{O}_n , допустимых к обслуживанию при условии выполнения объёмных ограничений (3.1) в состоянии (t, Q, V_t) , $Q^* \subseteq Q$.

Очевидно, что для любого $\theta \geq 0$ и $Q = \{\alpha\}$, где α — индекс допустимого к обслуживанию объекта потока \mathcal{O}_n , имеет место соотношение

$$E(t_n + \theta, \{\alpha\}, V_{t_n+\theta}) = \{\varphi_\alpha(t_n + \theta + \tau_\alpha), \psi_\alpha(t_n + \theta + \tau_\alpha)\}. \quad (3.5)$$

Если в состоянии (t, Q, V_t) в качестве очередного обслуживаемого объекта выбран объект с индексом $\alpha \in Q^*$, то следующим моментом принятия решения будет $t + \tau_\alpha$; при этом множество индексов объектов, ещё не обслуженных в этот момент времени, определяется как совокупность $Q \setminus \{\alpha\}$. Характеристика заполнения резервуара изменится и будет определяться выражением $V_t + w_\alpha v_\alpha$.

Суммируя приведенные рассуждения, запишем формулу

$$E(t, Q, V_t) = \text{eff} \left\{ \bigcup_{\alpha \in Q^*} [(\varphi(t + \tau_\alpha), \psi(t + \tau_\alpha)) \otimes E(t + \tau_\alpha, Q \setminus \{\alpha\}, V_t + w_\alpha \cdot v_\alpha)] \right\}, \quad (3.6)$$

образующую совместно с (3.5) замкнутую систему решающих задачи $\text{Set1R}(\sum, \max)$, $\text{Set1R}(\sum, \sum)$ и $\text{Set1R}(\max, \max)$ соотношений динамического программирования с учетом специфики определенной в таблице 3.1 операции объединения вектора и множества векторов \otimes .

Процесс решения задачи на основе построенных соотношений состоит из трех последовательных этапов.

На первом этапе осуществляется разметка для определения достижимых состояний системы и выполняется нумерация достижимых состояний системы в соответствии с оператором раскрытия состояний $\mathcal{R}(t, Q, V_t)$, введенном в п.2.1.3. Фиксируются финальные состояния, соответствующие завершению процесса обслуживания всех объектов потока \mathcal{O}_n . Таким состояниям соответствует одноэлементное множество оценок $E(t, Q, V_t) = \{(0, 0)\}$.

На втором этапе выполняется построение очередных множеств эффективных оценок для произвольных достижимых состояний. Эти состояния характеризуются тем, что множество эффективных оценок для них неизвестно, но для всех непосредственно следующих за ними состояний множество эффективных оценок уже известно. Для пояснения укажем, что состояние является непосредственно следующим за некоторым текущим состоянием, если в него можно перейти из текущего, выбрав на обслуживание любой объект с индексом из совокупности Q . Так, в начале второго этапа нам известны только одноэлементные множества оценок для финальных состояний системы обслуживания. Последним в процессе решения задачи определяется множество оценок для начального состояния системы $E(0, F(0), V_0)$ — полная совокупность эффективных по Парето оценок.

Для синтеза множества оптимальных по Парето стратегий на втором этапе дополнительно строится список переходов. В этот список для каждой включаемой в множество $E(t, Q, V_t)$ оценки (m_1, m_2) вносится запись $((m^1, m^2), \alpha, z.n, z^*.n^*)$, где:

$z.n$ — двуэлементный номер текущего состояния: z — получаемый в результате применения оператора \mathcal{R} номер состояния, а n — порядковый номер оценки (m^1, m^2) в множестве $E(t, Q, V_t)$;

$\alpha \in Q$ — индекс объекта, при выборе которого множество $E(t, Q, V_t)$ порождаемых оценок содержит оценку (m^1, m^2) ;

$z^*.n^*$ — двуэлементный номер непосредственно следующего состояния (полагаем равным « $\emptyset.\emptyset$ », если состояние (t, Q, V_t) финальное): z^* — номер состояния в результате применения оператора \mathcal{R} , а n^* — порядковый номер оценки, получаемой в результате обслуживания объекта с индексом α .

На третьем, последнем, этапе работы алгоритма с помощью построенного списка переходов последовательно синтезируется стратегия обслуживания, соответствующая выбранной ЛПР эффективной по Парето оцен-

ке (m_0^1, m_0^2) . Для этого в списке отыскивается запись с первым элементом, равным (m_0^1, m_0^2) , и третьим элементом, равным начальному состоянию системы $0.n_0$; обозначим его как $((m_0^1, m_0^2), \alpha_0, 0.n_0, z_0^*.n_0^*)$. Вторым элементом этой записи α_0 представляет собой индекс объекта, который при реализации искомой стратегии обслуживается первым. Следующим за $0.n_0$ в синтезируемой стратегии будет состояние с номером $z_0^*.n_0^*$. В списке переходов отыскивается запись с четвертым элементом $z_1.n_1$, равным $z_0^*.n_0^*$; обозначим её как $((m_1^1, m_1^2), \alpha_1, z_1.n_1, z_1^*.n_1^*)$. Вторым элементом записи α_1 представляет собой индекс объекта, который при реализации искомой стратегии выбирается вторым. Действуя описанным выше образом, последовательно получаем соответствующую оценке (m_0^1, m_0^2) полную парето-оптимальную стратегию. Процесс синтеза заканчивается отысканием записи соответствующей финальному состоянию системы с четвертой компонентой равной $\varnothing.\varnothing$.

Оценки трудоемкости алгоритмов

Предварительно отметим следующее. Пусть M_1 и M_2 — два заданных множества двумерных векторов-оценок, упорядоченных по убыванию второй координаты. В каждом из этих множеств нет оценок, доминирующих одна другую. Пусть множество M_1 состоит из n_1 элементов, множество M_2 — из n_2 элементов. Тогда множество $\text{eff}(M_1 \cup M_2)$, также упорядоченное по убыванию второй координаты, может быть построено путем выполнения $O(n_1 + n_2)$ элементарных операций. Соответствующий алгоритм строится без затруднений.

Дадим верхнюю оценку числа элементарных операций, выполняемых при вычислениях по соотношениям (3.5) и (3.6) в процессе решения задачи $\text{Set1R}(\sum, \max)$.

Процедура решения задачи $\text{Set1R}(\sum, \max)$ предусматривает последовательное определение множеств $E(t, Q, V_t)$. В рассматриваемых состояниях (t, Q, V_t) число значений первого аргумента t сверху ограничено ве-

личиной $H = \sum_{i=1}^n \tau_i$, а Q является произвольным подмножеством индексов объектов. Таким образом, число состояний системы не превышает $H \cdot 2^n$. Значение V_t в состоянии (t, Q, V_t) используется для выбора из множества Q подмножества допустимых в данном состоянии объектов Q^* . Эта операция мажорируется линейной функцией от n . Максимальное из возможных значений вторых координат векторов-оценок в множествах $E(t, Q, V_t)$ не может превышать $\max_{i: o_i \in \mathcal{O}^-} \psi_i(H)$. Поэтому число оценок в каждом из множеств $E(t, Q, V_t)$ при всех возможных значениях аргументов не может превышать $\max_{i: o_i \in \mathcal{O}^-} \psi_i(H) + 1$. Будем считать, что в каждом из найденных в процессе решения задачи множеств $E(t, Q, V_t)$ векторы упорядочены по убыванию значений второй координаты. Тогда общее число элементарных операций, выполняемых при определении каждого очередного множества $E(t, Q, V_t)$, оценивается как $O(\max_{i: o_i \in \mathcal{O}^-} \psi_i(H))$. Следовательно, число элементарных операций, выполняемых при отыскании полной совокупности эффективных оценок, характеризуется величиной $O(H 2^n n \max_{i: o_i \in \mathcal{O}^-} \psi_i(H))$.

Очевидно, что для решения задачи $\text{Set1R}(\max, \max)$ верхняя оценка числа реализуемых алгоритмом элементарных операций такая же, как при решении задачи $\text{Set1R}(\sum, \max)$.

Дадим верхнюю оценку числа элементарных операций, выполняемых при вычислениях по соотношениям (3.5) и (3.6) в процессе решения задачи $\text{Set1R}(\sum, \sum)$. Процедура решения задачи $\text{Set1R}(\sum, \sum)$ также предусматривает последовательное определение множеств $E(t, Q, V_t)$ в состояниях (t, Q, V_t) . Число состояний системы не превышает значения $H \cdot 2^n$, при этом необходимо не более n операций для выбора подмножества Q^* допустимых в данном состоянии объектов. Максимальное из возможных значений вторых координат векторов-оценок в множествах $E(t, Q, V_t)$ не может превышать $n \cdot \max_{i: o_i \in \mathcal{O}^-} \psi_i(H)$. Поэтому число оценок в каждом из множеств $E(t, Q, V_t)$ при всех возможных значениях аргументов не больше

$n \cdot \max_{i: o_i \in \mathcal{O}^-} \psi_i(H) + 1$. Можно считать, что в каждом из найденных в процессе решения задачи множеств $E(t, Q, V_t)$ векторы упорядочены по убыванию значений второй координаты. Тогда общее число элементарных операций, выполняемых при определении каждого очередного множества эффективных оценок, ограничено величиной порядка $O(n \max_{i: o_i \in \mathcal{O}^-} \psi_i(H))$, а число элементарных операций, выполняемых при отыскании полной совокупности эффективных оценок, характеризуется величиной $O(H2^n n^2 \max_{i: o_i \in \mathcal{O}^-} \psi_i(H))$.

Как видно из описанной выше процедуры, алгоритм определения по принадлежащей множеству $E(0, F(0), V_0)$ векторной оценке соответствующей парето-оптимальной стратегии имеет линейно зависящую от n оценку числа выполняемых элементарных операций.

Полученные оценки трудоемкости процедур синтеза полных совокупностей эффективных оценок для задач $\text{Set1R}(\sum, \max)$, $\text{Set1R}(\sum, \sum)$ и $\text{Set1R}(\max, \max)$ экспоненциальны.

3.1.4. Результаты вычислительных экспериментов

С целью выяснения эффективности применения алгоритма была выполнена серия массовых вычислительных экспериментов в условиях, описанных в приложении А.

Результаты экспериментов представлены в таблице 3.2.

Таблица 3.2

Зависимости длительностей отработки алгоритмов решения задач от размерности потока объектов \mathcal{O}_n , с

n	$\text{Set1R}(\sum, \max)$		$\text{Set1R}(\sum, \sum)$		$\text{Set1R}(\max, \max)$	
	t_{avg}	t_{max}	t_{avg}	t_{max}	t_{avg}	t_{max}
8	0.004	0.016	0.003	0.013	0.004	0.016
9	0.033	0.094	0.029	0.085	0.035	0.031
10	0.245	1.094	0.213	1.012	0.256	0.141
11	2.297	9.016	2.125	10.012	3.031	9.200
12	21.059	81.265	23.569	98.652	20.265	86.325
13	236.065	1569.255	256.365	1485.236	308.899	1625.365

Как очевидно, для решения задач $\text{Set1R}(\sum, \max)$, $\text{Set1R}(\sum, \sum)$ и $\text{Set1R}(\max, \max)$ необходимое количество процессорного времени одного порядка.

Разработанные алгоритмы позволяют выполнить синтез парето-оптимальной совокупности стратегий обслуживания потока объектов до 13 за промежутков времени в среднем от нескольких секунд до нескольких минут.

§ 3.2. Задачи обслуживания бинарного потока объектов процессором с накопительно-расходным компонентом

3.2.1. Математическая модель

Для каждого объекта o_i , $i = \overline{1, n}$ бинарного детерминированного потока \mathcal{O}_n определен целочисленный параметр t_i — момент поступления в очередь на обслуживание; объекты пронумерованы в порядке поступления в систему обслуживания, т.е. $0 \leq t_1 \leq \dots \leq t_n$. Остальные характеристики и параметры потока, процессора и накопительно-расходного компонента принимаются такими же, как и в модели п. 3.1.1.

3.2.2. Постановки оптимизационных задач

Задача $\text{Flow1R}(\sum, \max)$. *Найти полную совокупность парето-оптимальных оценок в проблеме минимизации суммы индивидуальных штрафов $\varphi_i(t)$ по всем объектам входящего подпотока \mathcal{O}^+ и величины максимального из индивидуальных штрафов $\psi_i(t)$ по всем объектам исходящего подпотока \mathcal{O}^- :*

$$\left\{ \min_{S \in \Omega} \left(\sum_{k: o_{i_k} \in \mathcal{O}^+} \varphi_{i_k}(\bar{t}(i_k, S)) \right), \min_{S \in \Omega} \left(\max_{l: o_{i_l} \in \mathcal{O}^-} \psi_{i_l}(\bar{t}(i_l, S)) \right) \right\}. \quad (3.7)$$

Задача $\text{Flow1R}(\sum, \sum)$. *Найти полную совокупность парето-оптимальных оценок в проблеме минимизации суммы индивидуальных штрафов $\varphi_i(t)$ по всем объектам подпотока \mathcal{O}^+ и суммы индивидуальных*

штрафов $\psi_i(t)$ по всем объектам подпотока \mathcal{O}^- :

$$\{\min_{S \in \Omega} (\sum_{k: o_{i_k} \in \mathcal{O}^+} \varphi_{i_k}(\bar{t}(i_k, S))), \min_{S \in \Omega} (\sum_{l: o_{i_l} \in \mathcal{O}^-} \psi_{i_l}(\bar{t}(i_l, S)))\}. \quad (3.8)$$

Задача Flow1R(max, max). Найти полную совокупность парето-оптимальных оценок в проблеме минимизации величины максимального из индивидуальных штрафов $\varphi_i(t)$ по всем объектам подпотока \mathcal{O}^+ и величины максимального из индивидуальных штрафов $\psi_i(t)$ по всем объектам подпотока \mathcal{O}^- :

$$\{\min_{S \in \Omega} (\max_{k: o_{i_k} \in \mathcal{O}^+} \varphi_{i_k}(\bar{t}(i_k, S))), \min_{S \in \Omega} (\max_{l: o_{i_l} \in \mathcal{O}^-} \psi_{i_l}(\bar{t}(i_l, S)))\}. \quad (3.9)$$

3.2.3. Решающие соотношения динамического программирования

Выполним решение задач Flow1R(\sum , max), Flow1R(\sum , \sum) и Flow1R(max, max) алгоритмами, основанными на идеологии динамического программирования в его бикритериальном расширении.

Для этого воспользуемся введенными в п. 3.1.3 операциями: $\text{eff}(\mathbf{M})$ — выделения максимального по включению подмножества недоминируемых по Парето двумерных векторов и \otimes — объединения двумерного вектора и множества векторов. Специфика определения операции \otimes для задач Flow1R(\sum , max), Flow1R(\sum , \sum) и Flow1R(max, max), приведенная в таблице 3.1, позволяет записать рекуррентные соотношения динамического программирования для их решения формально в едином виде.

Алгоритм решения задач

$$\text{Flow1R}(\sum, \max), \text{Flow1R}(\sum, \sum), \text{Flow1R}(\max, \max)$$

Обозначим через $F(t)$ определяемое исходными данными подмножество индексов объектов потока \mathcal{O}_n , которые поступают в систему обслуживания в момент времени t . Совокупность индексов объектов потока \mathcal{O}_n , поступающих в систему обслуживания на временном отрезке $[t + 1, t + \delta]$, $\delta \geq 1$, обозначим через $D(t, \delta)$. Очевидно, что $D(t, \delta) = \bigcup_{j=1}^{\delta} F(t + j)$.

При обслуживании объектов потока \mathcal{O}_n решения принимаются в те моменты времени, когда процессор свободен и необходимо выбрать следующий для обслуживания объект из входящего или исходящего подпотока, учитывая текущее заполнение резервуара. Соответственно, тройка значений (t, Q, V_t) определяет текущее состояние системы в момент t принятия решения, где Q — множество индексов объектов потока \mathcal{O}_n , ожидающих обслуживания в этот момент времени.

Для любого состояния системы обслуживания (t, Q, V_t) непустота множества Q обеспечивается включением фиктивного объекта с параметрами $t_0 = t$, $\tau_0 = \min\{\delta | D(t, \delta) \neq \emptyset\}$, $d_0 = \infty$, $v_0 = 0$; функции индивидуального штрафа $\varphi_0(t)$ и $\psi_0(t)$ при этом тождественно равны нулю. Для определенности будем считать, что фиктивный объект включен в подпоток \mathcal{O}^+ и, следовательно, $w_0 = +1$. Выбор фиктивного объекта на обслуживание означает простой процессора, начиная от момента t до момента времени поступления в очередь какого-либо нового объекта потока \mathcal{O}_n . Фиктивный объект исключается из множества Q при $t \geq t_n$, т.е. в случае, когда все объекты потока \mathcal{O}_n поступили в систему и ожидают своего обслуживания.

Пусть $E(t, Q, V_t)$ обозначает совокупность эффективных по Парето двумерных оценок, сгенерированную за период времени от момента t и до окончания процесса обслуживания всех объектов потока \mathcal{O}_n . Тогда полная совокупность эффективных по Парето оценок будет представлять собой множество $E(0, F(0), V_0)$.

В текущем состоянии (t, Q, V_t) системы объект может быть допущен к обслуживанию только при условии выполнения объёмных ограничений (3.1). Обозначим через Q^* множество индексов объектов потока \mathcal{O}_n , допустимых к обслуживанию в состоянии (t, Q, V_t) , $Q^* \subseteq Q$.

Очевидно, что для любого $\theta \geq 0$ и $Q = \{\alpha\}$, где α — индекс допусти-

мого к обслуживанию объекта потока \mathcal{O}_n , имеет место соотношение

$$E(t_n + \theta, \{\alpha\}, V_{t_n + \theta}) = \{\varphi_\alpha(t_n + \theta + \tau_\alpha), \psi_\alpha(t_n + \theta + \tau_\alpha)\}. \quad (3.10)$$

Если в состоянии (t, Q, V_t) в качестве очередного обслуживаемого объекта выбран объект с индексом $\alpha \in Q^*$, то следующим моментом принятия решения будет $t + \tau_\alpha$; при этом множество индексов объектов, ожидающих обслуживания в этот момент времени, определяется как совокупность $(Q \setminus \{\alpha\}) \cup D(t, \tau_\alpha)$. Характеристика заполнения резервуара изменится и будет равна $V_t + w_\alpha \cdot v_\alpha$. Суммируя приведенные рассуждения, запишем рекуррентное выражение

$$E(t, Q, V_t) = \text{eff}\left\{ \bigcup_{\alpha \in Q^*} [(\varphi(t + \tau_\alpha), \psi(t + \tau_\alpha)) \otimes E(t + \tau_\alpha, (Q \setminus \{\alpha\}) \cup D(t, \tau_\alpha), V_t + w_\alpha \cdot v_\alpha)] \right\}, \quad (3.11)$$

образующее совместно с (3.10) замкнутую систему решающих задачи $\text{Flow1R}(\sum, \max)$, $\text{Flow1R}(\sum, \sum)$ и $\text{Flow1R}(\max, \max)$ соотношений динамического программирования.

Процесс решения задачи на основе построенных соотношений аналогичен описанному в п. 3.1.3.

Оценки трудоемкости алгоритмов

Дадим верхнюю оценку числа элементарных операций, выполняемых алгоритмом при вычислениях по соотношениям (3.10) и (3.11) в процессе решения задачи $\text{Flow1R}(\sum, \max)$. Процедура решения задачи $\text{Flow1R}(\sum, \max)$ предусматривает последовательное определение множеств $E(t, Q, V_t)$. В состояниях (t, Q, V_t) число значений первого аргумента t сверху ограничено величиной $L = t_n + \sum_{i=1}^n \tau_i$, а Q является произвольным подмножеством индексов из числа прибывших на момент времени t объектов с учетом наличия фиктивного объекта. Таким образом, число состояний системы не превышает $L \cdot 2^{n+1}$. Значение V_t в состоянии (t, Q, V_t) используется для выбора из множества Q подмножества допустимых к обслужива-

нию в данном состоянии объектов Q^* ; эта операция выбора мажорируется линейной функцией от n . Максимальное из возможных значений вторых координат векторов-оценок в множествах $E(t, Q, V_t)$ не может превышать $\max_{i:o_i \in \mathcal{O}^-} \psi_i(L)$. Поэтому число оценок в каждом из множеств $E(t, Q, V_t)$ при всех возможных значениях аргументов не может превышать $\max_{i:o_i \in \mathcal{O}^-} \psi_i(L) + 1$. Будем считать, что в каждом из найденных в процессе решения задачи множеств $E(t, Q, V_t)$ векторы упорядочены по убыванию значений второй координаты. Тогда общее число элементарных операций, выполняемых при определении каждого очередного множества $E(t, Q, V_t)$, оценивается как величина $O(\max_{i:o_i \in \mathcal{O}^-} \psi_i(L))$. Число элементарных операций, выполняемых при отыскании полной совокупности эффективных оценок, характеризуется величиной $O(L2^{n+1}n \max_{i:o_i \in \mathcal{O}^-} \psi_i(L))$.

Для решения задачи $\text{FlowR}(\max, \max)$ верхняя оценка числа реализуемых алгоритмом элементарных операций такая же, как при решении задачи $\text{FlowR}(\sum, \max)$.

Дадим верхнюю оценку числа элементарных операций, выполняемых при вычислениях по соотношениям (3.10) и (3.11) в процессе решения задачи $\text{Flow1R}(\sum, \sum)$. Аналогично число состояний системы не превышает значения $L \cdot 2^{n+1}$. Величина V_t в состоянии (t, Q, V_t) используется для выбора из множества Q подмножества индексов допустимых в данном состоянии объектов Q^* ; эта операция мажорируется линейной функцией от n . Максимальное из возможных значений вторых координат векторов-оценок в множествах $E(t, Q, V_t)$ не может превышать $n \cdot \max_{i:o_i \in \mathcal{O}^-} \psi_i(L)$. Поэтому число оценок в каждом из множеств $E(t, Q, V_t)$ при всех возможных значениях аргументов не больше $n \cdot \max_{i:o_i \in \mathcal{O}^-} \psi_i(L) + 1$. Можно считать, что в каждом из найденных в процессе решения задачи множеств $E(t, Q, V_t)$ векторы упорядочены по убыванию значений второй координаты. Тогда общее число элементарных операций, выполняемых при определении каждого очередного множества эффективных оценок, оценивается величиной

$O(n \max_{i: o_i \in \mathcal{O}^-} \psi_i(L))$. Число элементарных операций, выполняемых при отыскании полной совокупности эффективных оценок, характеризуется величиной $O(L2^n n^2 \max_{i: o_i \in \mathcal{O}^-} \psi_i(L))$.

Как видно из описанной выше процедуры, алгоритм определения по принадлежащей множеству $E(0, F(0), V_0)$ векторной оценке соответствующей парето-оптимальной стратегии имеет линейно зависящую от n оценку числа выполняемых элементарных операций. Полученные оценки трудоемкости процедур синтеза полных совокупностей эффективных оценок для задач $\text{Flow1R}(\sum, \max)$, $\text{Flow1R}(\sum, \sum)$ и $\text{Flow1R}(\max, \max)$ экспоненциальны.

