Календарное планирование. Курс лекций

Автор: Ковалев М.Я.

Курс лекций в значительной степени основан на материалах серии книг по теории расписаний, включающей монографии:

- В.С. Танаев, В.В. Шкурба "Введение в теорию расписаний" (М.: Наука, 1975),
- В.С. Танаев, В.С. Гордон, Я.М. Шафранский "Теория расписаний. Одностадийные системы" (М.: Наука, 1984),
- В.С. Танаев, Ю.Н. Сотсков, В.А. Струсевич "Теория расписаний. Многостадийные системы" (М.: Наука, 1989),
- В.С. Танаев, М.Я. Ковалев, Я.М. Шафранский "Теория расписаний. Групповые технологии" (Минск: ИТК НАН Беларуси, 1998).

В курсе лекций рассматриваются детерминированные задачи построения оптимальных расписаний для различного рода обслуживающих систем. Приводятся описания практических ситуаций, в которых такие задачи возникают, и соответствующих математических моделей. Основное внимание уделено математическим методам решения задач и методам анализа их вычислительной сложности.

Курс лекций разбит на разделы. В первом разделе приводятся наиболее общие постановки задач, вводится схема обозначений и описываются основы наиболее распространенных методов решения и подходов к классификации вычислительной сложности задач. Все задачи построения оптимальных расписаний условно разделены на три класса в зависимости от характеристик обслуживающих систем. Каждый из разделов 2, 3 и 4 посвящен одному из этих классов.

Оглавление

Глава 1

Основные модели и методы

Задачи календарного планирования – это задачи оптимального распределения ограниченных ресурсов во времени. Математические модели и методы календарного планирования изучаются в рамках теории расписаний. Рассматриваемые в теории расписаний задачи обычно формулируются в терминах обслуживания *требований* в системе, состоящей из одного либо нескольких обслуживающих *приборов*. Под термином "требования" подразумеваются обрабатываемые детали, вычислительные программы, проекты, транспортные средства и т.п. Под термином "приборы" подразумеваются станки, вычислительные машины, группы исполнителей проектов, участки дорог и т.п. Приборы являются одним из ограниченных ресурсов. Требованию сопоставляется некоторое множество приборов, каждый из которых может или должен обслуживать данное требование. Если каждое требование может быть полностью обслужено любым из этих приборов, то обслуживающая система называется одностадийной (с одним или несколькими параллельными приборами). Если требование должно быть обслужено последовательно несколькими приборами, то говорят о многостадийных системах обслуживания. Многостадийные системы, в свою очередь, подразделяются на три основных типа: системы flow-, job- и open-shop, описание которых приведено ниже.

При групповых технологиях обслуживания предполагается, что требования обслуживаются группами либо партиями и при переходе от обслуживания требования одной группы (партии) к обслуживанию требования другой группы (партии) необходима переналадка прибора.

Построить расписание означает тем или иным способом указать для каждой пары "требование – прибор" интервал времени, в котором этот прибор обслуживает данное требование. При этом должен быть соблюден ряд ограничений, например, требование не может обслуживаться двумя и более приборами одновременно; на множестве требований может быть задано отношение частичного порядка, которое нельзя нарушать, и т.п.

Интерес обычно представляет построение не любых расписаний, а лишь тех из них, которые являются оптимальными относительно того или иного критерия. Критерием может быть минимизация момента завершения обслуживания всех требований, среднего времени пребывания требований в системе, суммарных затрат, суммарного либо максимального отклонения моментов завершения обслуживания требований от заданных директивных сроков и т.п.

Курс лекций включает четыре раздела: данный раздел, имеющий вводный и методологи-

ческий характер, и три раздела, посвященные различным типам обслуживающих систем: одностадийным системам, многостадийным системам и системам с групповыми технологиями обслуживания.