З а м е ч а н и е 3.1. *Верхняя оценка числа элементарных операций, выполняемых для решения задачи $\text{Flow1R}(\sum, \max)$, больше, чем для решения задачи $\text{Set1R}(\sum, \max)$, так как $L > H$ ($L = t_n + \sum_{i=1}^n \tau_i$, $H = \sum_{i=1}^n \tau_i$) и при решении задачи $\text{Set1R}(\sum, \max)$ нет необходимости учитывать фиктивный объект при синтезе стратегий обслуживания. Аналогичные замечания имеют место для задач $\text{Flow1R}(\sum, \sum)$ и $\text{Flow1R}(\max, \max)$.*

3.2.4. Пример синтеза стратегии обслуживания

Для иллюстрации технологии расчета по описанному алгоритму динамического программирования рассмотрим решение задачи $\text{Flow1R}(\sum, \max)$ на следующем примере.

Пример 3.1. Требуется найти полную совокупность эффективных по Парето оценок и соответствующие этим оценкам парето-оптимальные стратегии обслуживания. Максимальная вместимость резервуара $V^* = 19$, в начальный момент времени $t = 0$ заполнение $V_0 = 10$. Функции индивидуального штрафа определяются выражениями $\varphi_i(t) = a_i \cdot (t - \tau_i - t_i)$ и $\psi_i(t) = \max(t - d_i, 0)$, $i = \overline{1, 4}$, где a_i — целое число, штраф за единицу времени простоя в ожидании обслуживания объекта, $i = \overline{1, 4}$. Первая функ-

ция представляет собой суммарный штраф за простои объектов потока \mathcal{O}_n в ожидании обслуживания, а вторая — оценивает максимальное по продолжительности нарушение директивного срока завершения обслуживания среди всех объектов потока \mathcal{O}_n . Значения параметров модели представлены в таблице 3.3.

Таблица 3.3

Исходные данные

i	t_i	τ_i	a_i	d_i	v_i	w_i
1	0	3	2	5	5	+1
2	2	4	3	7	6	+1
3	4	2	5	9	2	-1
4	5	4	6	9	7	-1

По ходу решения будем составлять список переходов, в который вносятся записи описанного в п. 3.1.3 типа (таблица 3.4).

Полная совокупность эффективных оценок в этой задаче будет представлять собой $E(0, \{0, 1\}, 10)$. В множество \mathcal{Q} также включен индекс фиктивного объекта 0. Присвоим состоянию $(0, \{0, 1\}, 10)$ номер 0. По формуле (3.11) получаем $E(0, \{0, 1\}, 10) = \text{eff}\{(0, 0) \otimes E(2, \{0, 1, 2\}, 10), (0, 0) \otimes E(3, \{0, 2\}, 15)\}$. Присвоим состояниям $(2, \{0, 1, 2\}, 10)$ и $(3, \{0, 2\}, 15)$ номера 1 и 2 соответственно. По ходу решения каждое новое состояние системы будет получать следующий порядковый номер. В таблице 3.5 приведены все рассмотренные во время решения примера состояния.

Далее по формулам (3.10) и (3.11) найдем рекурсивно множества двумерных оценок $E(2, \{0, 1, 2\}, 10)$ и $E(3, \{0, 2\}, 15)$.

Для вычисления $E(2, \{0, 1, 2\}, 10)$ нет смысла в выборе фиктивного объекта, потому что обслуживание объекта с индексом 1 может завершиться до поступления объекта с индексом 3. Нецелесообразно также обслуживать объект с индексом 1, так как его обслуживание можно начать раньше момента времени $t = 2$. Следовательно, $E(2, \{0, 1, 2\}, 10) = \text{eff}[3(2 - 2), \max(6 - 7, 0)) \otimes E(6, \{1, 3, 4\}, 16)] = E(6, \{1, 3, 4\}, 16)$. Так

как в состоянии $(6, \{1, 3, 4\}, 16)$ все объекты уже поступили в систему обслуживания, то с этого шага исключаем фиктивный объект из рассмотрения. Тогда по формуле (3.11) $E(6, \{1, 3, 4\}, 16) = \text{eff}[(2(6 - 0), \max(9 - 5, 0)) \otimes E(9, \{3, 4\}, 21); (5(6 - 4), \max(8 - 9, 0)) \otimes E(8, \{1, 4\}, 14); (6(6 - 5), \max(10 - 9, 0)) \otimes E(10, \{1, 3\}, 9)] = \text{eff}[(12, 4) \otimes E(9, \{3, 4\}, 21); (10, 0) \otimes E(8, \{1, 4\}, 14); (6, 1) \otimes E(10, \{1, 3\}, 9)]$. Если в состоянии $(6, \{1, 3, 4\}, 16)$ обслужить объект с индексом 1, то заполнение резервуара превысит максимально возможное $V^* = 19$. Таким образом, для нахождения совокупности $E(6, \{1, 3, 4\}, 16)$ необходимо найти $E(8, \{1, 4\}, 14)$ и $E(10, \{1, 3\}, 9)$. $E(8, \{1, 4\}, 14) = \text{eff}[(16, 6) \otimes E(11, \{4\}, 19); (18, 3) \otimes E(12, \{1\}, 7)]$. Используя формулу (3.10), получаем $E(11, \{4\}, 19) = (36, 6)$, $E(12, \{1\}, 7) = (24, 10)$. Внесем следующие записи в список переходов: $((36, 6), 4, 6.1, \varnothing. \varnothing)$ и $((24, 10), 1, 7.1, \varnothing. \varnothing)$.

Таблица 3.4

Список переходов

№	$((m^1, m^2), \alpha, z.e, z^*.e^*)$	№	$((m^1, m^2), \alpha, z.e, z^*.e^*)$
1	$((36, 6), 4, 6.1, \varnothing. \varnothing)$	15	$((66, 6), 2, 12.1, 8.1)$
2	$((24, 10), 1, 7.1, \varnothing. \varnothing)$	16	$((52, 8), 3, 12.2, 13.1)$
3	$((52, 6), 1, 4.1, 6.1)$	17	$((66, 6), 4, 10.1, 12.1)$
4	$((42, 10), 4, 4.2, 7.1)$	18	$((52, 8), 4, 10.2, 12.2)$
5	$((45, 6), 3, 8.1, \varnothing. \varnothing)$	19	$((30, 5), 4, 14.1, \varnothing. \varnothing)$
6	$((24, 10), 1, 7.1, \varnothing. \varnothing)$	20	$((24, 7), 2, 15.1, \varnothing. \varnothing)$
7	$((65, 8), 1, 5.1, 8.1)$	21	$((42, 5), 2, 11.1, 14.1)$
8	$((54, 10), 3, 5.2, 7.1)$	22	$((30, 7), 4, 11.2, 15.1)$
9	$((62, 6), 3, 3.1, 4.1)$	23	$((42, 5), 3, 9.1, 11.1)$
10	$((52, 10), 3, 3.2, 4.2)$	24	$((30, 7), 3, 9.2, 11.2)$
11	$((62, 6), 2, 1.1, 3.1)$	25	$((42, 5), 0, 2.1, 9.1)$
12	$((52, 10), 2, 1.2, 3.2)$	26	$((30, 7), 0, 2.2, 9.2)$
13	$((45, 6), 3, 8.1, \varnothing. \varnothing)$	27	$((42, 5), 1, 0.1, 2.1)$
14	$((27, 8), 2, 13.1, \varnothing. \varnothing)$	28	$((30, 7), 1, 0.2, 2.2)$

В результате $E(8, \{1, 4\}, 14) = \text{eff}[(16, 6) \otimes (36, 6), (18, 3) \otimes (24, 10)] = \{(52, 6), (42, 10)\}$. В список переходов внесем записи $((52, 6), 1, 4.1, 6.1)$ и $((42, 10), 4, 4.2, 7.1)$.

Аналогично получаем $E(10, \{1, 3\}, 9) = \{(65, 8), (54, 10)\}$.

Теперь найдем $E(6, \{1, 3, 4\}, 16) = \{(62, 6), (52, 10)\}$. Так как $E(2, \{0, 1, 2\}, 10) = E(6, \{1, 3, 4\})$, то $E(2, \{0, 1, 2\}, 10) = \{(62, 6), (52, 10)\}$. Поэтому в список переходов вносим записи $((62, 6), 2, 1.1, 3.1)$ и $((52, 10), 2, 1.2, 3.2)$.

Рассуждая аналогичным образом, найдем $E(3, \{0, 2\}, 15) = \{(42, 5), (30, 7)\}$, постепенно заполняя таблицу 3.4. Наконец найдем полную совокупность эффективных оценок $E(0, \{0, 1\}, 10) = \text{eff}[E(2, \{0, 1, 2\}, 10), E(3, \{0, 2\}, 15)] = \text{eff}[(62, 6), (52, 10), (42, 5), (30, 7)] = \{(42, 5), (30, 7)\}$.

Таблица 3.5

Список состояний

№	(t, Q, V_t)	№	(t, Q, V_t)
0	$(0, \{0, 1\}, 10)$	8	$(13, \{3\}, 14)$
1	$(2, \{0, 1, 2\}, 10)$	9	$(4, \{0, 2, 3\}, 15)$
2	$(3, \{0, 2\}, 15)$	10	$(5, \{2, 3, 4\}, 15)$
3	$(6, \{1, 3, 4\}, 16)$	11	$(6, \{2, 4\}, 13)$
4	$(8, \{1, 4\}, 14)$	12	$(9, \{2, 3\}, 8)$
5	$(10, \{1, 3\}, 9)$	13	$(11, \{2\}, 6)$
6	$(11, \{4\}, 19)$	14	$(10, \{4\}, 19)$
7	$(12, \{1\}, 7)$	15	$(10, \{2\}, 6)$

В итоге полная совокупность эффективных оценок будет $\{(42, 5), (30, 7)\}$. Пусть из данной совокупности ЛПР выбрана оценка $(42, 5)$. Построим стратегию обслуживания, соответствующую этой оценке. Заметим, что оценка $(42, 5)$ находится в записи $((42, 5), 1, 0.1, 2.1)$. Второй элемент записи равен 1, поэтому, объект с индексом 1 будет обслуживаться первым в стратегии. Следующим состоянием будет состояние с номером 2.1. Находим в списке переходов соответствующую запись $((42, 5), 0, 2.1, 9.1)$, второй элемент которой равен 0. Следовательно, процессор будет простаивать до поступления в систему очередного объекта. Текущая оценка $(42, 5)$ в состоянии с номером 2.1 получена из

состояния с номером 9.1. Соответствующая запись в списке переходов будет $((42, 5), 3, 9.1, 11.1)$, и вторым будет обслуживаться объект с индексом 3. Оценка $(42, 5)$ в состоянии с номером 9.1 получена из состояния с номером 11.1. Находим в списке переходов запись $((42, 5), 2, 11.1, 14.1)$. Очередным будет обслуживаться объект с индексом 2. Следующая запись соответствует оценке в состоянии с номером 14.1. В списке переходов это финальное состояние $((30, 5), 4, 14.1, \varnothing.\varnothing)$, и последним обслуживается объект с индексом 4. В итоге получили стратегию обслуживания объектов вида $\{1, 3, 2, 4\}$. Аналогично можно найти парето-оптимальную стратегию обслуживания $\{1, 3, 4, 2\}$, характеризующуюся оценкой $(30, 7)$.

3.2.5. Результаты вычислительных экспериментов

Результаты экспериментов представлены в таблице 3.6, в которой показаны зависимости длительностей отработки алгоритма динамического программирования для решения задач $\text{Flow1R}(\sum, \max)$, $\text{Flow1R}(\sum, \sum)$ и $\text{Flow1R}(\max, \max)$ от размерности потока \mathcal{O}_n .

Таблица 3.6

Зависимости длительностей отработки алгоритмов решения задач от размерности потока объектов \mathcal{O}_n , с

n	$\text{Flow1R}(\sum, \max)$		$\text{Flow1R}(\sum, \sum)$		$\text{Flow1R}(\max, \max)$	
	t_{avg}	t_{max}	t_{avg}	t_{max}	t_{avg}	t_{max}
8	0.001	0.016	0.001	0.016	0.001	0.016
9	0.003	0.031	0.003	0.031	0.004	0.032
10	0.017	0.250	0.020	0.297	0.016	0.141
11	0.099	1.719	0.112	1.074	0.111	1.422
12	0.672	10.421	0.871	24.063	0.784	10.016
13	4.118	115.611	5.006	73.236	5.404	107.365
14	36.722	585.088	45.350	633.016	45.350	597.358
15	372.074	6058.464	447.542	6344.020	368.262	6680.045

Из результатов вычислительных экспериментов видно, что для решения задач $\text{Flow1R}(\sum, \max)$, $\text{Flow1R}(\sum, \sum)$ и $\text{Flow1R}(\max, \max)$ необходимо количество процессорного времени одного порядка.

Разработанные алгоритмы позволяют выполнить синтез парето-оптимальной совокупности стратегий обслуживания потока объектов до 15 за промежутков времени в среднем от нескольких секунд до нескольких минут. Такие временные интервалы приемлемы для синтеза план-графиков при обслуживании объектов в транспортно-технологических системах.

§ 3.3. Решение задач обслуживания методом ветвей и границ

3.3.1. Описание алгоритма

Общий подход к решению многокритериальной задачи методом ветвей и границ описан в работах [28, 101, 124].

Не нарушая общности, рассмотрим конкретизацию схемы алгоритма ветвей и границ на примере решения задачи $\text{Flow1R}(\sum, \max)$. Для задач $\text{Flow1R}(\sum, \sum)$ и $\text{Flow1R}(\max, \max)$ рассуждения легко повторить с учетом специфики расчета значений оптимизируемых критериев.

Обозначим через \mathcal{Z}^2 бикритериальную задачу вида (3.7). В дереве вариантов решения этой задачи \mathcal{Z}^2 каждое ребро фиксирует индекс очередного обслуживаемого объекта. Для каждой вершины N дерева вариантов однозначно определен путь, ведущий из корня в эту вершину. Число составляющих этот путь ребер будем именовать *рангом* вершины и обозначать как $r(N)$. При этом корень дерева решений — это вершина с рангом 0. Вершине N соответствует частная задача $\mathcal{Z}^2(N)$, отличие которой от исходной задачи \mathcal{Z}^2 заключается в том, что очередность обслуживания первых $r(N)$ объектов уже определена. Процесс ветвления состоит в выборе на первом уровне дерева ветвления первого объекта на обслуживание, на следующем уровне — объекта обслуживаемого вторым и т.д. [120].

Пусть $P(N) = \{p_1, p_2, \dots, p_{r(N)}\}$ — стратегия обслуживания объектов, которая фиксируется путем из корня дерева в вершину N . Тогда величина заполнения емкости резервуара $V(N)$ в вершине N вычисляется по формуле

$$V(N) = V_0 + \sum_{j=1}^{r(N)} w_{p_j} \cdot v_{p_j}.$$

При ветвлении дополнительно необходимо учитывать объёмные ограничения на обслуживание объектов. Следовательно, ребро из вершины N для объекта с индексом α , $o_\alpha \in \mathcal{O}^+$ строится только в том случае, если $V(N) + v_\alpha \leq V^*$, а для объекта с индексом β , $o_\beta \in \mathcal{O}^-$ ребро строится только если $V(N) \geq v_\beta$. Поскольку для вершины N первые $r(N)$ индексы объектов уже зафиксированы траекторией из корня в эту вершину, то при выполнении ветвления из N строится не более, чем $(n - r(N))$ ребер. В результате получается не более, чем $(n - r(N))$ новых вершин, ранг которых $r(N) + 1$.

Вершину с рангом $r(N) = n - 1$ назовём *конечной* вершиной дерева, так как в этом случае остается единственный объект, который необходимо включить в стратегию обслуживания, и нет необходимости в дальнейшем ветвлении. Путь от корня к конечной вершине дерева полностью определяет допустимую стратегию обслуживания потока объектов.

При анализе каждой вершины N дерева рассчитываются два оценочных вектора: *верхний* (примем для него обозначение $H(N)$) и *нижний* (примем для него обозначение $L(N)$). Верхний оценочный вектор порождается некоторым допустимым решением задачи. Нижний оценочный вектор $L(N) = \{l_1, l_2\}$ характеризует множество $K(N)$ эффективных по Парето оценок в задаче $\mathcal{Z}^2(N)$ следующим образом: для любой оценки $\{k_1, k_2\}$ из $K(N)$ выполняется условие $l_m \leq k_m$, $m = 1, 2$.

В процессе построения дерева решения определяющую роль играет текущее множество рекордных векторов-оценок R , которое формируется из верхних оценочных векторов. Наличие рекордного множества позволяет исключать заведомо неперспективные направления поиска.

Пусть \bar{T} — момент времени окончания обслуживания всех объектов согласно стратегии $P(N) = \{p_1, p_2, \dots, p_{r(N)}\}$. Для определения нижнего оценочного вектора будем использовать решение облегченной задачи. *Облегченной* назовем задачу, отличие которой от исходной в том, что одно-

временное обслуживание двух и более объектов разрешено и процессор может начать обслуживание объекта o_i , $i = \overline{1, n}$ в любой момент времени t ($t \geq t_i \geq \overline{T}$). Тогда нижний оценочный вектор будет рассчитываться по формуле

$$L(N) = \left\{ \sum_{j \notin M(N)} \varphi_j(\bar{t}(j, P(N))) + \sum_{l \in M(N)} \varphi_l(\max(\overline{T}, t_l) + \tau_l); \right. \\ \left. \max\left\{ \max_{j \notin M(N)} \psi_j(\bar{t}(j, P(N))), \max_{l \in M(N)} \psi_l(\max(\overline{T}, t_l) + \tau_l) \right\} \right\}. \quad (3.12)$$

Отметим, что в общем случае нижняя оценка является недостижимой.

Обозначим через $M(N)$ множество индексов объектов, для которых еще не определена очередность обслуживания, $|M(N)| = n - r(N)$. Через $P'(N) = \{p'_1, p'_2, \dots, p'_{n-r(N)}\}$ обозначим стратегию обслуживания объектов с индексами, входящими в множество $M(N)$, при которой объекты обслуживаются в порядке их поступления, т.е. $p'_1 < p'_2 < \dots < p'_{n-r(N)}$, но с учетом объемных ограничений (3.1). При условии, что объекты множества $M(N)$ обслуживаются в соответствии со стратегией $P'(N)$ и это обслуживание может начаться не раньше момента времени \overline{T} , верхний оценочный вектор рассчитывается по формуле

$$H(N) = \left\{ \sum_{j \notin M(N)} \varphi_j(\bar{t}(j, P(N))) + \sum_{l \in M(N)} \varphi_l(\bar{t}(l, P'(N))); \right. \\ \left. \max\left\{ \max_{j \notin M(N)} \psi_j(\bar{t}(j, P(N))), \max_{l \in M(N)} \psi_l(\bar{t}(l, P'(N))) \right\} \right\}. \quad (3.13)$$

На любой стадии построения дерева решений необходимо *выбирать вершину, в которой будет выполняться очередное ветвление*. При этом совокупность всех вершин дерева, как и в п. 3.3.1, делится на три группы: закрытые вершины, терминальные вершины и открытые вершины. Для каждого очередного ветвления выбирается открытая вершина N с наименьшим значением обеих координат h_1 и h_2 верхнего оценочного вектора $H(N) = (h_1, h_2)$. Если таких вершин несколько, из них выбирается вершина наибольшего ранга. Процесс решения задачи \mathcal{Z}^2 завершается в ситуации, когда в построенном фрагменте дерева вариантов не оказывается открытых

вершин.

Для исключения *бесперспективных для дальнейшего ветвления вершин* в дереве решений будем пользоваться следующим правилом.

Если для приписанной данной вершине нижней векторной оценки $L(N) = (l_1, l_2)$ в множестве R найдется векторная оценка (r_1, r_2) такая, что для $m = 1, 2$ выполняется условие $l_m \geq r_m$, то никакая из эффективных для задачи $Z^2(N)$ оценок пополнить совокупность R не сможет и вершину можно поместить в группу терминальных.

В качестве дополнительного инструмента отсева используем описанную в п. 3.3.1 операцию доминирования.

Опишем схему решения задачи Z^2 методом ветвей и границ, используя введенные выше построения.

Шаг 0. Ставим в соответствие вершине-корню N_0 задачу $Z^2(N_0)$, которая совпадает с исходной задачей. Вычисляются верхний и нижний оценочные векторы в единственной открытой вершине N_0 ; верхний оценочный вектор помещается в R .

Шаг 1. Если в построенном фрагменте дерева вариантов нет открытых вершин, то задача решена, иначе переходим к шагу 2.

Шаг 2. Осуществляется выбор вершины N для ветвления.

Шаг 3. Выполняя ветвление в вершине N , получаем $(n - r(N))$ новых открытых вершин; вершина N переходит в группу закрытых.

Шаг 4. В каждой новой вершине определяются нижний и верхний оценочные векторы по формулам (3.12) и (3.13) соответственно. Вновь полученный верхний оценочный вектор включается в имеющуюся совокупность R ; из таким образом пополненного множества R изымаются векторы, которые в пополненной совокупности оказались доминируемыми. В результате получаем новый состав текущего множества рекордных оценок R .

Шаг 5. Выполняется отсев заранее бесперспективных направлений, а соответствующие вершины переходят в группу терминальных.

Шаг 6. Выполняется переход к шагу 1.

Полученное в результате данной схемы текущее множество рекордных оценок R совпадает с полной парето-совокупностью эффективных в задаче Z^2 оценок.

3.3.2. Результаты вычислительных экспериментов

В таблице 3.7 представлены характеристики алгоритма динамического программирования и алгоритма ветвей и границ, полученные при решении задачи $\text{Flow1R}(\sum, \max)$ на тестовых наборах данных.

Таблица 3.7

Значения показателей быстродействия алгоритма BnB
при решении задачи $\text{Flow1R}(\sum, \max)$

n	DP	BnB				
	$t_{\text{avg}}, \text{с}$	$t_{\text{avg}}, \text{с}$	$t_{\text{min}}, \text{с}$	$t_{\text{max}}, \text{с}$	q_{avg}	g_{avg}
8	0.001	0.000	0.000	0.016	434	105
9	0.003	0.001	0.000	0.062	981	218
10	0.017	0.006	0.000	0.140	2126	431
11	0.099	0.031	0.000	0.922	4561	850
12	0.672	0.183	0.000	6.250	10572	1810
13	4.118	0.924	0.000	106.717	22234	3520
14	36.722	8.009	0.000	1865.336	51348	7609
15	372.074	46.122	0.000	3684.130	122100	16851
16	-	144.556	0.032	8239.035	240228	30818
17	-	298.953	0.095	14129.075	393933	46968

Полученные результаты сравнительных экспериментов позволяют сделать следующие выводы.

Метод ветвей и границ в среднем является более эффективным по сравнению с методом динамического программирования при любом значении величины размерности потока n . Сравнительная эффективность тем значительнее, чем больше число объектов в решаемой задаче. Но при этом не исключены ситуации, когда метод ветвей и границ может вырождаться в полный перебор. Это подтверждается результатами экспериментов. Так,

при обслуживании 15 объектов максимальное время решения задачи алгоритмом VnB составляет примерно час, в то же время максимальная продолжительность решения задачи алгоритмом DP составляет 14 минут.

Представленные экспериментальные данные также позволяют сделать вывод о том, что число вершин дерева ветвления обычно существенно меньше максимально возможного, что свидетельствует о достаточной эффективности применения алгоритма VnB в среднем.

§ 3.4. Полиномиально разрешимые подклассы задач обслуживания

Ввиду *NP*-трудности рассматриваемых задач зависимость продолжительности синтеза точного решения от размерности потока объектов экспоненциальна. Это ограничивает максимальную размерность задач, практически поддающихся решению. Соответственно, актуальной является проблема построения таких модификаций модели управления обслуживанием, которые, сохраняя адекватность описания прикладной специфике, порождают полиномиально разрешимые подклассы исследуемых оптимизационных задач. Именно таковыми, как будет ясно из нижеследующего изложения, являются модификации модели, учитывающие ограничения на допустимые величины опережений в очереди на обслуживание.

В данном параграфе рассматривается обобщение бикритериальной математической модели обслуживания потока объектов в системе с накопительно-расходным компонентом при наличии ограничений на порядок обслуживания объектов. Данный подход основан на введенных в [24] понятиях *d*-стратегии и *q*-стратегии, которые конструктивно реализуют концепцию параметризованной сложности. В работе [25] рассмотрена модификация метода *d*-стратегии для однокритериальной задачи обслуживания бинарного потока объектов в системе с накопительно-расходным компонентом.

3.4.1. Концепция $2d$ -стратегий

Модификация модели и постановка оптимизационной задачи

Известно, что объекты потока \mathcal{O}_n пронумерованы в порядке их поступления в систему обслуживания, $t_1 \leq t_2 \leq \dots \leq t_n$. Считаем, что при выполнении стратегии $S \in \Omega$ объект o_β опережает в обслуживании объект o_α , если $\alpha < \beta$, но объект с индексом β обслуживается раньше, чем объект с индексом α ; разность $d = \beta - \alpha$ назовем величиной опережения. Ниже полагаем, что величина опережения для объектов подпотока \mathcal{O}^+ не может превышать значения параметра d^+ , а для объектов подпотока \mathcal{O}^- — значения параметра d^- . Стратегию, удовлетворяющую данному ограничению, будем называть $2d$ -стратегией; множество таких стратегий будем обозначать как S_{2d} , при этом $S_{2d} \subseteq \Omega$.

Рассмотрим следующую оптимизационную задачу.

Задача Flow1Rd(*, *). *Найти полную совокупность оптимально-компромиссных оценок в бикритериальной проблеме минимизации двух критериев оценки с учётом того, что величина опережения для объектов подпотока \mathcal{O}^+ не превышает величину d^+ , а для объектов подпотока \mathcal{O}^- величина опережения не превышает d^- :*

$$\left\{ \min_{S \in S_{2d}} K_1(S), \min_{S \in S_{2d}} K_2(S) \right\}. \quad (3.14)$$

Здесь в качестве $K_1(S)$ и $K_2(S)$ могут выступать пары критериев в задачах Flow1R(\sum , max), Flow1R(\sum , \sum) и Flow1R(max, max) соответственно:

$$\star K_1(S) = \sum_{k: o_{i_k} \in \mathcal{O}^+} \varphi_{i_k}(\bar{t}(i_k, S)), K_2(S) = \max_{l: o_{i_l} \in \mathcal{O}^-} \psi_{i_l}(\bar{t}(i_l, S)) \text{ для задачи Flow1R}(\sum, \max);$$

$$\star K_1(S) = \sum_{k: o_{i_k} \in \mathcal{O}^+} \varphi_{i_k}(\bar{t}(i_k, S)), K_2(S) = \sum_{l: o_{i_l} \in \mathcal{O}^-} \psi_{i_l}(\bar{t}(i_l, S)) \text{ для задачи Flow1R}(\sum, \sum);$$

$$\star K_1(S) = \max_{k: o_{i_k} \in \mathcal{O}^+} \varphi_{i_k}(\bar{t}(i_k, S)), K_2(S) = \max_{l: o_{i_l} \in \mathcal{O}^-} \psi_{i_l}(\bar{t}(i_l, S)) \text{ для задачи Flow1R}(\max, \max).$$

Такая модификация исходных задач может быть применена в случае, когда при обслуживании бинарного потока объектов запрещен обгон в обслуживании одного вида объектов более чем d^+ (d^-) другими следующими за ним объектами того же вида соответственно.

З а м е ч а н и е 3.2. Рассматриваемая в данном разделе модификация математической модели может быть применена также для решения задач $\text{Set1R}(\sum, \max)$, $\text{Set1R}(\sum, \sum)$ и $\text{Set1R}(\max, \max)$, если учесть что объекты множества \mathcal{O}_n пронумерованы в порядке их значимости: каждый предшествующий объект «важнее» следующих за ним.

Построение рекуррентных соотношений алгоритма динамического программирования

Для построения рекуррентных соотношений динамического программирования воспользуемся введенными в п. 3.1.3 операциями $\text{eff}(M)$ и \otimes и определениями $F(t)$ и $D(t, \Delta)$ из п. 3.2.3.

Напомним, что операция \otimes объединения вектора и множества векторов для каждой задачи $\text{Flow1R}(\sum, \max)$, $\text{Flow1R}(\sum, \sum)$ и $\text{Flow1R}(\max, \max)$ определяется специфичным образом (таблица 3.1).

При решении рассматриваемой бикритериальной задачи (3.14) будем иметь в виду, что любой объект $o_j \in \mathcal{O}^+$ с индексом j может быть обойден в обслуживании только объектами с индексами из совокупности $\{j + 1, j + 2, \dots, j + d^+\}$. Аналогично объект $o_l \in \mathcal{O}^-$ с индексом l может быть обойден в обслуживании только объектами с индексами из совокупности $\{l + 1, l + 2, \dots, l + d^-\}$. При этом решения принимаются в те моменты времени, когда процессор свободен и необходимо выбрать следующий объект из входящего \mathcal{O}^+ или исходящего \mathcal{O}^- подпотока, учитывая текущее заполнение резервуара. Тогда пятерка значений (t, j, l, Q, V_t) определяет текущее состояние системы в момент t принятия решения, где j (l) — наименьший индекс в множестве индексов всех необслуженных к моменту

t объектов подпотока $\mathcal{O}^+(\mathcal{O}^-)$; Q — множество индексов объектов потока \mathcal{O}_n , ожидающих обслуживания в этот момент времени.

Пусть $E_{2d}(t, j, l, Q, V_t)$ — совокупность эффективных оценок, сгенерированная за период времени от момента t и до окончания процесса обслуживания всех объектов потока \mathcal{O}_n . Тогда полная совокупность эффективных оценок рассматриваемой задачи будет представлять собой множество $E_{2d}(0, \min_{j \in \mathcal{O}^+} j, \min_{l \in \mathcal{O}^-} l, F(0), V_0)$.

В текущем состоянии (t, j, l, Q, V_t) объект с индексом $\alpha \in Q$ допустим к обслуживанию при условии выполнения неравенств $V_t + v_\alpha \leq V^*$ и $\alpha - j \leq d^+$. Объект с индексом $\beta \in Q$ может быть обслужен, если $V_t - v_\beta \geq 0$ и $\beta - l \leq d^-$. Введем дополнительно следующие обозначения: Q^* — множество индексов объектов, допустимых к обслуживанию в состоянии (t, j, l, Q, V_t) , $Q^* \subseteq Q$; Q_t — совокупность индексов всех необслуженных к моменту времени t объектов; $\xi^+[Q_t]$ ($\xi^-[Q_t]$) — минимальный из положительных индексов объектов множества Q_t , принадлежащих подпотоку \mathcal{O}^+ (\mathcal{O}^-).

Очевидно, что для любого $\theta \geq 0$ и $Q = \{\alpha\}$, где α — индекс допустимого к обслуживанию объекта подпотока \mathcal{O}^+ , имеет место соотношение

$$E_{2d}(t_n + \theta, \alpha, 0, \{\alpha\}, V_{t_n + \theta}) = (\varphi_\alpha(t_n + \theta + \tau_\alpha), \psi_\alpha(t_n + \theta + \tau_\alpha)). \quad (3.15)$$

Аналогично соотношение

$$E_{2d}(t_n + \theta, 0, \beta, \{\beta\}, V_{t_n + \theta}) = (\varphi_\beta(t_n + \theta + \tau_\beta), \psi_\beta(t_n + \theta + \tau_\beta)) \quad (3.16)$$

выполняется для любого $\theta \geq 0$ и $Q = \{\beta\}$, где β — индекс допустимого к обслуживанию объекта подпотока \mathcal{O}^- .

Если в состоянии (t, j, l, Q, V_t) в качестве очередного обслуживаемого объекта выбран объект с индексом $\alpha \in Q^*$, то следующим моментом принятия решения будет $t + \tau_\alpha$. При этом множество индексов объектов, ожидающих обслуживания в этот момент времени, определяется как совокупность $(Q \setminus \{\alpha\}) \cup D(t, \tau_\alpha)$, а характеристика заполнения резервуара

изменится и будет равна $V_t + w_\alpha \cdot v_\alpha$. Суммируя приведенные рассуждения, запишем рекуррентное выражение

$$E_{2d}(t, j, l, Q, V_t) = \text{eff}\left\{ \bigcup_{\alpha \in Q^*} [(\varphi(t + \tau_\alpha), \psi(t + \tau_\alpha)) \otimes E(t + \tau_\alpha, \xi^+[Q_t \setminus \{\alpha\}], \xi^-[Q_t \setminus \{\alpha\}], (Q \setminus \{\alpha\}) \cup D(t, \tau_\alpha), V_t + w_\alpha \cdot v_\alpha)] \right\}, \quad (3.17)$$

образующее совместно с (3.15) и (3.16) замкнутую систему решающих задачу (3.14) соотношений динамического программирования.

Процесс синтеза полной совокупности эффективных по Парето оценок на основе построенных соотношений аналогичен описанному в п. 3.1.3.

Оценка трудоемкости алгоритма

Дадим верхнюю оценку числа элементарных операций, выполняемых при вычислениях по соотношениям (3.15)-(3.17) в процессе решения задачи (3.14).

Если в качестве пары критериев выступают критерии задачи $\text{Flow1R}(\sum, \max)$ или $\text{Flow1R}(\max, \max)$, то оценка вычислительной сложности характеризуется величиной $O(L2^\delta n \max_{i: o_i \in \mathcal{O}^-} \psi_i(L))$, где $\delta = \max(d^+, d^-)$. Действительно, в рассматриваемых состояниях (t, j, l, Q, V_t) число значений первого аргумента t сверху ограничено величиной $L = t_n + \sum_{i=1}^n \tau_i$. Два следующих аргумента j и l , а также V_t используются для выбора из Q допустимых для обслуживания объектов; эта операция выбора мажорируется линейной функцией от n . Из множества Q рассматриваются только те объекты, которые допустимы по объёмным или $2d$ -ограничениям с учетом наличия фиктивного объекта. Следовательно, количество таких объектов не может быть более, чем $\delta + 1$, $\delta = \max(d^+, d^-)$. Таким образом, число состояний системы не превышает $O(L2^\delta n)$. Максимальное из возможных значений вторых координат векторов-оценок в множествах $E_{2d}(t, j, l, Q, V_t)$ не может превышать $\max_{i: o_i \in \mathcal{O}^-} \psi_i(L)$. Поэтому общее число элементарных операций, выполняемых при определении каждого очередного множества $E_{2d}(t, j, l, Q, V_t)$, оценивается величиной $O(\max_{i: o_i \in \mathcal{O}^-} \psi_i(L))$.

Если в качестве пары критериев задачи (3.14) выступают критерии $\text{Flow1R}(\sum, \sum)$, то число элементарных операций, выполняемых при вычислениях по соотношениям (3.15)-(3.17), характеризуется величиной $O(L2^\delta n^2 \max_{i:o_i \in \mathcal{O}^-} \psi_i(L))$.

Алгоритм определения по принадлежащей множеству $E_{2d}(0, \min_{j \in \mathcal{O}^+} j, \min_{l \in \mathcal{O}^-} l, F(0), V_0)$ векторной оценке соответствующей парето-оптимальной стратегии имеет линейно зависящую от n оценку числа выполняемых элементарных операций. Полученные оценки трудоемкости процедур синтеза полных совокупностей эффективных оценок для задач $\text{Flow1R}(\sum, \max)$, $\text{Flow1R}(\sum, \sum)$ и $\text{Flow1R}(\max, \max)$ выражают идею параметризованной сложности.

Пример результата синтеза полной совокупности эффективных оценок

Рассмотрим пример решения задачи (3.14), в которой в качестве пары критериев оценки стратегий обслуживания выступают критерии задачи $\text{Flow1R}(\sum, \max)$. Необходимо построить полную совокупность эффективных оценок при $d^+ = d^- = 2$, $V^* = 200$, $V_0 = 85$, $\varphi_i(t) = a_i \cdot (t - \tau_i)$, $\psi_i(t) = \max(d_i - t, 0)$ и наборе значений параметров из таблицы 3.8.

Таблица 3.8

Исходные данные

i	1	2	3	4	5	6	7	8	9	10
t_i	0	3	11	12	12	14	18	25	34	39
τ_i	14	10	5	6	13	9	11	11	9	2
a_i	3	8	2	3	8	7	6	9	1	8
d_i	15	14	16	19	27	23	29	39	43	41
v_i	22	15	64	16	36	81	11	63	84	12
w_i	-1	-1	+1	+1	-1	+1	-1	-1	-1	-1

В таблице 3.9 приведены результаты синтеза по соотношениям (3.15)-(3.17) полной совокупности эффективных оценок и соответствующих им $2d$ -стратегией.

Таблица 3.9

Полная совокупность эффективных оценок и стратегий обслуживания

Эффективная оценка	Стратегия обслуживания
(1394, 47)	{1, 2, 5, 7, 3, 10, 4, 6, 8, 9}
(1379, 51)	{1, 2, 5, 7, 3, 10, 6, 4, 8, 9}
(1358, 62)	{1, 2, 5, 7, 3, 10, 6, 8, 4, 9}

Продолжительность решения рассматриваемого примера в условиях, описанных в приложении А, менее 1 с.

На рис. 3.1 представлены конфигурации расположения полных совокупностей эффективных оценок, построенных на данных рассматриваемого примера для моделей, характеризующихся парами $d^+ = 1, d^- = 1$; $d^+ = 3, d^- = 3$; $d^+ = 3, d^- = 5$, и без ограничений на допустимые величины опережений в очереди на обслуживание.

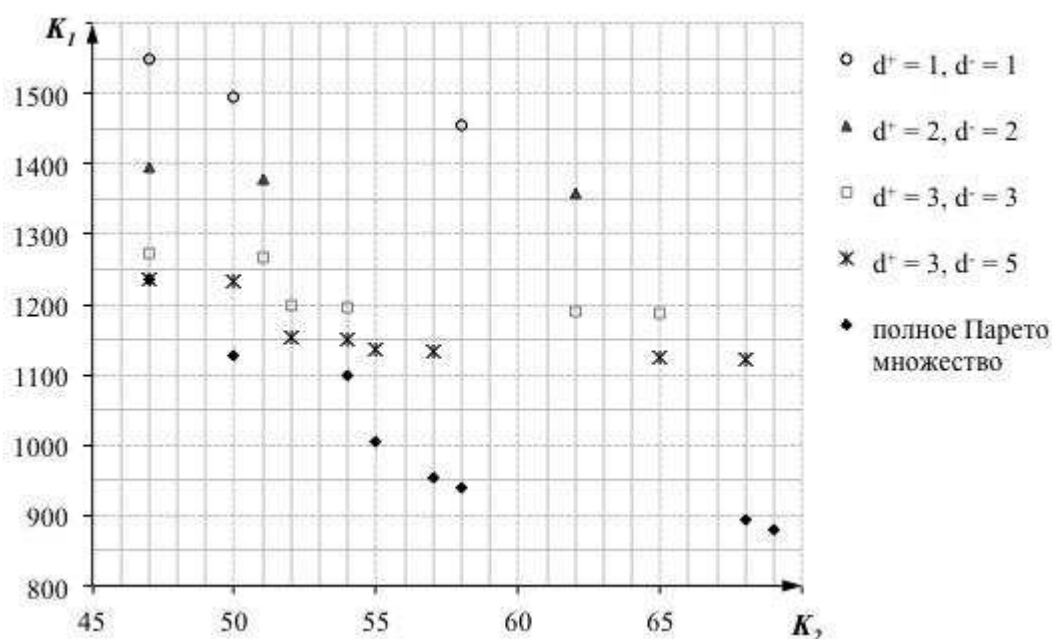


Рис. 3.1. Конфигурации расположения полных совокупностей эффективных оценок.

Как следует из характера расположения совокупностей эффективных оценок на рис. 3.1, с ростом значений параметров d^+ и d^- наблюдается в определенном смысле «приближение» конфигурации расположения совокупностей эффективных оценок к предельной конфигурации множества

оптимальных по Парето оценок, характеризующейся отсутствием ограничений на допустимые величины опережений в очереди на обслуживание. Данное обстоятельство позволяет трактовать $2d$ -стратегии в качестве приближенных решений задачи (3.14) в общем случае, т.е. без ограничений.

3.4.2. Концепция $2q$ -стратегий

Модификация модели и постановка оптимизационной задачи

Известно, что объекты потока \mathcal{O}_n пронумерованы в порядке их поступления в систему обслуживания. Считаем, что при реализации стратегии S объект o_α пропускает вперед на обслуживание объект o_β , если $\alpha < \beta$, но объект с индексом β обслуживается раньше, чем объект с индексом α . Ниже полагаем, что объект $o_i \in \mathcal{O}^+$ пропускает вперед на обслуживание не более q^+ объектов входящего подпотока, а объект $o_i \in \mathcal{O}^-$ пропускает вперед не более q^- объектов исходящего подпотока. Стратегию, удовлетворяющую данному ограничению, будем называть $2q$ -стратегией; множество таких стратегий будем обозначать как S_{2q} , при этом $S_{2q} \in \Omega$.

Рассмотрим следующую оптимизационную задачу.

Задача Flow1Rq(, *). Найти полную совокупность оптимально-компромиссных оценок в бикритериальной проблеме минимизации двух критериев оценки при условии, что объект подпотока \mathcal{O}^+ не пропускает вперед более q^+ объектов из того же подпотока, а объекты подпотока \mathcal{O}^- пропускают вперед не более q^- объектов из этого подпотока :*

$$\left\{ \min_{S \in S_{2q}} K_1(S), \min_{S \in S_{2q}} K_2(S) \right\}. \quad (3.18)$$

В качестве $K_1(S)$ и $K_2(S)$ могут выступать пары критериев также, как и в задачах Flow1R(\sum , max), Flow1R(\sum , \sum), Flow1R(max, max).

З а м е ч а н и е 3.3. Рассматриваемая в данном разделе модификация математической модели может быть применена также для решения задач Set1R(\sum , max), Set1R(\sum , \sum) и Set1R(max, max), если учесть что объекты множества \mathcal{O}_n пронумерованы в порядке их значимости: каждый предше-

ствующий объект «важнее» следующих за ним.

Построение рекуррентных соотношений алгоритма динамического программирования

Для построения рекуррентных соотношений динамического программирования воспользуемся введенными в п. 3.1.3 операциями $\text{eff}(M)$ и \otimes и определениями $F(t)$ и $D(t, \Delta)$ из п. 3.2.3.

При обслуживании объектов множества O_n решения принимаются в те моменты времени, когда процессор свободен и необходимо выбрать следующий для обслуживания объект из входящего или исходящего подмножества, учитывая текущее заполнение резервуара и ограничение на обход в обслуживании. Набор значений $(t, j, A_j^t, l, B_l^t, Q, V_t)$ определяет текущее состояние системы в момент t принятия решения, где j (l) — наименьший индекс в множестве индексов всех необслуженных к моменту t объектов подпотока O^+ (O^-); A_j^t (B_l^t) — множество индексов обслуженных объектов подпотока O^+ (O^-), обошедших на момент времени t объект с индексом j (l), $|A_j^t| \leq q^+$, $|B_l^t| \leq q^-$; Q — множество индексов объектов ожидающих обслуживания в этот момент времени.

Дополнительно введем следующие обозначения: Q_t — совокупность индексов всех необслуженных к моменту времени t объектов, т.е. $Q_t = Q \cup \{k : o_k \in O_n, t_k > t\}$; $\{Q_t\}_r$ — совокупность индексов объектов из множества Q_t , значения которых больше r ; $\xi^+[Q_t]$ ($\xi^-[Q_t]$) — минимальный из положительных индексов объектов множества Q_t принадлежащих подпотоку O^+ (O^-).

Пусть $E_{2q}(t, j, A_j^t, l, B_l^t, Q, V_t)$ — совокупность эффективных двумерных векторов-оценок, сгенерированная за период времени от момента t и до окончания процесса обслуживания всех объектов потока O_n . Тогда полная совокупность эффективных оценок исходной бикритериальной задачи (3.18) будет представлять собой множество $E_{2q}(0, \min_{j \in O_n} j, \emptyset, \min_{l \in O_n} l, \emptyset, F(0), V_0)$.

Очевидно, что для любого $\theta \geq 0$ и $Q = \{\alpha\}$, где α — индекс допустимого к обслуживанию объекта подпотока \mathcal{O}^+ , имеет место соотношение

$$E_{2q}(t_n + \theta, \alpha, A_\alpha^{t_n + \theta}, \emptyset, \mathcal{O}^-, \{\alpha\}, V_{t_n + \theta}) = (\varphi_\alpha(t_n + \theta + \tau_\alpha), \psi_\alpha(t_n + \theta + \tau_\alpha)). \quad (3.19)$$

Аналогично соотношение

$$E_{2q}(t_n + \theta, \emptyset, \mathcal{O}^+, \beta, B_\beta^{t_n + \theta}, \{\beta\}, V_{t_n + \theta}) = (\varphi_\beta(t_n + \theta + \tau_\beta), \psi_\beta(t_n + \theta + \tau_\beta)) \quad (3.20)$$

выполняется для любого $\theta \geq 0$ и $Q = \{\beta\}$, где β — индекс любого допустимого к обслуживанию объекта подпотока \mathcal{O}^- .

В текущем состоянии $(t, j, A_j^t, l, B_l^t, Q, V_t)$ системы обслуживания объект с индексом $\alpha \in Q$ из входящего подпотока \mathcal{O}^+ может быть допущен к обслуживанию только при условии выполнения неравенства $V_t + v_\alpha \leq V^*$ и если при $\alpha \neq j$ мощность множества $|A_j^t| < q^+$. Объект с индексом $\beta \in Q$ из исходящего подпотока \mathcal{O}^- может быть обслужен, если $V_t \geq v_\beta$ и если при $\beta \neq l$ мощность множества $|B_l^t| < q^-$. Обозначим через Q^* множество индексов объектов потока \mathcal{O}_n , допустимых к обслуживанию в состоянии $(t, j, A_j^t, l, B_l^t, Q, V_t)$, $Q^* \subset Q$.