В разд. 1.1 приводятся наиболее общие постановки задач построения оптимальных расписаний, дается обзор существующих типов обслуживающих систем и отмечаются практические ситуации, в которых возникают соответствующие задачи. В разд. 1.2 вводятся необходимые обозначения. В разд. 1.3 обсуждаются некоторые аспекты вычислительной сложности дискретных задач. В разд. 1.4 рассматривается ситуация, когда расписание однозначно определяется указанием последовательностей обслуживания требований приборами. Вводятся обозначения для наиболее распространенных последовательностей и описывается так называемый перестановочный прием, часто применяющийся при исследовании свойств оптимальных последовательностей. В разд. 1.5 и 1.6 описываются метод динамического программирования и способ построения ε-приближенных алгоритмов, основанный на применении алгоритма динамического программирования к некоторой релаксированной задаче. В разд. 1.7 описываются некоторые приемы получения гарантированных оценок точности приближенных алгоритмов. В разд. 1.8 описываются методы минимизации приоритето-порождающих функций.

1.1. Постановка задач

Одна из наиболее общих постановок задачи построения оптимального расписания состоит в следующем. Имеется множество, состоящее из n требований, которое должно быть обслужено m приборами. Обслуживание требования прибором может происходить *с прерываниями* или *без прерываний*. В первом случае процесс обслуживания требования прибором может быть прерван и затем возобновлен в более поздний момент времени. Во втором случае такие действия запрещены. Каждый прибор может обслуживать не более одного требования в каждый момент времени. Каждое требование может обслуживаться не более чем одним прибором в каждый момент времени.

Поскольку с точки зрения построения оптимального расписания природа требований безразлична, сопоставим им числа $j = 1, \ldots, n$, которые далее будем использовать в качестве их идентификаторов. Аналогично предположим, что приборам сопоставлены числа $l = 1, \ldots, m$.

Для требования j задана *длительность* его *обслуживания* p_{lj} прибором l, $j = 1, \ldots, n$, $l = 1, \ldots, m$. Кроме того, могут быть заданы другие характеристики такие, как *момент поступления* r_j, ранее которого обслуживание этого требования не может быть начато, *директивный срок* d_j, к которому необходимо либо желательно завершить обслуживание этого требования, *весовой коэффициент (вес)* w_j, характеризующий относительную важность требования j, и неубывающая функция $f_j(t)$, определяющая *стоимость* обслуживания этого требования при условии, что его обслуживание завершается в момент времени t.

На множестве требований может быть определено *отношение предшествования (порядка)* – бинарное, транзитивное, антирефлексивное отношение. Если требование i предшествует требованию j, то обслуживание требования j не может быть начато до завершения обслуживания

i.

Мы ограничиваемся рассмотрением ситуаций, в которых все параметры обслуживающей системы и требований являются *детерминированными*, т.е. заранее определенными.

1.1.1. Определение расписания. Под *расписанием* понимается функция, которая каждому прибору l и моменту времени t сопоставляет требование, обслуживаемое прибором l в момент времени t, либо указывает, что прибор l в момент t простаивает. С целью корректного определения моментов завершения обслуживания требований, эта функция является полунепрерывной слева по t. Поскольку существуют различные формы представления функций, формы представления расписаний также различаются. Ими могут быть формулы, таблицы, графики и т.п. Наиболее известной графической формой представления расписания является *диаграмма Ганнта*.

Пример 1. Два терминала аэропорта должны в течении дня принять и отправить 12 самолетов. Некоторое расписание работы терминалов представлено диаграммой Ганнта типа (время-приборы). Определить, является ли это расписание допустимым, если заданы желательные моменты начала и завершения обслуживания самолетов, длительности их обслуживания терминалами, а также ограничение о том, что допустимое отклонение от желательных моментов начала (r_j) и завершения (d_j) обслуживания самолетов 2, 6 и 11 не должна превышать 0.5 часа.

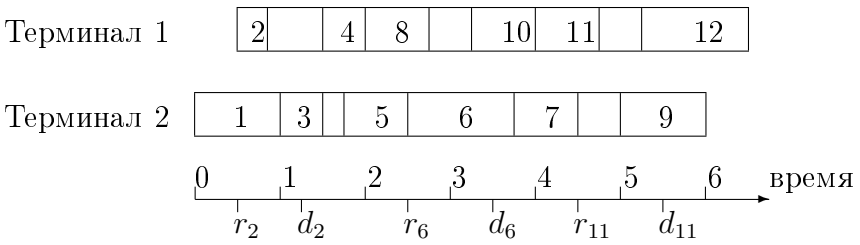

Рис. 1. Диаграмма Ганнта для примера 1.

Ответ: расписание недопустимо, так как отклонение от d_{11} превышает 0.5 часа.

Пример 2. Цех, состоящий из 5 специализированных приборов, должен изготовить три изделия. Для каждого изделия задана последовательность прохождения приборов и длительности соответствующих операций. Диаграмма Ганнта типа (время-требования) задает некоторое расписание. Определить, является ли это расписание допустимым, если каждый прибор может обрабатывать не более одного изделия в каждый момент времени.

Ответ: расписание недопустимо, так прибор 3 обслуживает одновременно два требования.

Критерием оптимальности расписания является соблюдение требованиями заданных директивных сроков, либо минимизация некоторой функции стоимости, зависящей от моментов завершения обслуживания требований.

1.1.2. Типы обслуживающих систем. Задачи построения оптимальных расписаний могут быть условно разделены на несколько классов в зависимости от типа обслуживающей си-

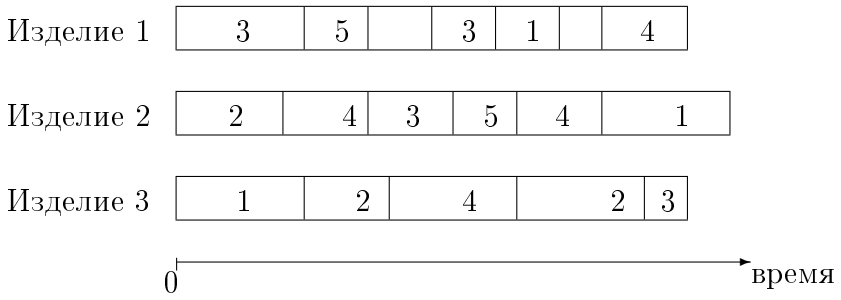

Рис. 2. Диаграмма Ганнта для примера 2.

стемы. Различают следущие типы обслуживающих систем.

Одностадийные системы. Каждое требование может быть полностью обслужено одним из m имеющихся приборов, о которых говорят, что они работают параллельно.

Многостадийные системы. Каждое требование должно быть последовательно обслужено несколькими приборами. Предполагается, что после окончания обслуживания требование немедленно освобождает прибор. Кроме того, время перехода требования с одного прибора на другой мало и им можно пренебречь. Процесс обслуживания требования прибором называется *операцией*. Момент завершения обслуживания требования равен моменту завершения выполнения последней операции этого требования. Многостадийные системы, в свою очередь, разделяются на следующие.

Система "flow-shop". Каждое требование должно быть обслужено каждым из приборов $1, \ldots, m$ в порядке $1, \ldots, m$, т.е. до начала обслуживания прибором $l+1$ обслуживание этого требования должно завершиться на приборе l, $l = 1, \ldots, m-1$.

Система "open-shop". Каждое требование должно быть обслужено каждым из приборов $1, \ldots, m$, проходя их в произвольном порядке. Маршруты прохождения приборов требованиями являются частью принимаемого решения.

Система "job-shop". В этой системе требования имеют заданные, не обязательно одинаковые маршруты прохождения приборов $1, \ldots, m$, в которых приборы могут повторяться.

Задание 1. Построить диаграммы Ганнта для некоторых задач и вычислить значения ряда целевых функций.