Если в качестве обслуживаемого объекта будет выбран объект с индексом $\alpha \in Q^*$, то следующим моментом принятия решения будет $t + \tau_\alpha$; множество ожидающих обслуживания объектов будет $(Q \setminus \{\alpha\}) \cup D(t, \tau_\alpha)$, а характеристика заполнения резервуара изменится и станет равной сумме $V_t + w_\alpha \cdot v_\alpha$. При этом, если выбран объект из подпотока \mathcal{O}^+ , то наименьшим индексом объекта из входящего подпотока будет $j^* = \xi^+[Q_t \setminus \{\alpha\}]$, а множество опередивших объект с индексом j^* объектов входящего подпотока будет $\{A_j^t \cup \{\alpha\}\}_{j^*}$. Если выбран объект из подпотока \mathcal{O}^- , то наименьшим индексом объекта будет $l^* = \xi^-[Q_t \setminus \{\beta\}]$, множество опередивших объект с индексом l^* объектов исходящего подпотока будет $\{B_l^t \cup \{\beta\}\}_{l^*}$. С учетом

всего выше сказанного запишем выражение

$$\begin{aligned}
E_{2q}(t, j, A_j^t, l, B_l^t, Q, V_t) = \text{eff}\{ & \bigcup_{\alpha: \alpha \in Q^*, o_\alpha \in \mathcal{O}^+} [(\varphi(t + \tau_\alpha), \psi(t + \tau_\alpha)) \otimes \\
& E_{2q}(t + \tau_\alpha, j^*, \{A_j^t \cup \{\alpha\}\}_{j^*}, l, B_l^t, (Q \setminus \{\alpha\}) \cup D(t, \tau_\alpha), \\
& V_t + w_\alpha \cdot v_\alpha)], \bigcup_{\beta: \beta \in Q^*, o_\beta \in \mathcal{O}^-} [(\varphi(t + \tau_\beta), \psi(t + \tau_\beta)) \otimes E_{2q}(t + \tau_\beta, j, A_j^t, \\
& l^*, \{B_l^t \cup \{\beta\}\}_{l^*}, (Q \setminus \{\alpha\}) \cup D(t, \tau_\alpha), V_t + w_\alpha \cdot v_\alpha)]\},
\end{aligned} \tag{3.21}$$

образующее совместно с (3.19) и (3.20) замкнутую систему решающих задачу (3.18) соотношений динамического программирования.

Процесс синтеза полной совокупности эффективных по Парето оценок на основе построенных соотношений аналогичен описанному в п. 3.1.3.

Оценка трудоемкости алгоритма

Рассуждая аналогично п. 3.4.1, получим верхние оценки числа элементарных операций, выполняемых при вычислениях по соотношениям (3.19)-(3.21) в процессе решения задачи (3.18) в зависимости от пар критериев.

Если в качестве пары критериев выступают критерии задачи $\text{Flow1R}(\sum, \max)$ или $\text{Flow1R}(\max, \max)$, то оценка вычислительной сложности характеризуется величиной $O(L2^{q^+ + q^-} n \max_{i: o_i \in \mathcal{O}^-} \psi_i(L))$, где $L = t_n + \sum_{i=1}^n \tau_i$.

Если в качестве пары критериев выступают критерии задачи $\text{Flow1R}(\sum, \sum)$, то число элементарных операций характеризуется величиной $O(L2^{q^+ + q^-} n^2 \max_{i: o_i \in \mathcal{O}^-} \psi_i(L))$.

Алгоритм определения по принадлежащей множеству $E_{2q}(0, \min_{j \in \mathcal{O}_n} j, \emptyset, \min_{l \in \mathcal{O}_n} l, \emptyset, F(0), V_0)$ векторной оценке соответствующей парето-оптимальной стратегии имеет линейно зависящую от n оценку числа выполняемых элементарных операций.

3.4.3. Результаты вычислительных экспериментов

Все результаты экспериментов приведены для случая, когда в качестве пары критериев выступают критерии задачи $\text{Flow1R}(\sum, \max)$.

Для оценки временных характеристик алгоритма решения задачи (3.14) была проведена серия вычислительных экспериментов в условиях, описанных в приложении А. Параметры d^+ и d^- принимали одинаковые значения на интервале от 1 до 3. В соответствующих узлах решетки $(n, d^+ = d^-)$ генерировались по равномерному закону распределения данные для серии из 1000 частных реализаций задачи (3.14). Для каждой такой серии определялось среднее время решения одной задачи. Результаты экспериментов представлены в таблице 3.10; при этом в правой колонке «Без ограничений» приведены данные для общего случая задачи, описанного в § 3.2.

Таблица 3.10

Оценки быстродействия алгоритма DP
при решении задачи $\text{Flow1Rd}(\sum, \max)$, с

n	$d = 1$	$d = 2$	$d = 3$	Без ограничений
8	0.000	0.001	0.001	0.001
9	0.001	0.002	0.004	0.003
10	0.002	0.007	0.018	0.017
11	0.007	0.028	0.073	0.099
12	0.032	0.138	0.377	0.672
13	0.141	0.606	1.692	4.118
14	0.756	3.557	10.505	36.722
15	2.062	9.139	103.465	372.074

Для оценки временных характеристик алгоритма решения задачи (3.18) была проведена серия вычислительных экспериментов в условиях, аналогичных описанным выше. Результаты экспериментов представлены в таблице 3.11.

Как видно из приводимых в таблицах 3.10 и 3.11 данных, продолжительность синтеза полной совокупности эффективных оценок в модели с учетом ограничений на допустимые величины опережений в очереди на обслуживание существенно меньше продолжительности решения общей задачи без учета ограничений. В частности, для практически используемых

Таблица 3.11

Оценки быстродействия алгоритма DP
при решении задачи $\text{Flow1R}(\sum, \max)$, с

n	$q = 1$	$q = 2$	$q = 3$	Без ограничений
8	0.000	0.001	0.001	0.001
9	0.002	0.004	0.009	0.003
10	0.004	0.016	0.045	0.017
11	0.016	0.078	0.178	0.099
12	0.072	0.278	0.813	0.672
13	0.367	1.256	3.465	4.118
14	1.395	7.543	23.456	36.722
15	4.134	32.908	263.095	372.074

значений $d^+ = d^- = 2$ разработанный алгоритм позволяет сократить время решения задачи на порядок, а для случаев повышенной размерности потока объектов — на два порядка. Также из результатов экспериментов следует, что концепция $2d$ -стратегий порождает стратегии обслуживания примерно в 2 раза быстрее, чем концепция $2q$ -стратегий.

§ 3.5. Синтез субоптимальных решений задач обслуживания

В практических приложениях допустимая производственным регламентом продолжительность синтеза оптимальной стратегии обслуживания бинарного потока объектов сильно ограничивает размерность задач, поддающихся точному решению. Поэтому существенный интерес представляет разработка быстро работающих алгоритмов построения субоптимальных решений с приемлемыми для приложений погрешностями.

Данный параграф посвящен алгоритмам решения задач $\text{Flow1R}(\sum, \max)$, $\text{Flow1R}(\sum, \sum)$ и $\text{Flow1R}(\max, \max)$ повышенной размерности с использованием следующих метаэвристических концепций мягких вычислений: эволюционно-генетическая парадигма, имитация отжига, поиск с запретами. Для описания этих алгоритмов введем следующие обозначения.

В качестве особи для эволюционно-генетического алгоритма или текущего решения в алгоритмах имитации отжига и поиска с запретами будем считать произвольную допустимую стратегию обслуживания $S = \{i_1, i_2, \dots, i_n\}$ совокупности объектов потока \mathcal{O}_n , $S \in \Omega$. Данная стратегия оценивается следующими парами критериев:

$$\star K_1(S) = \sum_{k: o_{i_k} \in \mathcal{O}^+} \varphi_{i_k}(\bar{t}(i_k, S)), K_2(S) = \max_{l: o_{i_l} \in \mathcal{O}^-} \psi_{i_l}(\bar{t}(i_l, S)) \text{ для задачи Flow1R}(\sum, \max);$$

$$\star K_1(S) = \sum_{k: o_{i_k} \in \mathcal{O}^+} \varphi_{i_k}(\bar{t}(i_k, S)), K_2(S) = \sum_{l: o_{i_l} \in \mathcal{O}^-} \psi_{i_l}(\bar{t}(i_l, S)) \text{ для задачи Flow1R}(\sum, \sum);$$

$$\star K_1(S) = \max_{k: o_{i_k} \in \mathcal{O}^+} \varphi_{i_k}(\bar{t}(i_k, S)), K_2(S) = \max_{l: o_{i_l} \in \mathcal{O}^-} \psi_{i_l}(\bar{t}(i_l, S)) \text{ для задачи Flow1R}(\max, \max).$$

При описании алгоритмов синтеза субоптимальных стратегий будем пары критериев обозначать как $(K_1(S), K_2(S))$ вне зависимости от рассматриваемой задачи, но, имея в виду, что способ их вычисления специфичен для каждой задачи. Дополнительно воспользуемся операцией восстановления допустимости решения, введенной в § 2.4.

Очевидно, что при использовании алгоритмов синтеза субоптимальных решений возникает необходимость оценки качества алгоритмов, чтобы определить, насколько значения оценок на построенном ими множестве стратегий отличаются от оптимальных по Парето.

В качестве способа оценки предлагается следующий. Обозначим через \mathbf{M} множество парето-оптимальных оценок, а через \mathbf{M}^* — множество квазипарето-оптимальных оценок. Тогда «отклонение» квазипаретовского множества от парето-множества будем вычислять по формуле [19]

$$\Delta(\mathbf{M}, \mathbf{M}^*) = \frac{\text{avg}_{(K_1, K_2) \in \mathbf{M}} \left(\min_{(K_1^*, K_2^*) \in \mathbf{M}^*} \rho((K_1, K_2); (K_1^*, K_2^*)) \right)}{\text{avg}_{(K_1, K_2) \in \mathbf{M}} (\rho((K_1, K_2); (0, 0)))}, \quad (3.22)$$

где $\rho((K_1, K_2); (K_1^*, K_2^*)) = \max(|K_1 - K_1^*|, |K_2 - K_2^*|)$ — расстояние

между двумя оценками, avg — операция вычисления среднего значения, $\rho((K_1, K_2); (0, 0)) = \max(K_1, K_2)$ — соответственно расстояние между оценками (K_1, K_2) и $(0, 0)$.

Таким образом, для каждой оценки из множества Парето M находим ближайшую к ней оценку из квазипаретовского множества M^* , затем проводим усреднение по всем оценкам из M и полученную величину «нормируем» на среднее значение расстояния эффективных оценок до нуля.

При этом, если для некоторой оценки из совокупности M расстояние до ближайшей оценки из совокупности M^* превышает некоторую наперед заданную величину ε , то ее вклад при нахождении среднего отклонения $\Delta(M, M^*)$ не учитывается, но при этом увеличивается число оценок L_ε , потерянных рассматриваемым алгоритмом приближенного синтеза. Качество субоптимальных решений тем лучше, чем меньшее число оценок L_ε из парето-совокупности было потеряно, и при этом минимизировано среднее относительное отклонение $\Delta(M, M^*)$.

Такой способ оценки определяет отклонение для тех точек квазипарето-множества, которые были приближены с заданной точностью аппроксимации.

3.5.1. Эволюционно-генетический алгоритм

Рассмотрим особенности каждого этапа эволюционно-генетического алгоритма в применении к задачам $\text{Flow1R}(\sum, \max)$, $\text{Flow1R}(\sum, \sum)$ и $\text{Flow1R}(\max, \max)$.

1. Инициализация начальной популяции $P^0 = \{S_1, S_2, \dots, S_{N_p}\}$ численностью N_p .

Сгенерировать случайным образом N_p особей S_j , $j = \overline{1, N_p}$.

Оценить каждую особь популяции с помощью функции приспособленности, которая представляет собой вектор-функцию $(K_1(S_j), K_2(S_j))$, $j = \overline{1, N_p}$.

Положить номер текущей популяции t равным 0.

2. Выбор особей, которые переходят в следующую популяцию.

Отобрать особи, значения функции приспособленности которых входят в множество

$$\text{eff}(\{(K_1(S_j), K_2(S_j)) : S_j \in P^t\}).$$

По итогам такой селекции будет выбрано W особей, которые перейдут в следующую популяцию. Число W может быть различным для разных популяций.

3. Воспроизводство $k_1(N_p - W)$ потомков согласно упорядоченному оператору скрещивания.

Случайным образом из текущей популяции P^t выбрать особи двух родителей. В каждой родительской особи случайно выбрать «разрезающую» точку. Скопировать левый сегмент одного родителя в потомка, остальные позиции потомка взять из другого родителя в упорядоченном виде слева направо, исключая те элементы, которые уже есть в потомке. Выполнить восстановление допустимости решения. Сгенерировать второго потомка подобным же образом. Оценить потомков с помощью функции приспособленности [14].

В итоге в следующую популяцию перейдет потомок с тем значением функции приспособленности, которое доминирует значение функции приспособленности другого потомка. Если значения функций приспособленности обоих потомков не доминируют друг друга, то в следующую популяцию перейдет потомок выбранный случайно.

4. Воспроизводство $k_2(N_p - W)$ потомков согласно оператору мутации.

Случайным образом из текущей популяции P^t выбрать особь родителя. В родительской особи случайно выбрать позицию, значение элемента в этой позиции обменять со значением рядом расположенного элемента и

выполнить восстановление допустимости решения.

Полученный таким образом потомок перейдет в следующую популяцию только, если значение его функции приспособленности доминирует значение функции приспособленности родителя; иначе в следующую популяцию копируется родительская особь.

5. Воспроизводство $k_3(N_p - W)$ потомков согласно оператору инверсии.

Из текущей популяции P^t выбрать особь родителя и случайным образом определить в ней «разрезающую» точку. Сформировать потомка из родителя путем инверсии сегмента, который расположен справа после «разрезающей» точки. Выполнить восстановление допустимости решения [14].

Потомок перейдет в следующую популяцию, если значение его функции приспособленности доминирует значение функции приспособленности родителя; иначе в следующую популяцию копируется родительская особь.

6. Замена текущей популяции P^t новой популяцией P^{t+1} .

В новую популяцию переходят:

- W особей, отобранных на шаге 2;
- $k_1(N_p - W)$ особей, появившихся в результате скрещивания;
- $k_2(N_p - W)$ особей, появившихся в результате мутации;
- $k_3(N_p - W)$ особей, появившихся в результате инверсии.

Считаем константы k_1 , k_2 и k_3 заданными; при этом $k_1 + k_2 + k_3 = 1$ и $0 \leq k_l \leq 1, l = \{1, 2, 3\}$.

Таким образом, каждая популяция P^t имеет постоянную численность, равную N_p .

7. Увеличить номер текущего поколения $t = t + 1$ и проверить условие останова алгоритма.

Критерием останова является заданное число генераций или неизменность множества особей, выбранных на шаге 2 алгоритма на протяжении заданного числа генераций.

Если критерий останова выполнен, то реализуется отбор особей с квазипарето-эффективными значениями функции приспособленности. Отобранные особи будут соответствовать субоптимальным стратегиям обслуживания, а их функции приспособленности — значениям оптимизируемых критериев.

3.5.2. Алгоритм имитации отжига

Алгоритм отжига или симуляции восстановления (Simulated Annealing) построен на основе физического процесса восстановления, заключающегося в нагреве и последующем охлаждении некоторой субстанции с целью получения прочной кристаллической структуры [18, 113]. Алгоритм отжига делится на пять этапов.

1 этап. Создается начальное решение S_0 , которое представляет собой случайно сгенерированное допустимое решение задачи. Для этого стратегия сначала инициализируется последовательностью чисел $\{1, 2, \dots, n\}$, а затем выполняется n случайных перестановок (элементы произвольно меняются местами), после этого восстанавливается допустимость решения. Начальное решение именуется текущим и помещается в множество субоптимальных решений M .

2 этап. Выполняется оценка решения S_j , которая состоит в декодировании стратегии и расчете значений пары критериев $(K_1(S_j), K_2(S_j))$.

3 этап. Выполняется случайный поиск решения. Для этого текущее решение S_j копируется в рабочее S'_j , затем рабочее решение произвольно модифицируется. Чтобы сохранить целостность решения, после модификации два элемента в рабочей стратегии произвольно переставляются один раз, затем выполняется восстановление допустимости решения. Полученное рабочее решение оценивается.

4 этап. На данном этапе мы имеем два решения: текущее S_j и только что модифицированное рабочее S'_j . С каждым решением связана

определенная энергия, представляющая собой значения пары критериев $(K_1(S_j), K_2(S_j))$ и $(K_1(S'_j), K_2(S'_j))$ соответственно.

Если рабочее решение имеет недоминируемое значение пары критериев по отношению к текущему, т.е. $K_1(S'_j) \leq K_1(S_j)$ или $K_2(S'_j) \leq K_2(S_j)$, то рабочее решение копируется в текущее и добавляется в множество субоптимальных решений без нарушения эффективности, т.е. $M = \text{eff}\{M \cup (K_1(S'_j), K_2(S'_j))\}$.

Но если рабочее решение доминируется текущим, то определяется критерий допуска с тем, чтобы выяснить, что делать с рабочим решением. Вероятность допуска $P(S'_j)$ основывается на уравнении, базирующемся на законе термодинамики $P(S'_j) = e^{-\Delta/T} > R$, где T — текущее значение температуры, R — случайно выбранное число из диапазона $(0, 1)$, а Δ вычисляется по формуле $\Delta = (K_1(S'_j) - K_1(S_j))^2 + (K_2(S'_j) - K_2(S_j))^2$.

5 этап. На пятом этапе алгоритма происходит снижение температуры, для этого используется простая функция $T_{i+1} = \alpha T_i$, где $\alpha < 1$. При одной температуре выполняется заданное количество итераций алгоритма. Процесс продолжается до тех пор, пока температура не достигнет нуля.

По завершении алгоритма множество M будет представлять собой совокупность субоптимальных стратегий обслуживания потока объектов в однопроцессорной системе с накопительно-расходным компонентом.

3.5.3. Алгоритм поиска с запретами

Алгоритм поиска с запретами относится к классу алгоритмов локального поиска и является одним из наиболее эффективных средств для решения задач дискретной оптимизации. В основе метода поиска с запретами TS (Taboo Search) лежит схема, позволяющая осуществлять поиск глобального оптимума, не останавливаясь в локальном оптимуме; при этом основной идеей является формирование в процессе поиска так называемого списка запретов, который блокирует исследование части или всей окрестности те-

кущего решения [106].

Под окрестностью $\mathcal{N}(S_j)$ решения S_j будем понимать $\mathcal{N}(S_j) = \{(i'_1, i'_2, \dots, i'_n) : \exists f, k \in \{1, \dots, n\} : i'_f = i_k, i'_k = i_f, i'_m = i_m \text{ для } \forall m \in \{1, \dots, n\}, m \neq f, k\}$ — множество стратегий, полученных из текущего решения S_j перестановкой двух соседних элементов.

Введем операцию *добавления без нарушения эффективности* [91]. Пусть имеется оценка (K_1, K_2) и множество парето-оптимальных оценок \mathbf{M} , т.е. $\mathbf{M} = \text{eff}(\mathbf{M})$. В результате применения операции добавления без нарушения эффективности оценки (K_1, K_2) к множеству \mathbf{M} получим множество $\mathbf{M} = \text{eff}(\mathbf{M} \cup \{(K_1, K_2)\})$. Если в множестве \mathbf{M} существуют оценки, доминируемые оценкой (K_1, K_2) , то эти оценки удаляются из \mathbf{M} , а оценка (K_1, K_2) добавляется; будем называть эту ситуацию успешной. Если же оценка (K_1, K_2) доминируется некоторой оценкой множества \mathbf{M} , то она не добавляется; будем называть эту ситуацию неуспешной.

Дополнительно в процессе поиска с запретами потребуется сравнивать некоторую оценку $(K_1(S_j), K_2(S_j))$ с другой оценкой $(K_1(S'_j), K_2(S'_j))$. Будем говорить, что оценка $(K_1(S_j), K_2(S_j))$ проходит операцию *сравнения с эталоном*, если $(K_1(S'_j) - K_1(S_j))^2 + (K_2(S'_j) - K_2(S_j))^2 \leq \delta$, где δ — наперед заданное оценочное число. Иначе будем считать, что оценка не проходит операцию сравнения с эталоном.

На j -м шаге алгоритма осуществляется переход из текущего решения S_j в соседнее S'_j , которое имеет значение оценки $(K_1(S'_j), K_2(S'_j))$ не хуже, чем значение оценки текущего решения $(K_1(S_j), K_2(S_j))$, т.е. $K_1(S'_j) \leq K_1(S_j)$ или $K_2(S'_j) \leq K_2(S_j)$. Для того чтобы алгоритм не остановился в точке локального оптимума, решение S_j удаляется из дальнейшего рассмотрения. Алгоритм хранит такие решения за последние l шагов и на каждом следующем шаге запрещает движение в этих направлениях. Упорядоченный список $\Phi_j = \{S_j, S_{j-1}, \dots, S_{j-l+1}\}$ будем называть *списком запретов*. Список запретов удаляет из окрестности $\mathcal{N}(S_j)$ не более l точек;

обозначим через $\mathcal{N}(S_j, \Phi_j)$ множество незапрещенных точек окрестности.

Пусть \mathbf{M} — множество субоптимальных оценок, найденных в процессе решения. Тогда с учетом введенных выше пояснений схема алгоритма поиска с запретами может быть представлена следующим образом.

1. Выбрать в качестве начального решения стратегию $S_0 = \{1, \dots, n\}$ обслуживания объектов в порядке их поступления. Если полученное решение недопустимо, восстановить допустимость решения. Положить список запретов Φ_0 пустым, текущее решение S равным S_0 , множество \mathbf{M} состоящим из единственной оценки $(K_1(S_0), K_2(S_0))$, объявляемой на данном шаге эталоном для сравнения, $j = 0$.

2. Выполнять пока не сработал критерий остановки:

2.1. Сформировать окрестность текущего решения $\mathcal{N}(S_j, \Phi_j)$;

2.2. Для каждой точки S'_j из окрестности текущего решения S_j вычислить значения функций штрафов $(K_1(S'_j), K_2(S'_j))$ и применить операцию добавления без нарушения эффективности оценки $(K_1(S'_j), K_2(S'_j))$ к множеству \mathbf{M} .

Если операция неуспешна и оценка $(K_1(S'_j), K_2(S'_j))$ не проходит сравнения с эталоном, то добавить решение S'_j в список запретов.

Если операция успешна, то считать оценку $(K_1(S'_j), K_2(S'_j))$ эталоном для сравнения, а решение S'_j добавить в список запретов.

2.3. Если эталон изменился, то положить текущее решение равным решению, соответствующему эталону $S_{j+1} = S'_j$. Иначе в качестве текущего взять решение, случайно выбранное из множества \mathbf{M} .

2.4. Обновить список запретов Φ_j и счетчик $j = j + 1$.

3. Предъявить множество \mathbf{M} в качестве решения.

В качестве критерия останова алгоритма используется остановка по общему числу шагов либо шагов, в ходе которых не меняется значение эталона. Величина l является управляющим параметром алгоритма, ее значение подбирается экспериментально.

3.5.4. Результаты вычислительных экспериментов

Целью проведения массовых экспериментов являлось определение оценок быстродействия и точности разработанных алгоритмов эволюционно-генетического GA, имитации отжига SA и поиска с запретами TS.

Количество обслуживаемых объектов n изменялось от 8 до 30. Для значений n , приемлемых для решения точными методами, в каждой задаче сначала строилась полная совокупность эффективных по Парето оценок и множество соответствующих им стратегий алгоритмом динамического программирования DP и алгоритмом ветвей и границ BnB. Затем полученные задачи и соответствующие им решения использовались для оценки точности алгоритмов GA, SA и TS.

В таблице 3.12 приведены средние длительности отработки алгоритмов на примере решения задачи $\text{Flow1R}(\sum, \max)$. Для задач $\text{Flow1R}(\sum, \sum)$ и $\text{Flow1R}(\max, \max)$ результаты имеют тот же порядок.

Таблица 3.12

Зависимости средней длительностей отработки алгоритмов решения задачи $\text{Flow1R}(\sum, \max)$ от размерности потока объектов \mathcal{O}_n , с

n	DP	BnB	SA	GA	TS
8	0.001	0.000	0.655	0.627	0.799
9	0.003	0.001	0.796	0.891	0.918
10	0.017	0.006	0.954	1.254	1.573
11	0.099	0.031	1.110	1.698	1.899
12	0.672	0.183	1.293	2.405	2.918
13	4.118	0.924	1.474	3.081	4.230
14	36.722	8.009	1.678	4.861	5.773
15	372.074	46.122	2.254	7.523	7.123
16	-	144.556	11.874	10.141	12.364
17	-	298.953	13.273	17.098	16.097
25	-	-	21.256	39.258	35.365
30	-	-	289.316	414.698	367.163

Настраиваемые параметры метаэвристических алгоритмов были сле-

дующими.

— Для алгоритма SA коэффициент изменения температуры $\alpha = 0.99$, количество итераций при одном значении температуры $100n$.

— Для алгоритма GA размер популяции брался равным $100n$, максимальное число генераций популяций — $200n$, число генераций, в течение которых допустима неизменность отобранных хромосом 50.

— Для алгоритма TS размер списка запретов полагался равным $2^n/n$, общее число шагов — $100n$, число шагов в течение которых допустима неизменность эталона для сравнения, 50.

На рис. 3.2 представлены результаты сравнения субоптимальных решений с оптимальными, которые оценивались по нормированному отклонению $\Delta(M, M^*)$, вычисляемому по формуле (3.22).

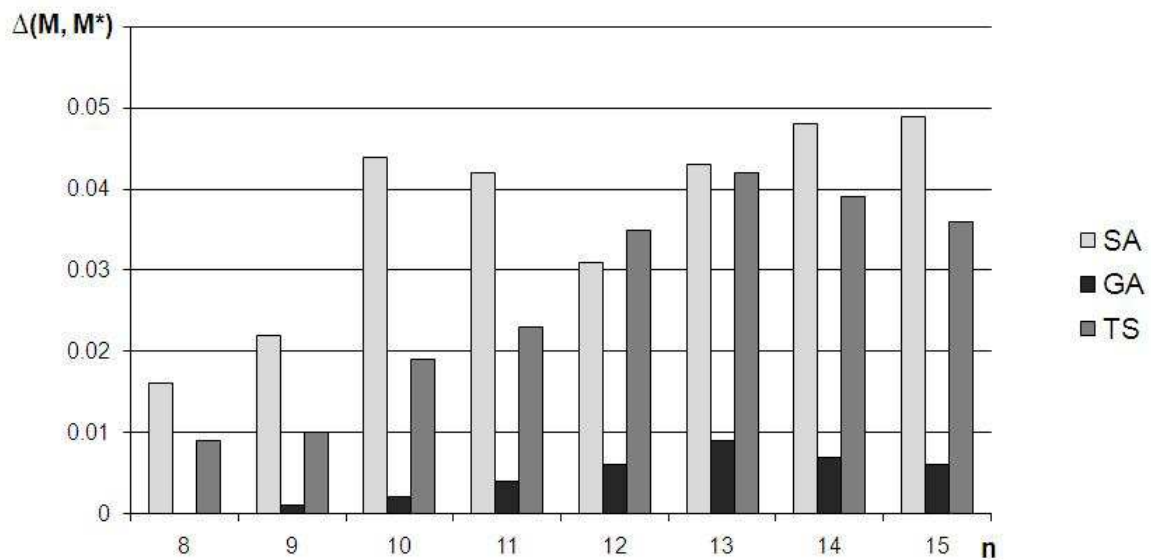


Рис. 3.2. Диаграмма статистики отклонений от точных решений приближенных, полученных алгоритмами GA, SA и TS.

Естественно, что на разных наборах данных одним и тем же метаэвристическим алгоритмом могут быть получены стратегии разного качества. Поэтому имеет смысл говорить о качестве стратегии, синтезируемой конкретной метаэвристикой, лишь в усредненном смысле. Более того, точность

каждого алгоритма можно улучшить, изменив его характеристики и тем самым увеличив среднее время решения одной задачи.

Выводы по главе 3

1) Построены бикритериальные модели обслуживания множества и потока объектов стационарным процессором с накопительно-расходным компонентом.

2) Поставлены бикритериальные задачи синтеза оптимальных стратегий обслуживания для каждой модели, различающиеся типами оптимизируемых критериев.

3) Сконструированы решающие процедуры для задач п. 2, основанные на идеологии динамического программирования и метода ветвей и границ, получены оценки их трудоёмкости, приведены результаты массовых вычислительных экспериментов.

4) Предложены полиномиальные подклассы моделей, учитывающие естественные с точки зрения приложений ограничения на структуру допустимых стратегий обслуживания, и разработаны алгоритмы решения соответствующих оптимизационных задач.

5) Для синтеза субоптимальных стратегий обслуживания разработаны метаэвристические алгоритмы, основанные на эволюционно-генетическом подходе, имитации отжига и поиске с запретами.

Глава 4. Модели, бикритериальные задачи и алгоритмы синтеза стратегий обслуживания потока объектов в однопроцессорной системе с двумя накопительно-расходными компонентами

Описывается математическая модель обслуживания объектов в однопроцессорной системе с двумя накопительно-расходными компонентами. Формулируются бикритериальные задачи оптимизации. Разрабатываются алгоритмы синтеза парето-оптимальных стратегий управления обслуживанием на основе многокритериального расширения метода динамического программирования и идеологии ветвей и границ. Приводятся оценки трудоемкости предложенных алгоритмов.

§ 4.1. Математическая модель

Рассматривается конечный детерминированный поток независимых объектов $\mathcal{O}_n = \{o_1, o_2, \dots, o_n\}$, подлежащих однократному однофазному обслуживанию стационарным процессором с двумя накопительными компонентами. Каждый накопительно-расходный компонент R_μ процессора представляет собой резервуар ограниченной емкости V_μ^* , предназначенный для промежуточного хранения продукта вида μ ($\mu = \{1, 2\}$). Поток \mathcal{O}_n , обладающий свойством бинарности, состоит из двух подпотоков: входящего \mathcal{O}^+ и исходящего \mathcal{O}^- таких, что $\mathcal{O}^+ \cup \mathcal{O}^- = \mathcal{O}_n$ и $\mathcal{O}^+ \cap \mathcal{O}^- = \emptyset$.

Для каждого объекта o_i , $i = \overline{1, n}$ определены следующие целочисленные параметры: t_i — момент поступления в очередь на обслуживание; τ_i — норма длительности обслуживания; d_i — мягкий директивный срок завершения обслуживания ($d_i \geq t_i + \tau_i$); v_i — объемная характеристика (вместимость); w_i — принадлежность объекта o_i тому или иному подпотоку ($w_i = +1$, если $o_i \in \mathcal{O}^+$, и $w_i = -1$, если $o_i \in \mathcal{O}^-$). Тип λ_i объекта однозначно характеризуется видом продукта μ , для транспортировки которого он предназначен, $\lambda_i = \{1, 2\}$. При этом в процессе обслуживания потока \mathcal{O}_n , объекты, предназначенные для транспортировки продукта

вида 1, используют только компонент R_1 . Аналогично объекты потока \mathcal{O}_n , предназначенные для транспортировки продукта вида 2, используют только компонент R_2 . Объекты пронумерованы в порядке их поступления, т.е. $0 \leq t_1 \leq \dots \leq t_n$. С каждым объектом o_i ассоциируются две монотонно возрастающие (в нестрогом смысле) функции индивидуального штрафа $\varphi_i(t)$ и $\psi_i(t)$, выражающие величины потерь, зависящие от времени завершения обслуживания объекта.

Процессор готов к обслуживанию потока \mathcal{O}_n в момент времени $t = 0$. Обслуживание объекта o_i , $i = \overline{1, n}$ может быть начато свободным процессором в любой момент времени t , $t \geq t_i$ и осуществляется без прерываний; необслуженный объект не может покинуть очередь; одновременное обслуживание процессором двух и более объектов и его непроизводительные простои запрещены.

В момент времени t объемная величина заполнения компонента R_μ характеризуется переменной $V_\mu(t)$ с начальным значением $V_\mu(0)$. Обозначим через $cv(\mu, \alpha)$, где α — произвольный индекс объекта из потока \mathcal{O}_n , функцию изменения характеристики заполнения резервуаров:

$$cv(\mu, \alpha) = \begin{cases} 0, & \text{если } \lambda_\alpha \neq \mu \\ w_\alpha \cdot v_\alpha, & \text{если } \lambda_\alpha = \mu. \end{cases} \quad (4.1)$$

Обслуживание любого объекта из входящего подпотока \mathcal{O}^+ может быть начато при наличии достаточного свободного объема в соответствующем ему резервуаре; обслуживание любого объекта из исходящего подпотока \mathcal{O}^- может быть начато при наличии достаточного количества продукта в соответствующем резервуаре. Таким образом, объект с индексом α из потока \mathcal{O}_n допустим к обслуживанию, если выполняются *объемные ограничения*

$$0 \leq V_\mu(t) + cv(\mu, \alpha) \leq V_\mu^*, \mu = \{1, 2\}. \quad (4.2)$$

Стратегия обслуживания объектов S представляет собой произвольную перестановку $S = \{i_1, i_2, \dots, i_n\}$ совокупности индексов $N =$

$\{1, 2, \dots, n\}$; при её реализации объект с индексом i_k обслуживается k -м по очереди ($k = \overline{1, n}$). Стратегию S именуем допустимой, если удовлетворяются отмеченные выше объемные ограничения на обслуживание объектов. Множество всех допустимых стратегий обслуживания обозначим через Ω .

Здесь и ниже будем считать истинными всегда выполняющиеся в практических приложениях неравенства

$$2 \cdot v_i \leq \min(V_1^*, V_2^*), i = \overline{1, n}. \quad (4.3)$$

При выполнении этого неравенства имеет место следующая теорема.

Теорема 4.1. *Необходимым и достаточным условием непустоты множества Ω является выполнение следующих неравенств:*

$$\begin{aligned} 0 &\leq V_1(0) + \sum_{i=1}^n cv(1, i) \leq V_1^*; \\ 0 &\leq V_2(0) + \sum_{i=1}^n cv(2, i) \leq V_2^*. \end{aligned} \quad (4.4)$$

Доказательство. Условия теоремы являются необходимыми. Действительно, если $V_1(0) + \sum_{i=1}^n cv(1, i) < 0$, то при любой стратегии система окажется в состоянии, при котором для обслуживания следующего объекта типа $\lambda_i = 1$ из исходящего подмножества \mathcal{O}^- недостаточно продукта в резервуаре R_1 и больше не осталось объектов, способных пополнить этот резервуар. Если $V_1(0) + \sum_{i=1}^n cv(1, i) > V_1^*$, то при любой стратегии система окажется в состоянии, при котором обслуживание следующего объекта типа $\lambda_i = 1$ из входящего подмножества \mathcal{O}^+ невозможно, поскольку в резервуаре R_1 недостаточно места и не осталось объектов, способных забрать часть продукта из этого резервуара. Аналогичные рассуждения применимы и по отношению к резервуару R_2 .

Перейдем к доказательству достаточности. Допустим, что при соблюдении введенных условий множество допустимых решений задачи пусто. Это значит, что в процессе решения возникла ситуация, при которой невозможно обслужить любой из оставшихся объектов, так как объемные огра-

ничения не выполняются. Обозначим через M множество индексов оставшихся необслуженными объектов; в этом случае реализуется одна из следующих ситуаций.

1) В множестве M присутствуют только индексы объектов входящего подмножества O^+ типа $\lambda_i = 1$, и в результате обслуживания любого из оставшихся объектов резервуар R_1 переполнится, т.е. $V_1(t) + v_j > V_1^*$ для каждого $j \in M$. Но это противоречит выполнению неравенств (4.4).

2) В множестве M присутствуют только индексы объектов исходящего подмножества O^- , и в результате обслуживания любого из оставшихся объектов типа $\lambda_i = 1$ резервуар переполнится, т.е. $V_1(t) - v_j < 0$ для каждого $j \in M$. Но это также противоречит выполнению неравенств (4.4).

3) В множестве M присутствуют как индексы входящего O^+ , так и исходящего O^- подмножеств объектов типа $\lambda_i = 1$. При этом в результате обслуживания любого из оставшихся объектов резервуар R_1 либо переполнится, либо обслуживание будет невозможно из-за отсутствия нужного количества продукта в резервуаре. С учетом истинности условий (4.3) в первом случае будет иметь место соотношение $V_1^* - V_1(t) < v_j \leq V_1^*/2$, $j \in M$ и, следовательно, $V_1(t) > V_1^*/2$. А во втором случае — соотношение $V_1(t) < v_j \leq V_1^*/2$, $j \in M$ и, следовательно, $V_1(t) < V_1^*/2$. Получаем противоречие.

Рассуждения, изложенные в пунктах 1–3, можно также провести по отношению к объектам типа $\lambda_i = 2$ и соответствующему резервуару R_2 .

Таким образом, предположение о пустоте множества допустимых решений при соблюдении условий (4.4) ложно. Теорема доказана. \square

§ 4.2. Постановки оптимизационных задач

Считаем, что между введенными в п. 2.1.2 временными характеристиками $t^*(i_k, S)$ и $\bar{t}(i_{k-1}, S)$ в рассматриваемой модели также имеют место

соотношения

$$t^*(i_1, S) = 0; \bar{t}^*(i_k, S) = \bar{t}(i_{k-1}, S), k = \overline{2, n};$$

$$\bar{t}(i_k, S) = t^*(i_1, S) + \tau_{i_k}, k = \overline{1, n}.$$

Формализация подхода парето-оптимальности для рассматриваемой в этом параграфе модели обслуживания приводит к следующим бикритериальным задачам.

Задача Flow2R(\sum, \max). Найти полную совокупность парето-оптимальных оценок в проблеме минимизации суммы индивидуальных штрафов $\varphi_i(t)$ по всем объектам входящего подпотока \mathcal{O}^+ и величины максимального из индивидуальных штрафов $\psi_i(t)$ по всем объектам исходящего подпотока \mathcal{O}^- :

$$\{\min_{S \in \Omega} (\sum_{k: o_{i_k} \in \mathcal{O}^+} \varphi_{i_k}(\bar{t}(i_k, S))), \min_{S \in \Omega} (\max_{l: o_{i_l} \in \mathcal{O}^-} \psi_{i_l}(\bar{t}(i_l, S)))\}.$$

Задача Flow2R(\sum, \sum). Найти полную совокупность парето-оптимальных оценок в проблеме минимизации суммы индивидуальных штрафов $\varphi_i(t)$ по всем объектам подпотока \mathcal{O}^+ и суммы индивидуальных штрафов $\psi_i(t)$ по всем объектам подпотока \mathcal{O}^- :

$$\{\min_{S \in \Omega} (\sum_{k: o_{i_k} \in \mathcal{O}^+} \varphi_{i_k}(\bar{t}(i_k, S))), \min_{S \in \Omega} (\sum_{l: o_{i_l} \in \mathcal{O}^-} \psi_{i_l}(\bar{t}(i_l, S)))\}.$$

Задача Flow2R(\max, \max). Найти полную совокупность парето-оптимальных оценок в проблеме минимизации величины максимального из индивидуальных штрафов $\varphi_i(t)$ по всем объектам подпотока \mathcal{O}^+ и величины максимального из индивидуальных штрафов $\psi_i(t)$ по всем объектам подпотока \mathcal{O}^- :

$$\{\min_{S \in \Omega} (\max_{k: o_{i_k} \in \mathcal{O}^+} \varphi_{i_k}(\bar{t}(i_k, S))), \min_{S \in \Omega} (\max_{l: o_{i_l} \in \mathcal{O}^-} \psi_{i_l}(\bar{t}(i_l, S)))\}.$$

§ 4.3. Решающие соотношения динамического программирования

Выполним решение задач Flow2R(\sum, \max), Flow2R(\sum, \sum) и Flow2R(\max, \max) алгоритмами, основанными на идеологии динамическо-

го программирования в его бикритериальном расширении.

Для этого воспользуемся введенными в п. 3.1.3 двуместной операцией \otimes объединения вектора и множества векторов и операцией $\text{eff}(M)$ выделения максимального по включению подмножества недоминируемых по Парето двумерных векторов в множестве M . Напомним, что для каждой задачи операция \otimes определяется специфическим образом, но она позволяет записать решающие соотношения динамического программирования в одинаковом виде для задач $\text{Flow2R}(\sum, \max)$, $\text{Flow2R}(\max, \max)$ и $\text{Flow2R}(\max, \max)$. В таблице 3.1 приведены определения операции объединения.

Алгоритм решения задач

При обслуживании объектов потока \mathcal{O}_n решения принимаются в те моменты времени, когда процессор свободен и необходимо выбрать следующий для обслуживания объект из входящего или исходящего подпотока; при этом необходимо учитывать текущее заполнение обоих резервуаров. Соответственно, четверка значений $(t, Q, V_1(t), V_2(t))$ определяет текущее состояние системы в момент t принятия решения, где Q — множество индексов объектов потока \mathcal{O}_n , ожидающих обслуживания в этот момент времени t .

Для записи рекуррентных соотношений динамического программирования воспользуемся также введенными в п. 3.2.3 обозначениями: $F(t)$ — подмножество индексов объектов потока \mathcal{O}_n , поступающих в систему обслуживания в момент t ; $D(t, \delta)$ — совокупность индексов объектов, поступающих в очередь на интервале времени $[t, t + \delta]$, $\delta \geq 1$.

Для любого состояния системы обслуживания $(t, Q, V_1(t), V_2(t))$ при $t < t_n$ множество Q считается непустым, что обеспечивается включением фиктивного объекта с параметрами $t_0 = t$, $\tau_0 = \min\{\delta \mid D(t, \delta) \neq \emptyset\}$, $d_0 = \infty$, $v_0 = 0$, а функции индивидуального штрафа $\varphi_0(t)$ и $\psi_0(t)$ при этом тождественно равны нулю. Для определенности будем считать, что фик-

тивный объект включен в подпоток \mathcal{O}^+ и, следовательно, $w_0 = 1$. Выбор фиктивного объекта на обслуживание означает простой процессора, начиная от момента времени t до момента поступления в очередь какого-либо нового объекта потока \mathcal{O}_n . Фиктивный объект исключается из множества Q при $t \geq t_n$, т.е. когда все объекты потока \mathcal{O}_n поступили в систему и ожидают своего обслуживания.

Пусть $E(t, Q, V_1(t), V_2(t))$ обозначает совокупность эффективных по Парето двумерных оценок, сгенерированную за период времени от момента t и до окончания процесса обслуживания всех объектов потока \mathcal{O}_n . Тогда полная совокупность эффективных по Парето оценок исходной задачи будет представлять собой множество $E(0, F(0), V_1(0), V_2(0))$.

В текущем состоянии $(t, Q, V_1(t), V_2(t))$ системы обслуживания объект с индексом $\alpha \in Q$ может быть допущен к обслуживанию только при выполнении одного из объёмных ограничений (4.2). Пусть Q^* — множество индексов объектов, допустимых к обслуживанию в состоянии $(t, Q, V_1(t), V_2(t))$, $Q^* \subseteq Q$. Очевидно, что для любого $\theta \geq 0$ и $Q = \{\alpha\}$, где α — индекс произвольного объекта, имеет место соотношение

$$E(t_n + \theta, \{\alpha\}, V_1(t), V_2(t)) = \{\varphi_\alpha(t_n + \theta + \tau_\alpha), \psi_\alpha(t_n + \theta + \tau_\alpha)\}. \quad (4.5)$$

Если в состоянии $(t, Q, V_1(t), V_2(t))$ в качестве очередного обслуживаемого объекта выбран объект с индексом $\alpha \in Q^*$, то следующим моментом принятия решения будет $t + \tau_\alpha$; множество индексов объектов, ожидающих обслуживания в этот момент времени, определяется как совокупность $(Q \setminus \{\alpha\}) \cup D(t, \tau_\alpha)$. В результате обслуживания объекта произойдет изменение заполнения соответствующего компонента на величину $w_\alpha \cdot v_\alpha$. С учетом введенной ранее операции (4.1) значения заполнения компонентов будут равны $V_1(t) + cv(1, \alpha)$ и $V_2(t) + cv(2, \alpha)$ соответственно. Суммируя

приведенные рассуждения, запишем формулу

$$E(t, Q, V_1(t), V_2(t)) = \text{eff}\left\{ \bigcup_{\alpha \in Q^*} [(\varphi(t + \tau_\alpha), \psi(t + \tau_\alpha)) \otimes \right. \quad (4.6)$$

$$\left. E(t + \tau_\alpha, (Q \setminus \{\alpha\}) \cup D(t, \tau_\alpha), V_1(t) + \text{cv}(1, \alpha), V_2(t) + \text{cv}(2, \alpha)) \right\}.$$

Выражения (4.5) и (4.6) представляют собой рекуррентные соотношения динамического программирования для решения бикритериальной задачи $\text{Flow2R}(\sum, \max)$. Процесс решения задачи на основе построенных соотношений аналогичен описанному в п. 3.1.3.

Оценки трудоемкости алгоритмов

Дадим верхнюю оценку числа элементарных операций, выполняемых при вычислениях по соотношениям (4.5) и (4.6) в процессе решения задачи $\text{Flow2R}(\sum, \max)$. Процедура решения задачи $\text{Flow2R}(\sum, \max)$ предусматривает последовательное определение множеств $E(t, Q, V_1(t), V_2(t))$. В рассматриваемых состояниях $(t, Q, V_1(t), V_2(t))$ число значений первого аргумента t сверху ограничено величиной $L = t_n + \sum_{i=1}^n \tau_i$, а Q является произвольным подмножеством индексов из числа прибывших на момент времени t объектов с учетом наличия фиктивного объекта. Таким образом, число состояний системы не превышает $L \cdot 2^{n+1}$. Значения $V_1(t)$ и $V_2(t)$ в состоянии $(t, Q, V_1(t), V_2(t))$ используются для выбора из множества Q подмножества допустимых в данном состоянии объектов Q^* ; эта операция мажорируется линейной функцией от n . Максимальное из возможных значений вторых координат векторов-оценок в множествах $E(t, Q, V_1(t), V_2(t))$ не может превышать $\max_{i: o_i \in \mathcal{O}^-} \psi_i(L)$. Поэтому число оценок в каждом из множеств $E(t, Q, V_1(t), V_2(t))$ при всех возможных значениях аргументов не может превышать $\max_{i: o_i \in \mathcal{O}^-} \psi_i(L) + 1$. Будем считать, что в каждом из найденных в процессе решения задачи множеств $E(t, Q, V_1(t), V_2(t))$ векторы упорядочены по убыванию значений второй координаты. Тогда общее число элементарных операций, выполняемых при определении каждого очередного множества $E(t, Q, V_1(t), V_2(t))$, оценивает-

ся величиной $O(\max_{i:o_i \in \mathcal{O}^-} \psi_i(L))$. Число элементарных операций, выполняемых при отыскании полной совокупности эффективных оценок, характеризуется величиной $O(L2^{n+1}n \max_{i:o_i \in \mathcal{O}^-} \psi_i(L))$.

Для решения задачи $\text{Flow2R}(\max, \max)$ верхняя оценка числа реализуемых алгоритмом элементарных операций такая же, как при решении задачи $\text{Flow2R}(\sum, \max)$.

Дадим верхнюю оценку числа элементарных операций, выполняемых при вычислениях по соотношениям (4.5) и (4.6) в процессе решения задачи $\text{Flow2R}(\sum, \sum)$. С учетом того, что максимальное из возможных значений вторых координат векторов-оценок в множествах $E(t, Q, V_1(t), V_2(t))$ не может превышать $n \cdot \max_{i:o_i \in \mathcal{O}^-} \psi_i(L)$, число элементарных операций, выполняемых при отыскании полной совокупности эффективных оценок, характеризуется величиной $O(L2^n n^2 \max_{i:o_i \in \mathcal{O}^-} \psi_i(L))$.

Как видно из описанной п. 3.1.3 процедуры, алгоритм определения по принадлежащей множеству $E(0, F(0), V_1(0), V_2(0))$ векторной оценке соответствующей парето-оптимальной стратегии имеет линейно зависящую от n оценку числа выполняемых элементарных операций.

Таким образом, полученные оценки трудоемкости процедур синтеза полных совокупностей эффективных оценок для задач $\text{Flow2R}(\sum, \max)$, $\text{Flow2R}(\sum, \sum)$ и $\text{Flow2R}(\max, \max)$ экспоненциальны.

§ 4.4. Пример синтеза стратегии обслуживания

Для иллюстрации технологии расчета по описанному алгоритму динамического программирования рассмотрим решение задачи $\text{Flow2R}(\sum, \max)$ на следующем примере.

Пример 4.1. Требуется найти полную совокупность эффективных по Парето оценок и соответствующие этим оценкам парето-оптимальные стратегии обслуживания потока объектов \mathcal{O}_6 . Накопительно-расходные компоненты характеризуются значениями $V_1^* = 50$, $V_1(0) = 38$, $V_2^* = 50$,

$V_2(0) = 31$. Функции индивидуального штрафа определяются выражениями $\varphi_i(t) = a_i \cdot (t - \tau_i - t_i)$ и $\psi_i(t) = \max(t - d_i, 0)$, $i = \overline{1, 6}$, где a_i — целое число, штраф за единицу времени простоя в ожидании обслуживания объекта, $i = \overline{1, 6}$. Значения параметров модели представлены в таблице 4.1.

Таблица 4.1

Исходные данные

i	t_i	τ_i	a_i	d_i	v_i	w_i	λ_i
1	0	2	8	2	7	-1	1
2	2	3	4	7	14	+1	1
3	3	10	3	16	8	-1	2
4	10	13	2	24	4	-1	2
5	17	8	7	26	9	-1	1
6	21	14	7	35	15	+1	2

По ходу решения будем составлять список переходов, в который вносятся записи описанного в п. 3.1.3 типа (таблица 4.2).

Полная совокупность эффективных оценок в этой задаче будет представлять собой $E(0, \{0, 1\}, 38, 31)$. В множество Q также включен индекс фиктивного объекта 0. Присвоим состоянию $(0, \{0, 1\}, 38, 31)$ номер 0. По формуле (4.6) получаем $E(0, \{0, 1\}, 38, 31) = \text{eff}\{(0, 0) \otimes E(2, \{0, 1, 2\}, 38, 31), (0, 0) \otimes E(2, \{0, 2\}, 31, 31)\}$.

Таблица 4.2

Список переходов

№	$((m^1, m^2), \alpha, z.e, z^*.e^*)$	№	$((m^1, m^2), \alpha, z.e, z^*.e^*)$
1	$((217, 30), 5, 13.1, \varnothing.\varnothing)$	30	$((210, 29), 5, 29.1, \varnothing.\varnothing)$
2	$((66, 32), 4, 14.1, \varnothing.\varnothing)$	31	$((252, 20), 5, 25.1, 28.1)$
3	$((267, 30), 4, 12.1, 13.1)$	32	$((126, 39), 3, 30.1, \varnothing.\varnothing)$
4	$((192, 32), 5, 12.2, 14.1)$	33	$((224, 25), 3, 26.1, 28.1)$
5	$((267, 30), 6, 10.1, 12.1)$	34	$((196, 39), 6, 26.2, 30.1)$
6	$((192, 32), 6, 10.2, 12.2)$	35	$((312, 31), 3, 27.1, 29.1)$
7	$((119, 17), 6, 15.1, \varnothing.\varnothing)$	36	$((266, 39), 5, 27.2, 30.1)$
8	$((58, 28), 4, 16.1, \varnothing.\varnothing)$	37	$((312, 20), 3, 24.1, 25.1)$
9	$((149, 17), 4, 11.1, 15.1)$	38	$((266, 25), 5, 24.2, 26.1)$
10	$((86, 28), 6, 11.2, 16.1)$	39	$((238, 39), 5, 24.3, 26.2)$

№	$((m^1, m^2), \alpha, z.e, z^*.e^*)$	№	$((m^1, m^2), \alpha, z.e, z^*.e^*)$
11	$((149, 17), 5, 8.1, 11.1)$	40	$((312, 20), 3, 22.1, 24.1)$
12	$((86, 28), 5, 8.2, 11.2)$	41	$((266, 25), 5, 22.2, 24.2)$
13	$((112, 16), 6, 17.1, \emptyset.\emptyset)$	42	$((238, 39), 5, 22.3, 24.3)$
14	$((182, 25), 5, 18.1, \emptyset.\emptyset)$	43	$((105, 15), 6, 32.1, \emptyset.\emptyset)$
15	$((196, 16), 5, 9.1, 17.1)$	44	$((175, 24), 5, 33.1, \emptyset.\emptyset)$
16	$((149, 17), 0, 6.1, 8.1)$	45	$((182, 15), 5, 31.1, 32.1)$
17	$((86, 28), 0, 6.2, 8.2)$	46	$((149, 17), 0, 23.1, 8.1)$
18	$((208, 16), 4, 6.3, 9.1)$	47	$((86, 28), 0, 23.2, 8.2)$
19	$((184, 44), 2, 21.1, \emptyset.\emptyset)$	48	$((192, 15), 4, 23.3, 31.1)$
20	$((240, 30), 2, 19.1, 17.1)$	49	$((155, 17), 3, 4.1, 23.1)$
21	$((334, 36), 2, 20.1, 18.1)$	50	$((92, 28), 3, 4.2, 23.2)$
22	$((292, 22), 2, 7.1, 9.1)$	51	$((198, 15), 3, 4.3, 23.3)$
23	$((193, 17), 2, 5.1, 6.1)$	52	$((155, 17), 2, 2.1, 4.1)$
24	$((130, 28), 2, 5.2, 6.2)$	53	$((92, 28), 2, 2.2, 4.2)$
25	$((252, 16), 2, 5.3, 6.3)$	54	$((198, 15), 2, 2.3, 4.3)$
26	$((193, 17), 2, 3.1, 5.1)$	55	$((155, 17), 1, 0.1, 2.1)$
26	$((130, 28), 2, 3.2, 5.2)$	56	$((92, 28), 1, 0.2, 2.2)$
28	$((252, 16), 2, 3.3, 5.3)$	57	$((198, 15), 1, 0.3, 2.3)$
29	$((140, 20), 6, 28.1, \emptyset.\emptyset)$		

Присвоим состояниям $(2, \{0, 1, 2\}, 38, 31)$ и $(2, \{0, 2\}, 31, 31)$ номера 1 и 2 соответственно. По ходу решения каждое новое состояние системы будет получать следующий порядковый номер. В таблице 4.3 приведены все рассмотренные во время решения примера состояния.

Далее по формулам (4.5) и (4.6) найдем рекурсивно множества двумерных оценок $E(2, \{0, 1, 2\}, 38, 31)$ и $E(2, \{0, 2\}, 31, 31)$.

Для вычисления $E(2, \{0, 1, 2\}, 38, 31)$ нет смысла в выборе фиктивного объекта, потому что обслуживание объекта с индексом 1 может завершиться до поступления объекта с индексом 3. Нецелесообразно также обслуживать объект с индексом 1, так как его обслуживание можно начать раньше момента времени $t = 2$, а при попытке обслужить объект с индексом 2 резервуар R_1 переполнится до уровня 52. Следовательно, множество двумерных оценок $E(2, \{0, 1, 2\}, 38, 31)$ пусто.

Перейдем к нахождению $E(2, \{0, 2\}, 31, 31)$.

По формуле (4.6) получим $E(2, \{0, 2\}, 31, 31) = \text{eff}[(0, 0) \otimes E(3, \{0, 2, 3\}, 31, 31); (0, 0) \otimes E(5, \{0, 3\}, 45, 31)]$. В состоянии $(3, \{0, 2, 3\}, 31, 31)$ нет смысла обслуживать фиктивный объект, так как обслуживание объекта с индексом 2 может завершиться раньше, чем прибудет объект с индексом 4. Следовательно, $E(3, \{0, 2, 3\}, 31, 31) = \text{eff}[(44, 9) \otimes E(16, \{0, 4\}, 45, 23); (6, 2) \otimes E(26, \{2, 5, 6\}, 31, 19)]$.

Действуя аналогично и заполняя соответствующие списки, в итоге находим $E(3, \{0, 2, 3\}, 31, 31) = \{(193, 17), (130, 28), (252, 16)\}$. Продолжая рассуждения, найдем полную совокупность эффективных оценок $E(0, \{0, 1\}, 38, 31) = \{(155, 17), (92, 28), (198, 15)\}$.

Таблица 4.3

Список состояний

№	$(t, Q, V_1(t), V_2(t))$	№	$(t, Q, V_1(t), V_2(t))$
0	$(0, \{0, 1\}, 38, 31)$	17	$(37, \{6\}, 36, 19)$
1	$(2, \{0, 1, 2\}, 38, 31)$	18	$(43, \{5\}, 45, 34)$
2	$(2, \{0, 2\}, 31, 31)$	19	$(34, \{2, 6\}, 22, 19)$
3	$(3, \{0, 2, 3\}, 31, 31)$	20	$(40, \{2, 5\}, 31, 34)$
4	$(5, \{0, 3\}, 45, 31)$	21	$(48, \{2\}, 22, 34)$
5	$(13, \{0, 2, 4\}, 31, 23)$	22	$(10, \{0, 3, 4\}, 45, 31)$
6	$(16, \{0, 4\}, 45, 23)$	23	$(15, \{0, 4\}, 45, 23)$
7	$(26, \{2, 5, 6\}, 31, 19)$	24	$(23, \{3, 5, 6\}, 45, 27)$
8	$(17, \{0, 4, 5\}, 45, 23)$	25	$(33, \{5, 6\}, 45, 19)$
9	$(29, \{5, 6\}, 45, 19)$	26	$(31, \{3, 6\}, 36, 27)$
10	$(21, \{4, 5, 6\}, 45, 23)$	27	$(37, \{3, 5\}, 45, 42)$
11	$(25, \{4, 6\}, 36, 23)$	28	$(41, \{6\}, 36, 19)$
12	$(35, \{4, 5\}, 45, 38)$	29	$(47, \{5\}, 45, 34)$
13	$(48, \{5\}, 45, 34)$	30	$(45, \{3\}, 36, 42)$
14	$(43, \{4\}, 36, 38)$	31	$(28, \{5, 6\}, 45, 19)$
15	$(38, \{6\}, 36, 19)$	32	$(36, \{6\}, 36, 19)$
16	$(39, \{4\}, 36, 38)$	33	$(42, \{5\}, 45, 34)$

Пусть из этой совокупности ЛПР выбрана оценка $(92, 28)$. Построим соответствующую стратегию обслуживания. Заметим, что оценка $(92, 28)$ находится в записи номер 56 списка переходов. Второй элемент записи равен 1, поэтому, объект с индексом 1 будет обслуживаться первым. Следу-

ющим состоянием будет состояние с номером 2.2. Находим в списке переходов запись $((92, 28), 2, 2.2, 4.2)$, второй элемент которой равен 2. Следовательно, следующим будет обслужен объект с индексом 2. Текущая оценка $(92, 28)$ в состоянии с номером 2.2 получена из состояния с номером 4.2. Соответствующая запись в списке переходов будет $((92, 28), 3, 4.2, 23.2)$, и третьим будет обслуживаться объект с индексом 3. Находим далее в списке переходов запись $((86, 28), 0, 23.2, 8.2)$. Очередным будет обслуживаться объект с индексом 0, то есть процессор будет простаивать. Следующая запись соответствует состоянию с номером 8.2 — $((86, 28), 5, 8.2, 11.2)$. Следовательно, следующим будет обслужен объект с индексом 5. Следующая запись с номером 11.2 в списке переходов соответствует состоянию $((86, 28), 6, 11.2, 16.1)$ и объект с индексом 6 будет обслужен следующим. Оценка $(86, 28)$ в состоянии с номером 11.2 получена из состояния номер 8 списка переходов, которое является финальным. Следовательно, последним обслуживается объект с индексом 4. В итоге получили стратегию обслуживания объектов вида $\{1, 2, 3, 5, 6, 4\}$. Аналогично можно найти парето-оптимальные стратегии обслуживания $\{1, 2, 3, 5, 4, 6\}$ и $\{1, 2, 3, 4, 5, 6\}$, характеризующиеся оценками $(155, 17)$ и $(198, 15)$ соответственно.

§ 4.5. Алгоритм ветвей и границ

Для задач $\text{Flow2R}(\sum, \max)$, $\text{Flow2R}(\sum, \sum)$ и $\text{Flow2R}(\max, \max)$ разработан алгоритм на основе метода ветвей и границ, который отличается от описанного в § 3.3 только способом учёта объёмных ограничений. Рассмотрим ниже эту особенность подробнее.

Если $P(N) = \{p_1, p_2, \dots, p_{r(N)}\}$ — стратегия обслуживания объектов, которая фиксируется траекторией из корня дерева решений в вершину N , тогда величина заполнения ёмкости каждого из резервуаров R_1 и R_2 в вер-

шине N вычисляется по формулам

$$V_1(N) = V_1(0) + \sum_{j=1}^{r(N)} cv(1, j),$$

$$V_2(N) = V_2(0) + \sum_{j=1}^{r(N)} cv(2, j).$$

При ветвлении из вершины N ребро для объекта с индексом α , $o_\alpha \in \mathcal{O}^+$, $\lambda_\alpha = 1$ строится только в том случае, если $V_1(N) + v_\alpha \leq V_1^*$, а для объекта с индексом β , $o_\beta \in \mathcal{O}^-$, $\lambda_\beta = 1$ ребро строится, если $V_1(N) \geq v_\beta$. Аналогично ребро для объекта с индексом α , $o_\alpha \in \mathcal{O}^+$, $\lambda_\alpha = 2$ строится только в том случае, если $V_2(N) + v_\alpha \leq V_2^*$, для объекта с индексом β , $o_\beta \in \mathcal{O}^-$, $\lambda_\beta = 2$ ребро строится только если $V_2(N) \geq v_\beta$.

Остальные построения алгоритма ветвей и границ для решения задач $\text{Flow2R}(\sum, \max)$, $\text{Flow2R}(\sum, \sum)$ и $\text{Flow2R}(\max, \max)$ аналогичны описанным в § 3.3.

§ 4.6. Результаты вычислительных экспериментов

С целью оценки эффективности алгоритма ветвей и границ BnB и сравнения его с методом динамического программирования DP была проведена серия вычислительных экспериментов в условиях, описанных в приложении А.

Эксперименты для алгоритма BnB выполнялись на тех же наборах исходных данных, что и для алгоритма DP. В данном пункте представлены результаты экспериментов на примере решения задачи $\text{Flow2R}(\sum, \max)$; для задач $\text{Flow2R}(\sum, \sum)$ и $\text{Flow2R}(\max, \max)$ результаты имеют тот же порядок. В таблице 4.4 представлены характеристики каждого алгоритма.

Как видно из результатов вычислительных экспериментов, алгоритм BnB в среднем является более эффективным по сравнению с алгоритмом DP при любом значении величины размерности потока n . Но при этом не исключены ситуации, в которых алгоритм ветвей и границ может вырождаться в полный перебор. Это подтверждается результатами экспериментов; так

Таблица 4.4

Зависимости длительностей отработки алгоритмов решения задачи $\text{Flow2R}(\sum, \max)$ от размерности потока объектов \mathcal{O}_n , с

n	DP			BnB		
	t_{avg}	t_{min}	t_{max}	t_{avg}	t_{min}	t_{max}
8	0.001	0.000	0.016	0.000	0.000	0.016
9	0.005	0.000	0.125	0.001	0.000	0.031
10	0.026	0.000	0.375	0.006	0.000	0.297
11	0.092	0.000	1.828	0.019	0.000	0.828
12	0.961	0.000	27.094	0.121	0.000	8.172
13	4.301	0.000	115.747	0.538	0.000	93.572
14	27.410	0.000	920.145	2.458	0.000	2198.059
15	446.165	0.016	3596.678	38.504	0.000	8174.576
18	-	-	-	548.314	0.168	19845.584

при обслуживании 14 объектов максимальное время решения задачи алгоритмом BnB составляет примерно полчаса, в то же время алгоритмом DP — 15 минут.

Разработанные алгоритмы позволяют выполнить синтез парето-оптимальной совокупности стратегий обслуживания потока размерностью до 18 единиц за промежуток времени от нескольких секунд до 10 минут в среднем.

Выводы по главе 4

1) Построена бикритериальная модель обслуживания потока объектов стационарным процессором с двумя накопительно-расходными компонентами.

2) Поставлены бикритериальные задачи синтеза оптимальных стратегий обслуживания, различающиеся типами оптимизируемых критериев.

3) Сконструированы решающие процедуры для задач п. 2, основанные на идеологии динамического программирования и метода ветвей и границ, получены оценки их трудоёмкости, приведены результаты массовых вычислительных экспериментов.

Глава 5. Архитектура программных средств синтеза оптимально-компромиссных стратегий обслуживания потока объектов

Описана архитектура разработанных программных средств, предназначенных для синтеза стратегий управления обслуживанием бинарного потока объектов на одном стационарном процессоре с одним или двумя накопительно-расходными компонентами. Описываются назначение и функциональные возможности программных средств, излагаются особенности реализации алгоритмов, направленные на повышение их быстродействия.

§ 5.1. Назначение и функциональные возможности

Конечной стадией разработки любого алгоритма является его программная реализация с целью применения в автоматизированных системах принятия решений.

В основу математического обеспечения разработанного программного модуля [68] легли рассмотренные в работе однокритериальные и бикритериальные задачи, а также сконструированные решающие алгоритмы. Его назначение — предоставить диспетчеру математически обоснованные варианты стратегий управления обслуживанием множеств и потоков объектов с учетом складывающейся оперативной обстановки.

Программный комплекс реализован в среде Microsoft Visual Studio 2005 на языке программирования C++ и имеет следующие функциональные возможности, реализованные на данном этапе через командную строку.

- 1) Выбор модели управления обслуживанием.
- 2) Редактирование характеристик обслуживаемых объектов и накопительно-расходного компонента процессора.
- 3) Установка критериев оценки качества обслуживания и их количества: поддерживаются однокритериальные и бикритериальные задачи.
- 4) Синтез оптимальной стратегии обслуживания и соответствующей

ей стратегии для случая однокритериальной задачи. Синтез полной совокупности эффективных по Парето оценок или квазипаретовского множества с возможностью синтеза по выбранной оценке порождающей её стратегии обслуживания объектов в случае бикритериальной задачи.

Каждый изучаемый алгоритм был реализован в виде отдельного класса. Дополнительно разработан программный модуль, позволяющий случайным образом генерировать параметры модели в зависимости от решаемой задачи.

§ 5.2. Описание архитектуры программы

Одной из задач, решаемой при разработке программного комплекса, является построение его достаточно рациональной архитектуры. Применительно к рассматриваемому комплексу это означает что, с одной стороны, должны быть реализованы все описанные выше функциональные возможности, а с другой — предусмотрена возможность расширения программного модуля в случае необходимости внедрения в него модифицированных моделей обслуживания, других алгоритмов синтеза, а также дополнительных сервисных инструментов.

Все рассмотренные в работе модели являются различными с точки зрения описания и построения алгоритмов решения соответствующих задач. В рамках объектно-ориентированного подхода каждая из них реализована в виде отдельного класса, содержащего соответствующие члены и методы:

- класс `Problem_1cr_1r_res_set` содержит описание решателя DP однокритериальных задач обслуживания множества объектов процессором с накопительно-расходным компонентом;

- класс `Problem_1cr_1r_res` содержит описание решателя DP однокритериальных задач обслуживания потока объектов процессором с накопительно-расходным компонентом;

— класс `Problem_2cr_1r_res_set` содержит описание решателя DP бикритериальных задач в рамках модели обслуживания множества объектов процессором с накопительно-расходным компонентом;

— класс `Problem_2cr_1r_res` содержит описание решателя DP бикритериальных задач в рамках модели обслуживания потока объектов процессором с накопительно-расходным компонентом;

— класс `Problem_2cr_2r_res` содержит описание решателя DP бикритериальных задач в рамках модели обслуживания потока объектов процессором с двумя накопительно-расходными компонентами.

Каждый из классов в зависимости от описываемой модели содержит функции, необходимые для определения множества допустимых стратегий управления обслуживанием.

Структура `initial_data` описывает характеристики объектов, подлежащих обслуживанию. Отдельные классы соответствуют решателям, использующим концепции ветвей и границ и различные метаэвристические подходы.

§ 5.3. Замечания о повышении эффективности программной реализации

Основная цель данного параграфа состоит в описании некоторых подходов к повышению быстродействия программ, реализующих ресурсоёмкие вычислительные алгоритмы, за счет оптимизации программного кода.

Вопросы оптимизации программы всегда имеют индивидуальный характер и требуют творческого подхода. Тем не менее, существует инструментарий для достижения этих целей. В качестве модельного алгоритма для оптимизации выбран алгоритм DP решения задачи $\text{Flow1R}(\sum)$ (см. § 2.2) и именно он был реализован в первой версии программы. Результаты экспериментов этой реализации показали, что при решении задачи обслуживания 11 объектов необходимо 50 минут компьютерного времени.

После первой реализации алгоритма было проведено её детальное исследование с использованием инструмента Time Profiler среды разработчика XCode. Данное инструментальное средство позволяет выявлять критические с точки зрения быстродействия участки кода (так называемые «узкие места») и, в некоторых случаях, предлагает возможные варианты устранения выявленных проблем.

По результатам профилирования были выявлены такие узкие места и в программу были внесены следующие изменения.

- Исключена работа со списком переходов для хранения оценок, по которому впоследствии генерировались соответствующие стратегии. Вместо него та часть стратегии, которая получена на данном этапе решения задачи, сохраняется в одной структуре вместе с полученной оценкой.

- Изменена структура данных для запоминания текущего состояния: отдельно запоминается список объектов, которые целесообразно обслуживать на данный момент, и список всех готовых к обслуживанию объектов в данный момент времени. Это позволило сократить количество необходимых вычислений.

- Вместо стратегии целиком или ее части теперь хранится номер перестановки в лексикографическом порядке, соответствующий стратегии обслуживания.

Проведенные вычислительные эксперименты на модифицированной реализации алгоритма показали, что время решения задачи существенно сократилось. Теперь для решения задачи обслуживания потока из 11 объектов потребовалось менее одной секунды, а задача обслуживания потока из 15 объектов стала решаться в среднем за 5 минут.

Выводы по главе 5

Для решения задач диспетчерского управления обслуживанием потока объектов процессором с накопительно-расходными компонентами раз-

работан программный модуль, архитектура и элементы интерфейса которого позволяют осуществлять расширение функциональных возможностей. Кратко освещен вопрос оптимизации программной реализации алгоритмов с использованием инструментальных средств профилирования.

Заключение

Основным результатом диссертационной работы является постановка новой научной проблемы, имеющей существенное значение для построения компьютерных систем поддержки оперативного управления обслуживанием бинарных потоков объектов стационарным процессором в системе с накопительно-расходными компонентами.

При решении указанной проблемы получены следующие научно-технические результаты.

1. Построены новые модели управления обслуживанием бинарных потоков объектов стационарным процессором с накопительно-расходными компонентами.

2. Сформулированы задачи оптимизации управления обслуживанием бинарных потоков объектов стационарным процессором с накопительно-расходными компонентами.

3. В рамках концепции Парето разработаны алгоритмы динамического программирования, а также ветвей и границ, позволяющие строить оптимальные стратегии обслуживания бинарных потоков объектов; получены оценки их трудоемкости.

4. Для практических значимых модификаций моделей обслуживания бинарных потоков объектов разработаны процедуры синтеза стратегий обслуживания, обладающие повышенными скоростными характеристиками.

5. Сконструированы основанные на метаэвристических концепциях мягких вычислений алгоритмы построения субоптимальных множеств оценок.

Вышеуказанные результаты использованы при разработке специализированных программных средства для систем поддержки оперативного диспетчерского управления однопроцессорным обслуживанием бинарными

потоками объектов транспортного типа.

Таким образом, итогом системного анализа технологических процессов типа ГРП и выполненных исследований является *разработка* семейства алгоритмов, *предназначенных* для решения задач синтеза стратегий обслуживания бинарных потоков объектов стационарным процессором.

Результаты диссертационной работы *отличаются от известных* учетом в моделях обслуживания накопительно-расходных компонентов, а также возможностью реализации комбинаций двух независимых критериев оптимизации по Парето для оценки стратегий обслуживания.

Применение разработанных в диссертационной работе математических моделей и решающих алгоритмов и созданных на их основе программных средств *позволяет*, в частности, повысить эффективность управления процессами обслуживания на терминальных комплексах внутреннего водного транспорта.

Литература

1. *Батищев, Д.И.* Методы оптимального проектирования [Текст] / Д.И. Батищев — М.: Радио и связь, 1984. — 248 с.
2. *Батищев, Д.И.* Многокритериальный выбор с учетом индивидуальных предпочтений [Текст] / Д.И. Батищев, Д.Е. Шапошников — Н.Новгород: Изд-во ИПФ РАН, 1994. — 92 с.
3. *Батищев, Д.И.* Применение генетических алгоритмов к решению задач дискретной оптимизации [Текст] : Учеб. пособие / Д.И. Батищев, Е.А. Неймарк, Н.В. Старостин — Н.Новгород: Изд-во ННГУ им. Н.И. Лобачевского, 2006. — 123 с.
4. *Батищев, Д.И.* Решение дискретных задач с помощью генетических алгоритмов [Текст] : учеб. пособие / Д.И. Батищев, В.Е. Костюков, Е.А. Неймарк [и др.]. — Н. Новгород: Изд-во ННГУ им. Н.И. Лобачевского, 2011. — 199 с.
5. *Беленький, А.С.* Применение моделей и методов теории расписаний в задачах оптимального планирования на грузовом транспорте [Текст] / А.С. Беленький, Е.В. Левнер // Автоматика и телемеханика. — 1989. — №2 — С. 3–77.
6. *Беленький, А.С.* Исследование операций в транспортных системах: идеи и схемы методов оптимизации планирования [Текст] / А.С. Беленький, Е.В. Левнер — М.: Мир, 1992. — 582 с.
7. *Беллман, Р.* Динамическое программирование [Текст] / Р. Беллман — М.: Изд-во иностр. лит., 1960. — 400 с.
8. *Беллман, Р.* Прикладные задачи динамического программирования [Текст] / Р. Беллман, С. Дрейфус. — М.: Наука, 1965. — 460 с.
9. *Бугаев, Ю.В.* Обобщение схемы динамического программирования [Текст] / Ю.В. Бугаев, С.В. Чикунов // Автоматика и телемеханика. — 2009. — №2 — С. 90–100.
10. *Бурков, В.Н.* Эвристический подход к решению динамических задач распределения ресурсов [Текст] / В.Н. Бурков, С.Е. Ловецкий // Автоматика и телемеханика. — 1966. — №5 — С. 89–90.
11. *Гермейер, Ю.Б.* Введение в теорию исследования операций [Текст] / Ю.Б. Гермейер — М.: Наука, 1971. — 384 с.
12. *Гимади, Э.Х.* Полиномиальная разрешимость задач календарного планирования со складываемыми ресурсами и директивными сроками [Текст] / Э.Х. Гимади, В.В. Залюбовский, С.В. Севастьянов // Дискретный анализ и исследование операций. — Январь–июнь 2000. — Серия 2, Т.7, №1. — С. 9–34.

13. *Гимади, Э.Х.* Новая версия асимптотически точного алгоритма решения евклидовой задачи коммивояжера [Текст] / Э.Х. Гимади // Труды XII Байкальской международной конференции. Методы оптимизации и их приложения. — Иркутск, 2001. — Т.1. — С. 117–124
14. *Гладков, Л.А.* Генетические алгоритмы [Текст] / Л.А. Гладков, В.В. Курейчик, В.М. Курейчик // 2-е изд., испр. и доп. / под ред. В.М. Курейчика. — М.: ФИЗМАТЛИТ, 2010. — 368 с.
15. *Гордон, В.С.* Минимизация стоимости, связанной с переменными директивными сроками, в задаче теории расписаний с одним прибором [Текст] / В.С. Гордон // Автоматика и телемеханика. — 1992. — № 2. — С. 105–112.
16. *Гордон, В.С.* О минимаксных задачах теории расписаний с одним прибором [Текст] / В.С. Гордон, В.С. Танаев // Изв. АН БССР. Сер. физ.-мат. наук. — №3. — 1982. — С. 3–9.
17. *Гэри, М.* Вычислительные машины и труднорешаемые задачи [Текст] / М. Гэри, Д. Джонсон. — М.: Мир, 1982. — 416 с.
18. *Джонс, М.Т.* Программирование искусственного интеллекта в приложениях [Текст] / М.Т. Джонс // пер. с англ. Осипов А.И. — М.: ДМК Пресс, 2006. — 312 с.
19. *Дуничкина, Н.А.* Об оценке приближенных решений бикритериальных задач дискретного программирования [Текст] / Н.А. Дуничкина, Ю.С. Федосенко // Информационные системы и технологии ИСТ-2009. Материалы XV Международной научно-технической конференции. — Н.Новгород : НГТУ, 2009. — С. 300–302.
20. *Емеличев, В.А.* Лексикографические оптимумы многокритериальной задачи дискретной оптимизации [Текст] / В.А. Емеличев, М.К. Кравцов, О.А. Янушкевич // Математические заметки. — 1995. — Т.58, вып.3. — С. 365–371.
21. *Игудин, Р.В.* Задачи теории расписаний на транспорте и алгоритмы их решения [Текст] / Р.В. Игудин // Экономика и математические методы. — 1975. — № 3. — С. 491–499.
22. *Карп, Р.М.* Сводимость комбинаторных проблем [Текст] / Р.М. Карп // Кибернетический сборник. — 1972. — Вып.12. — С. 16–38.
23. *Кнут, Д.* Искусство программирования, том 3. Сортировка и поиск [Текст] / Д. Кнут — 2-е изд. — М.: «Вильямс», 2007. — 824 с.
24. *Коган, Д.И.* Задача диспетчеризации: анализ вычислительной сложности и полиномиально разрешимые подклассы [Текст] / Д.И. Коган, Ю.С. Федосенко // Дискретная математика, 1996. — Т.8, вып.3. — С. 135–147.

25. Коган, Д.И. Моделирование и оптимизация управления обслуживанием потока объектов в системе с изодромным элементом [Текст] / Д.И. Коган, Ю.С. Федосенко, А.В. Шеянов // Межвузовский сб. науч. тр. — Н.Новгород : Издательство ВГАВТ, 1996. — Вып.273, Ч. 1. — С. 44–54.
26. Коган, Д.И. Проблема синтеза оптимального расписания обслуживания бинарного потока объектов mobile-процессором [Текст] / Д.И. Коган, Ю.С. Федосенко, А.В. Шеянов // Труды III Международной конференции «Дискретные модели в теории управляющих систем» — М.: Изд-во МГУ им. М.В. Ломоносова, 1998. — С. 43–46.
27. Коган, Д.И. Задача синтеза оптимального расписания обслуживания бинарного потока объектов в рабочей зоне mobile-процессора [Текст] / Д.И. Коган, Ю.С. Федосенко // Вестник Нижегородского университета. Математическое моделирование и оптимальное управление. — Н.Новгород, 1999. — Вып.1(20). — С. 179–187.
28. Коган, Д.И. Решение бикритериальной задачи о ранце методом ветвей и границ [Текст] / Д.И. Коган, А.Н. Федорин // Моделирование и оптимизация сложных систем: Межвузовский сб. научных тр. — Н.Новгород: ВГАВТ, 2004. — Вып.9. — С. 108–117.
29. Коган, Д.И. Динамическое программирование и дискретная многокритериальная оптимизация [Текст] : учеб. пособие / Д.И. Коган // Н.Новгород : Издательство ННГУ им. Н.И. Лобачевского, 2005. — 260 с.
30. Коган, Д.И. Математическая модель и алгоритм синтеза субоптимальных расписаний однопроцессорного обслуживания пространственно рассредоточенной группы стационарных объектов [Текст] / Д.И. Коган, Ю.С. Федосенко, А.Ю. Шлюгаев // Труды VI Международной конференции «Идентификация систем и задачи управления» SICPRO'07. — М., 2007. — С. 1026–1038.
31. Коган, Д.И. Задачи синтеза оптимальных стратегий обслуживания стационарных объектов в одномерной рабочей зоне процессора [Текст] / Д.И. Коган, Ю.С. Федосенко // Автоматика и телемеханика, 2010. — №10. — С. 50–62.
32. Коган, Д.И. О синтезе стратегий обслуживания группы стационарных объектов в одномерной рабочей зоне процессора при наличии двух оценочных критериев [Текст] / Д.И. Коган, Ю.С. Федосенко, Н.А. Дуничкина // Новые информационные технологии. Сборник трудов XIII Всероссийской научно-технической конференции (Москва, 19-21 апреля 2010 г.). — М.: МГУПИ, 2010. — С. 77–81.

33. Коган, Д.И. Бикритериальные задачи обслуживания mobile-процессором рассредоточенных в одномерной рабочей зоне объектов [Текст] / Д.И. Коган, Ю.С. Федосенко, Н.А. Дуничкина // Проблемы теоретической кибернетики. Материалы XVI Международной конференции. (Н.Новгород, 20–25 июня 2011 г.). — Нижний Новгород : Издательство Нижегородского госуниверситета, 2011. — С. 202–205.
34. Коган, Д.И. Бикритериальные модели и парето-оптимальные стратегии обслуживания группировки стационарных объектов [Текст] / Д.И. Коган, Ю.С. Федосенко, Н.А. Дуничкина // Труды IX Международной конференции «Идентификация систем и задачи управления – SICPRO' 12». Москва 30 января – 2 февраля 2012 г. Институт проблем управления им. В.А. Трапезникова РАН. — М.: Институт проблем управления им. В.А. Трапезникова РАН, 2012. — Вып.9. — С. 287–300.
35. Конвей, Р.В. Теория расписаний [Текст] / Р.В. Конвей, В.Л. Максвелл, Л.В. Миллер — ММ.: Наука, 1975. — 359 с.
36. Коновальчук, Е.В. Модели и методы оперативного управления проектами [Текст] / Е.В. Коновальчук, Д.А. Новиков. — М.: ИПУ РАН, 2004. — 63 с.
37. Корбут, А.А. Дискретное программирование [Текст] / А.А. Корбут, Ю.Ю. Финкельштейн // под. ред. Д.Б. Юдина. — М.: Наука, 1969. — 368 с.
38. Корбут, А.А. Метод ветвей и границ. Обзор теории, алгоритмов, программ и приложений [Текст] / А.А. Корбут, И.Х. Сигал, Ю.Ю. Финкельштейн // Math. Operation Forsch. Statist. Ser. Optimization. — 1977. — V.8, №2. — P. 253–280.
39. Кормен, Т. Алгоритмы: построение и анализ [Текст] / Т. Кормен, Ч. Лейзерсон, Р. Ривест, К. Штайн — М.: Вильямс, 2007. — 1296 с.
40. Куимова, А.С. О парето-оптимальных стратегиях однопроцессорного обслуживания потока объектов [Текст] / Н.А. Дуничкина, А.С. Куимова // Информационные системы и технологии ИСТ-2010. Материалы XVI Международной научно-технической конференции. — Н.Новгород : НГТУ, 2010. — С. 329–330.
41. Куимова, А.С. Развитие математической модели для компьютерной системы поддержки управления очередностью обработки речных судов [Текст] / А.С. Куимова, Д.В. Минаев // Материалы I Межвузовской научно-практической конференции студентов и аспирантов «Современные тенденции и перспективы развития водного транспорта России» 12–13 мая 2010 года. — СПб.: СПГУВК, 2010. — С. 195–197.

42. Куимова, А.С. Модель и алгоритм синтеза стратегий обслуживания потока объектов в системе с накопительным элементом при наличии двух оценочных критериев [Текст] / А.С. Куимова, Д.В. Минаев // Технологии Microsoft в теории и практике программирования. Материалы Всероссийской конференции молодых ученых / под редакцией проф. В.П. Гергеля. — Н.Новгород : Издательство Нижегородского государственного университета, 2010. — С. 241–243.
43. Куимова, А.С. Особенности распараллеливания алгоритма оптимизации в задаче однопроцессорного обслуживания потока объектов [Текст] / А.С. Куимова // Сборник трудов научного конгресса 12-го Международного научно-промышленного форума «Великие реки'2010». — Н.Новгород : ННГАСУ, 2010. — Т.2. — С. 149–151.
44. Куимова, А.С. Диспетчеризация однопроцессорного обслуживания потока объектов [Текст] / А.С. Куимова, Д.В. Минаев, Ю.С. Федосенко // Будущее технической науки: тезисы докладов IX Международной молодежной научно-технической конференции. — Н.Новгород : НГТУ, 2010. — С. 104–109.
45. Куимова, А.С. Задача синтеза стратегий обслуживания потока объектов в системе с накопительным компонентом [Текст] / А.С. Куимова, Д.В. Минаев, Ю.С. Федосенко // Проблемы теоретической кибернетики. Тезисы докладов XVI Международной конференции. г. Нижний Новгород, 20–25 июня 2011. — Н.Новгород : Издательство Нижегородского государственного университета, 2010. — С. 495–499.
46. Куимова, А.С. Параллельная реализация алгоритма синтеза оптимальной стратегии однопроцессорного обслуживания потока объектов в стандарте Open MP [Текст] / А.С. Куимова // 16-я Нижегородская сессия молодых ученых (математические науки), «Красный плес», 18–21 апреля 2010 г. — Н.Новгород : изд. Гладкова О.В., 2010. — С. 129–132.
47. Куимова, А.С. Синтез ограниченных по структуре оптимально-компромиссных стратегий обслуживания потока объектов [Текст] / А.С. Куимова, Ю.С. Федосенко // Вестник Национального технического университета «ХПИ». Сборник научных трудов. Тематический выпуск: Информатика и моделирование. — Харьков : НТУ «ХПИ», 2011. — №17. — С. 70–75.
48. Куимова, А.С. Опыт распараллеливания алгоритма динамического программирования для задачи синтеза стратегий обслуживания потока объектов [Текст] / А.С. Куимова // Высокопроизводительные параллельные вычисления на кластерных системах. Материалы XI Всероссийской конференции (Н.Новгород, 2–3 ноября 2011 г.) / под редакцией проф. В.П. Гергеля. — Н.Новгород : Издательство Нижегородского государственного университета, 2011. — С. 181–185.

49. Куимова, А.С. Модели и алгоритмы для компьютерной системы поддержки управления обработкой танкерного флота в условиях водных путей приполярного региона [Текст] / А.С. Куимова, Д.В. Минаев // Научно-техническая международная молодежная конференция «Системы, методы, техника и технологии обработки медиаконтента»: Сборник тезисов. — М.: МГУП им. Ивана Федорова, 2011. — С. 63.
50. Куимова, А.С. Управление однопроцессорным обслуживанием бинарного потока объектов в системе с накопительным компонентом [Текст] / А.С. Куимова, Д.В. Минаев, Ю.С. Федосенко // Журнал Информационно-измерительные и управляющие системы. — М. : Издательство «Радиотехника», 2011. — Т. 9, №3. — С. 33–37.
51. Куимова, А.С. Синтез стратегий обслуживания бинарного потока объектов стационарным процессором с накопительным элементом [Текст] / А.С. Куимова, Д.В. Минаев, Ю.С. Федосенко // Вестник АГТУ. Серия: Управление, вычислительная техника и информатика. — Астрахань : Издательство АГТУ, 2011. — №2. — С. 150–157.
52. Куимова, А.С. Управление однопроцессорным обслуживанием бинарного потока объектов в системе с накопительным компонентом [Текст] / А.С. Куимова, Д.В. Минаев, Ю.С. Федосенко // Журнал Информационно-измерительные и управляющие системы. — Москва : Издательство «Радиотехника», 2011. — т.9, №3. — С. 33–37.
53. Куимова, А.С. Синтез стратегий обслуживания бинарного потока объектов стационарным процессором с накопительным элементом [Текст] / А.С. Куимова, Д.В. Минаев, Ю.С. Федосенко // Вестник АГТУ. Серия: Управление, вычислительная техника и информатика. — Астрахань : Издательство АГТУ, 2011. — №2. — С. 150–157.
54. Куимова, А.С. Параллельные алгоритмы решения задачи диспетчеризации грузовой обработки бинарного судопотока в условиях Салехардского речного порта [Текст] / А.С. Куимова, Д.В. Минаев // Материалы Международной научно-практической конференции «Водный транспорт России: инновационный путь развития». 6-7 октября 2010 года. — СПб.: СПГУВК, 2011. — С. 104–109.
55. Куимова, А.С. О канонической задаче диспетчеризации с двумя критериями оценки [Текст] / А.С. Куимова // Материалы Международной научно-практической конференции «Водный транспорт России: инновационный путь развития». 6-7 октября 2010 года. — СПб.: СПГУВК, 2011. — С. 109–114.
56. Куимова, А.С. Синтез оптимально-компромиссных стратегий в однопроцессорной модели с ограниченной дисциплиной обслуживания потока объектов [Текст] / А.С. Куимова // Theoretical and Applied Aspects

- of Cybernetics. Proceedings of the International Scientific Conference of Students and Young Scientists. — Kyiv : Bukrek, 2011. — С. 333–335.
57. Куимова, А.С. Опыт реализации технологии CUDA в вычислительных задачах диспетчеризации [Текст] / А.С. Куимова, Д.В. Минаев, Ю.С. Федосенко // Информационные системы и технологии ИСТ–2011. Материалы XVII Международной научно-технической конференции. — Н.Новгород : НГТУ, 2011. — С. 339.
58. Куимова, А.С. Сравнительный анализ точного и приближенного алгоритма синтеза стратегий обслуживания в канонической модели диспетчеризации [Текст] / А.С. Куимова // Материалы II Межвузовской научно-практической конференции студентов и аспирантов «Современные тенденции и перспективы развития водного транспорта России» 12-13 мая 2011 года. — СПб.: СПГУВК, 2011. — С. 275–280.
59. Куимова, А.С. Реализация концепции параметризованной сложности в бикритериальной модели управления обслуживанием потока объектов [Текст] / А.С. Куимова // Вестник Нижегородского университета им. Н.И. Лобачевского. — Н.Новгород : Издательство Нижегородского госуниверситета, 2012. — №5, Ч.2. — С. 131–139.
60. Куимова, А.С. Об алгоритмах синтеза стратегий управления однопроцессорным обслуживанием потока объектов в системе с накопительным элементом [Текст] / А.С. Куимова // 17-ая Нижегородская сессия молодых ученых (технические науки), «Морозовский», 19–22 марта 2012 года. — Н.Новгород : НИУ РАНХиГС, 2012. — С. 93–96.
61. Куимова, А.С. О параметризации вычислительной сложности бикритериальной задачи управления обслуживанием потока объектов в системе с накопительным элементом [Текст] / А.С. Куимова, Ю.С. Федосенко // Информационные системы и технологии ИСТ–2012. Материалы XVIII Международной научно-технической конференции. — Н.Новгород : НГТУ, 2012. — С. 319.
62. Куимова, А.С. Модификация концепции d-расписаний для бикритериальной задачи обслуживания бинарного потока объектов в стационарной однопроцессорной системе с накопительным элементом [Текст] / А.С. Куимова, Ю.С. Федосенко // Труды 14-го Международного научно-промышленного форума «Великие реки'2012». Материалы научно-методической конференции профессорско-преподавательского состава, аспирантов, специалистов и студентов «Проблемы использования и инновационного развития внутренних водных путей в бассейнах великих рек». — Н.Новгород : Издательство ФБОУ ВПО «ВГАВТ», 2012. — Т.1. — С. 74–77.

63. Куимова, А.С. Модели сравнения двух множеств оценок для бикритериальных задач синтеза оптимизированных по Парето решений [Текст] / А.С. Куимова, Н.А. Дуничкина // Труды 14-го Международного научно-промышленного форума «Великие реки'2012». Материалы научно-методической конференции профессорско-преподавательского состава, аспирантов, специалистов и студентов «Проблемы использования и инновационного развития внутренних водных путей в бассейнах великих рек». — Н.Новгород : Издательство ФБОУ ВПО «ВГАВТ», 2012. — Т.1. — С. 71–74.
64. Куимова, А.С. Модель и алгоритмы синтеза стратегий обслуживания бинарного потока объектов при наличии двух критериев оценки [Текст] / А.С. Куимова // XI Белорусская математическая конференция: Тезисы докладов Международной научной конференции. Минск, 5–9 ноября 2012. — Минск : Институт математики НАН Беларуси, 2012. — Ч.4. — С. 91–92.
65. Kuimova, A.S. Bi-criteria optimization problem of binary objects flow servicing by stationary service processor with storage container [Текст] / A.S. Kuimova // 25th European Conf. on Operational Research, EURO 2012 : Conference Programme, Vilnius, Lithuania, 8–11 July, 2012. — Sci. Comm. : M. Christiansen [et.al.]. The Association of European Operational Research Societies, 2012. — P. 193.
66. Куимова, А.С. Алгоритмы решения задачи диспетчеризации грузовой обработки судопотока в условиях Салехардского речного порта [Текст] / А.С. Куимова // Сборник докладов 60-й Международной молодежной научно-технической конференции «МОЛОДЕЖЬ. НАУКА. ИННОВАЦИИ», 17–18 сентября 2012 г. — Владивосток : Мор. гос. ун-т, 2012. — Т. 1. — С. 133–138.
67. Куимова, А.С. Оптимизация оперативных планов обработки танкерного флота в условиях салехардского порта: модели и алгоритмы синтеза [Текст] / А.С. Куимова, Ю.С. Федосенко // «ІНФОРМАЦІЙНІ УПРАВЛЯЮЧІ СИСТЕМИ ТА ТЕХНОЛОГІЇ» (ІУСТ–ОДЕСА–2012). Тези доповідей. Матеріали Всеукраїнської наукової — практичної конференції, 17–18 жовтня 2012 р., Одеса / видп. ред. В.В. Вичужанін. — Суми : ТОВ «Друкарський дім «Паріус», 2012. — С. 124–126.
68. Куимова, А.С. Программа синтеза оптимально-компромиссных стратегий обслуживания потока объектов в задаче диспетчеризации [Текст] / А.С. Куимова // Свидетельство об официальной регистрации программ для ЭВМ № 2012661024. Зарегистрировано в реестре программ для ЭВМ Федеральной службы по интеллектуальной собственности, патентам и товарным знакам РФ от 5 декабря 2012.

69. *Лазарев, А.А.* Решение NP-трудной задачи теории расписаний минимизации суммарного запаздывания [Текст] / А.А. Лазарев // Журнал вычислительной математики и математической физики. — 2007. — Т.47. — №6. — С. 1087–1099.
70. *Лазарев, А.А.* Теория расписаний. Минимизация максимального временного смещения и суммарного взвешенного числа запаздывающих требований [Текст] / А.А. Лазарев, Р.Р. Садыков. — Вычислительный центр им. А.А. Дородницына РАН, М., 2007. — 180 с.
71. *Лотов, В.А.* Многокритериальные задачи принятия решений [Текст] / В.А. Лотов, И.И. Поспелова — М.: МАКС Пресс, 2008. — 197 с.
72. *Малышкин, А.Г.* Организация и планирование работы речного флота [Текст] / А.Г. Малышкин. — М.: Транспорт, 1985. — 215 с.
73. *Ногин, В.Д.* Принятие решений в многокритериальной среде: количественный подход [Текст] / В.Д. Ногин // 2-е изд., испр. и доп. — М.: Физматлит, 2004. — 176 с.
74. *Поддиновский, В.В.* Оптимизация по последовательно применяемым критериям [Текст] / В.В. Поддиновский, В.М. Гаврилов. — М.: Советское радио, 1975. — 192 с.
75. *Поддиновский, В.В.* Об относительной важности критериев в многокритериальных задачах принятия решений [Текст] / В.В. Поддиновский. // Многокритериальные задачи принятия решений — М.: Машиностроение, 1978. — С. 48–82.
76. *Поддиновский, В.В.* Принцип гарантированного результата для частичных отношений предпочтения [Текст] / В.В. Поддиновский. // Журнал вычислительной математики и математической физики, 1979. — №6. — С. 1437–1450.
77. *Поддиновский, В.В.* Парето-оптимальные решения многокритериальных задач [Текст] / В.В. Поддиновский, В.Д. Ногин — 2-е изд., испр. и доп. — М.: Физматлит, 2007. — 256 с.
78. *Подчасова, Т.П.* Эвристические методы календарного планирования [Текст] / Т.П. Подчасова, В.М. Португал, В.А. Татаров [и др.]. — Киев : Техніка, 1980. — 144 с.
79. *Прилуцкий, М.Х.* Детерминированная модель и алгоритм определения оптимальной очередности обработки судов с учетом времени их прибытия [Текст] / М.Х. Прилуцкий, Ю.С. Федосенко // Труды Горьковского ин-та инж. вод. тр-та. — Горький, 1991. — Вып. 257. — С. 55–72.
80. *Савин, В.И.* Определение оптимальной очередности обработки судов [Текст] / В.И. Савин — Горький: Волго-Вятское книжное изд-во, 1965. — 30 с.

81. Северный завоз [Текст] / Материал из Википедии — свободной энциклопедии. URL: http://ru.wikipedia.org/wiki/Северный_завоз (дата обращения: 01.03.2013)
82. Сигал, И.Х. Вычислительная реализация комбинированного алгоритма ветвей и границ для задачи коммивояжера [Текст] / И.Х. Сигал // ЖВМиМФ. — 1986. — Т.26. — №5. — С. 664–672.
83. Сигал, И.Х. Введение в прикладное дискретное программирование. Модели и вычислительные алгоритмы [Текст] / И.Х. Сигал, А.П. Иванова — М.: Физматлит, 2007. — 304 с.
84. Синий, А.В. Моделирование и оптимизация управления обслуживанием линейно рассредоточенной группы стационарных объектов процессорами транспортного типа [Текст] : дис... канд. техн. наук: 05.13.01; защищена: 15.06.06 / А.В. Синий — Н.Новгород, 1998. — 165 с. — 61:06-5/2760.
85. Танаев, В.С. Введение в теорию расписаний [Текст] / В.С. Танаев, В.В. Шкурба. — М.: Наука, 1975. — 256 с.
86. Танаев, В.С. Теория расписаний. Одностадийные системы [Текст] / В.С. Танаев, В.С. Гордон, Я.М. Шафранский — М.: Наука, 1984. — 384 с.
87. Танаев, В.С. Теория расписаний. Многостадийные системы [Текст] / В.С. Танаев, Ю.Н. Сотсков, В.А. Струевич. — М.: Наука, 1989. — 328 с.
88. Танаев, В.С. Теория расписаний. Групповые технологии [Текст] / В.С. Танаев, М.Я. Ковалев, Я.М. Шафранский. — Минск : Ин-т техн. кибернетики НАН Беларуси, 1998. — 290 с.
89. Ульянов, М.В. Математическая логика и теория алгоритмов. ч. 2. Теория алгоритмов [Текст] / М.В. Ульянов, М.В. Шептунов. — М.: МГА-ПИ, 2003. — 80 с.
90. Ульянов, М.В. Ресурсно-эффективные компьютерные алгоритмы. Разработка и анализ [Текст] / М.В. Ульянов. — М.: ФИЗМАТЛИТ, 2008. — 304 с.
91. Федосенко, Ю.С. Задача синтеза оптимально-компромиссных стратегий обслуживания бинарного потока объектов в линейной рабочей зоне двух mobile-процессоров [Текст] / Ю.С. Федосенко, А.И. Цветков. // Вестник Нижегородского университета им. Н.И. Лобачевского. — Н.Новгород, 2011. — №4 — С. 160–165.
92. Финкельштейн, Ю.Ю. Приближенные методы и прикладные задачи дискретного программирования [Текст] / Ю.Ю. Финкельштейн. — М.: Наука, 1976. — 264 с.

93. *Шлюгаев, А.Ю.* Об эффективности комплексного применения метаэвристических алгоритмов в бикритериальной задаче однопроцессорного обслуживания пространственно рассредоточенной группировки объектов [Текст] / А.Ю. Шлюгаев, Н.А. Дуничкина // XIII нижегородская сессия молодых ученых. Математические науки: Материалы докладов. — Н.Новгород, 2008. — С. 37–38.
94. *Bellman R.* Mathematical aspects of scheduling theory [Text] / R. Bellman // Journal of the Society of Industrial and Applied Mathematics. — 1956. — Vol. 4. — P. 168–205.
95. *Brucker, P.* Scheduling algorithms [Text] / P. Brucker. — Springer-Verlag, 2007. — 378 p.
96. *Coffman, E.G.* Computer and job-shop scheduling theory [Text] / E.G. Coffman. — John Wiley & Sons, 1976. — 299 p.
97. *Coffman, E.G.* Scheduling Theory and Its Applications [Text] / E.G. Coffman [et al.]. — Chichester : Wiley, 1995. — 382 p.
98. *Cohon, J.L.* Multiobjective Programming and Planning [Text] / J.L. Cohon. — New York, Academic Press, 1978. — 333 p.
99. *Conway, R.W.* Theory of Scheduling [Text] / R.W. Conway, W.L. Maxwell, L.W. Miller. — Addison-Wesley, Reading, MA, 1967. — 304 p.
100. *Dorigo, M.* Ant Colony Optimization [Text] / M. Dorigo, T. Stutzle. — MIT Press, 2004. — 319 p.
101. *Fedorin, A.N.* Branch and bound approaches to solve bicriterion discrete optimization problems [Text] / A.N. Fedorin. // VI International Congress on Mathematical Modeling. Book of Abstracts. September 20-26, 2004, Nizhniy Novgorod. — Nizhniy Novgorod : University of Nizhniy Novgorod, 2004. — 79 p.
102. *Feldmann, M.* Single-machine scheduling for minimizing earliness and tardiness penalties by meta-heuristic approaches [Text] / M. Feldmann, D. Biskup // Comput. Ind. Eng. — 2003. — Vol. 44, No. 2. — P. 307–323.
103. *Gafarov, E.R.* Single machine scheduling problems with financial resource constraints: Some complexity results and properties [Text] / E.R. Gafarov, A.A. Lazarev, F. Werner. // Mathematical Social Sciences. — Elsevier, 2011. — Vol.62(1), July. — P. 7–13.
104. *Gantt, H.L.* A graphical daily balance in manufacture [Text] / H.L. Gantt. // Transactions of the American Society of Mechanical Engineers. — 1903. — Vo.XXIV. — P. 1322–1336.
105. *Glover, F.* Future Paths for Integer Programming and Links to Artificial Intelligence [Text] / F. Glover // Comput. & Ops. — Res.Vol. 13, No.5. — 1986. — P. 533–549.

106. *Glover, F.* Tabu Search [Text] / F. Glover, M. Laguna // Modern Heuristic Techniques for Combinatorial Problems. — 1993. — P. 70–150.
107. *Graham, R.L.* Optimization and approximation in deterministic sequencing and scheduling: a survey [Text] / R.L. Graham [et al.]. // Annals of Discrete Math. — 1979. — Vo.5. — P. 287–326.
108. *Grefenstette, J.* Genetic algorithms for the traveling salesman problem [Text] / J. Grefenstette, R. Gopal, R. Rosmaita, D. Gucht. // Proceedings of the Second International Conference on Genetic Algorithms. — Mahwah, NJ : Lawrence Erlbaum Associates, 1985. — P. 539–546.
109. *Holland, J.H.* Adaptation in Natural and Artificial Systems [Text] / J.H. Holland — Ann Arbor: University of Michigan Press, 1975. — 183 p.
110. *Jackson, J.R.* An Extension of Johnson's Results on Job Lot Scheduling [Text] / J.R. Jackson. // Naval Research Logistics Quarterly. — 1956. — Vol.3. — P. 201–203.
111. *Janiak, W.A.* Single machine scheduling with job ready and delivery times subject to resource constraints [Text] / W.A. Janiak, A. Janiak, M.-C. Portmann // Parallel and Distributed Processing. IPDPS 2008. IEEE International Symposium. — 2008. — P. 1–7.
112. *Johnson, S.M.* Optimal two- and three-stage production schedules with setup times included [Text] / S.M. Johnson. // Naval Research Logistics Quarterly. — 1954. — Vol.1. — P. 61–68.
113. *Kirkpatrick, S.* Optimization by Simulated Annealing [Text] / S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi // Science — 1983. — Vo.220. — №4598. — P. 671–680.
114. *Klamroth, K.* Dynamic Programming Approaches to the Multiple Criteria Knapsack Problem [Text] / K. Klamroth, M. Wiecek // Technical Report #666. Dept. of Math. Sc., Clemson University — Clemson, SC, 1998. — 30 p.
115. *Kolesar, P.J.* A branch and bound algorithm for the knapsack problem [Text] / P.J. Kolesar // Management Sci., 13 — 1967 — pp. 723–735.
116. *Land, A.H.* An automatic method of solving discrete programming problems [Text] / A.H. Land, A.G. Doig. // Econometrica. — 1960. — V.28, №3. — P. 497–520.
117. *Lawler, E.L.* Sequencing and Scheduling: Algorithms and Complexity [Text] / E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, D.B. Shmoys. // Elsevier Science Publishers B.V. — 1993. — V.4. — P. 445–520.
118. *Lenstra, J.K.* Complexity of machine scheduling problems [Text] / J.K. Lenstra, A.H.G. Rinnooy Kan, P. Brucker. // European Journal of Operational Research. — 1980. — V.4. — P. 270–275.

119. *Papadimitriou, C.H.* Combinatorial Optimization: Algorithms and Complexity [Text] / C.H. Papadimitriou, K. Steiglitz // Englewood Cliffs, NJ: Prentice-Hall. — 1982. — 496 p.
120. *Pinedo, M.L.* Scheduling. Theory, Algorithms, and Systems [Text] / M.L. Pinedo. — Springer, 2008. — 671 p.
121. *Shen, H.* Optimal Scheduling for Satellite Refueling in Circular Orbits [Текст] / H. Shen. // PhD thesis, School of Aerospace Engineering, Georgia Institute of Technology, Atlanta, Georgia, March 2003. — 606 p.
122. *Smith, W.E.* Various optimizers for single stage production [Text] / W.E. Smith. // Naval Research Logistics Quarterly. — 1956. — V.3. — P. 59–66.
123. *T'kindt, V.* Multicriteria scheduling: models and algorithms [Text] / V. T'kindt, J. Billaut. — Springer, 2006. — 359 p.
124. *Ulungu, E.L.* Solving multi-objective knapsack problems by a branch and bound procedure [Text] / E.L. Ulungu, J. Teghem // Multicriteria Analysis. — Springer-Verlag, 1997. — P. 269–278.
125. *Villareal, B.* Multicriteria Dynamic Programming with an Application to the Integer Case [Text] / B. Villareal, M. Karwan // Journal of optimization theory and applications. — 1982 — V.38, №1. — P. 43–69.
126. *Visee, M.* Two-phases method and branch and bound procedures to solve the bi-objective knapsack problem [Text] / M. Visee, J. Teghem, M. Pirlot, E. Ulungu. // Journal of Global Optimization. — 1998. — №12(2). — P. 139–155.
127. *Zadeh, L.A.* Fuzzy Logic, Neural Networks, and Soft Computing [Text] / L.A. Zadeh // Communications of the ACM. March 1994. — Vol. 37. — No 3. — P. 77–84.
128. *Zitzler, E.* Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications [Text] / E. Zitzler. // Ph. D. thesis, Swiss Federal Institute of Technology (ETH) Zurich, Switzerland. TIK-Schriftenreihe Nr. 30, Diss ETH No. 13398, Shaker Verlag, Aachen, Germany — 1999. — 132 p.

Условия проведения вычислительных экспериментов

Все алгоритмы синтеза решений, описанные в диссертации, были реализованы программно [23].

Реализация была выполнена на языке программирования C++ в среде Microsoft Visual Studio 2005.

Вычислительные эксперименты выполнялись на компьютере с процессором Intel Core 2 Duo, 3,16 ГГц и оперативной памятью 4 Гб.

Целью проведения вычислительных экспериментов являлось определение значений скоростных характеристик разработанных программных реализаций алгоритмов.

Для значений параметров потока O_n были установлены следующие диапазоны изменения:

$$\begin{aligned} t_{i-1} &\leq t_i \leq t_{i-1} + 5, \\ 1 &\leq a_i \leq 11, \\ 1 &\leq \tau_i \leq 15, \\ t_i + \tau_i &\leq d_i \leq t_i + \tau_i + 4, \\ 1 &\leq v_i \leq V^*/2, \quad i = \overline{1, n}. \end{aligned}$$

Назначенные таким образом границы интервалов изменений исходных данных соответствуют воднотранспортным приложениям, обеспечивают достаточную плотность потока объектов и гарантируют получение весьма пессимистических оценок характеристик алгоритмов.

Размерность потока объектов O_n изменялась от 8 и до значения, при котором длительность решения задачи в среднем не превышала десяти минут.

Для каждого рассматриваемого значения размерности потока n генерировались по равномерному закону распределения данные серии из 1000 частных задач. Для каждой задачи выявлялась либо оценка, дающая минимальное значение критерию, и соответствующая ей стратегия, либо полная совокупность эффективных по Парето оценок и множество соответствующих им стратегий.

**Открытое акционерное общество
«Салехардский речной порт»**

Место нахождения:
629007, ЯНАО, г. Салехард, ул. Ленина, д. 7
Почтовый адрес:
629007, ЯНАО, г. Салехард, ул. Ленина, д. 7
Тел./факс: (34922) 4-16-11
E-mail: srp@srpmrf.ru
srp_07@mail.ru

ИНН 8901001660 / КПП 890101001
ОГРН 1028900509615 ОКПО 05162032
Р/с 407 028 101 000 000 034 40
К/с 301 018 104 000 000 008 60
в ООО КБ «Сургутский Центральный»
г.Сургут БИК 047 144 860

**АКТ ВНЕДРЕНИЯ
научно-технических результатов**

Настоящим актом подтверждается, что научно-технические результаты, полученные А.С. Куимовой при выполнении исследований по теме «Модели и алгоритмы управления обслуживанием бинарного потока объектов в однопроцессорной системе с накопительно-расходными компонентами», выполненной в ФБОУ ВПО «Волжская государственная академия водного транспорта», внедрены в ОАО «Салехардский речной порт».

ВИД ВНЕДРЕННЫХ РЕЗУЛЬТАТОВ

Методики, алгоритмы и реализованные на их основе программные средства для оценки эффективности работы диспетчерского персонала при оперативном планировании и регулировании обработки судов в порту.

НОВИЗНА РЕЗУЛЬТАТОВ НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЫ

Впервые создана диалоговая программная система поддержки принятия решений и оперативного планирования грузовой обработки судов в порту.

НАУЧНО-ТЕХНИЧЕСКИЙ УРОВЕНЬ НИР (опубликованы работы)

1. Куимова А.С., Минаев Д.В. Развитие математической модели для компьютерной системы поддержки управления очередностью обработки речных судов. Материалы I-й межвузовской научно-практической конференции студентов и аспирантов «Современные тенденции и перспективы развития водного транспорта России» 12-13 мая 2010 года. Санкт-Петербург: Санкт-Петербургский государственный университет водных коммуникаций. 2010. С. 195-197.
2. Куимова А.С. Особенности распараллеливания алгоритма оптимизации в задаче однопроцессорного обслуживания потока объектов. Сборник трудов 12-го международного научно-промышленного форума «Великие реки 2010». Нижний Новгород. 2010. Т. 2. С. 149-151.
3. Куимова А.С., Минаев Д.В., Федосенко Ю.С. Синтез стратегий обслуживания бинарного потока объектов стационарным процессором с накопительным элементом. Вестник АГТУ. Серия: Управление, вычислительная техника и информатика. Астрахань: Изд-во Астраханского государственного технического университета. 2011. №2. С. 150-157.
5. Куимова А.С. О канонической задаче диспетчеризации с двумя критериями оценки. Материалы международной научно-практической конференции «Водный транспорт России: инновационный путь развития». 6-7 октября 2010 года. СПб.: СПГУВК. 2011. Т. 3. С. 109-114.
4. Куимова А.С. Синтез оптимально-компромиссных стратегий в однопроцессорной модели с ограниченной дисциплиной обслуживания потока объектов. Theoretical and Applied Aspects of Cybernetics. Proceedings of the International Scientific Conference of Students and Young Scientists. Kyiv: Bukrek, 2011. С. 333-335.
5. Куимова А.С. Сравнительный анализ точного и приближенного алгоритма синтеза стратегий обслуживания в канонической модели диспетчеризации. Материалы II межвузовской научно-практической конференции студентов и аспирантов «Современные тенденции и

научно-практической конференции студентов и аспирантов «Современные тенденции и перспективы развития водного транспорта России» 12-13 мая 2011 года. СПб.: СПГУВК, 2011. С. 275-280.

6. Куимова А.С., Федосенко Ю.С. Модификация концепции d-расписаний для бикритериальной задачи обслуживания бинарного потока объектов в стационарной однопроцессорной системе. Сборник трудов научного конгресса 14-го международного научно-промышленного форума «Великие реки 2012». Нижний Новгород. 2012. С. 74-77.



Исполнительный директор
ОАО «Салехардский речной порт»

С.В.Кузнецов

Копия верна!

И. секретарь ФБУ ВТО «ВТБТ»



/О.А.Жукова/

РОССИЙСКАЯ ФЕДЕРАЦИЯ



СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2012661024

Программа синтеза оптимально-компромиссных стратегий
обслуживания потока объектов в задаче диспетчеризации

Правообладатель(ли): *Куимова Анастасия Сергеевна (RU)*

Автор(ы): *Куимова Анастасия Сергеевна (RU)*

Заявка № 2012619054

Дата поступления 24 октября 2012 г.

Зарегистрировано в Реестре программ для ЭВМ
5 декабря 2012 г.



Руководитель Федеральной службы
по интеллектуальной собственности

Б.П. Симонов

УТВЕРЖДАЮ
Декан факультета Вычислительной математики
и кибернетики Нижегородского государственного
университета им. Н.И. Лобачевского

профессор д.т.н.  В.П. Гергель


« ____ » _____ 2013 г.

А К Т

о внедрении в учебный процесс результатов диссертационной работы
А.С. Куимовой, представленной на соискание ученой степени кандидата
технических наук по специальности 05.13.01 «Системный анализ,
управление и обработка информации

Научные результаты, полученные А.С. Куимовой при выполнении
диссертационной работы «Модели и алгоритмы управления
обслуживанием бинарного потока объектов в однопроцессорной системе
с накопительно-расходными компонентами», внедрены в учебный
процесс со студентами факультета Вычислительной математики и
кибернетики.

Заведующий кафедрой Информатики
и автоматизации научных исследований

профессор д.т.н. 

М.Х. Прилуцкий



УТВЕРЖДАЮ

Проректор по учебно-методической работе
ФБОУ ВПО «Волжская государственная
академия водного транспорта»

доцент К.т.н.

А.А. Никитин

« 20 »

2013 г.



АКТ

о внедрении в учебный процесс результатов диссертационной работы
Куимовой Анастасии Сергеевны
«Модели и алгоритмы управления обслуживанием бинарного потока объектов
в однопроцессорной системе с накопительно-расходными компонентами»,
представленной на соискание ученой степени кандидата технических наук по специальности
05.13.01 «Системный анализ, управление и обработка информации
(в науке и промышленности)»

Материалы диссертационной работы А.С. Куимовой «Модели и алгоритмы
управления обслуживанием бинарного потока объектов в однопроцессорной системе с
накопительно-расходными компонентами» с 2012/2013 учебного года внедрены в учебный
процесс электромеханического факультета при выполнении дипломных работ студентами
5 курса специализации «Информационные и телекоммуникационные системы на
транспорте».

Декан электромеханического факультета

доцент К.т.н.

С.Г. Яковлев