Задание 2. К какому классу относится обслуживающая система в следующих задачах:

2.1. Один строитель в течение смены должен выполнить 7 заданий, каждое из которых требует исходные материалы, поставляемые в разные моменты времени. Составить расписание работы строителя такое, чтобы все задания были выполнены в заданные сроки.

2.2. Два крана обслуживают подъемные работы на 17 строительных объектах. Заданы временные интервалы выполнения работ и прибыль от их выполнения. Составить расписание работы кранов, максимизирующее суммарную прибыль.

2.3. Ряд деталей должен пройти обработку на токарном, фрезерном и шлифовальном стан-

ках в указанной последовательности. Заданы длительности обработки каждой детали на каждом станке. Определить порядок обработки деталей каждым станком такой, чтобы минимизировать среднее время пребывания деталей в обслуживающей системе.

2.4. На каждом из 10 полей необходимо провести 5 сельскохозяйственных работ в любой последовательности. Каждая работа выполняется уникальной машиной. Составить план выполнения работ машинами такой, чтобы завершить все работы как можно быстрее.

2.5. При выпуске лекарств используются химические ванны. Для каждого типа лекарства задана определенная последовательность прохождения некоторых из этих ванн. Одновременно в ванне может находиться один тип лекарства. Составить расписание выпуска заданного количества лекарств разных типов, максимизирующее загрузку всех ванн.

1.2. Обозначения

Для представления задач построения оптимальных расписаний используется обозначение $\alpha/\beta/\gamma$, состоящее из трех полей. Ниже приводятся обозначения для классических задач теории расписаний, не затрагивающие задачи с групповыми технологиями обслуживания.

1.2.1. Первое поле $\alpha = \alpha_1\alpha_2$ описывает обслуживающую систему. Предполагается, что $\alpha_1 \in \{\cdot, P, Q, R, O, F, J\}$, где $\cdot$ обозначает пустой символ. Первые четыре символа служат для обозначения одностадийной системы обслуживания:

- $\alpha_1 = \cdot$: один прибор; $p_{1j} = p_j$;
- $\alpha_1 = P$: *параллельные идентичные приборы*; $p_{lj} = p_j$ для всех приборов;
- $\alpha_1 = Q$: *параллельные приборы разной производительности*; $p_{lj} = p_j/v_l$, где v_l – *производительность* прибора l;
- $\alpha_1 = R$: *параллельные различные приборы*; p_{lj} являются произвольными целыми неотрицательными числами;
- $\alpha_1 = O$: open-shop;
- $\alpha_1 = F$: flow-shop;
- $\alpha_1 = J$: job-shop.

Если $\alpha_1 \in \{O, F, J\}$, то p_{lj} являются произвольными целыми неотрицательными числами.

Символ α_2 служит для описания количества приборов. В случае $\alpha_2 = m$ имеется в виду, что количество приборов фиксировано и равно m. В случае $\alpha_2 = \cdot$ количество приборов – переменная величина, значение которой является элементом входных данных задачи. Обозначение $\alpha_1 = \cdot$ применяется лишь в сочетании с $\alpha_2 = 1$.

1.2.2. Второе поле β описывает характеристики требований. Предполагается, что $\beta \subset \{\beta_1, \ldots, \beta_5\}$, где β_k определены следующим образом:

- $\beta_1 \in \{\cdot, pmtn\}$;
- $\beta_1 = \cdot$: прерывания обслуживания любого требования запрещены;
- $\beta_1 = pmtn$: прерывания разрешены;
- $\beta_2 \in \{\cdot, p_j = p, p_{lj} = p\}$;
- $\beta_2 = \cdot$: длительности обслуживания требований p_j либо p_{lj} произвольны;
- $\beta_2 \in \{p_j = p, p_{lj} = p\}$: p_j либо p_{lj} одинаковы и равны p для всех требований и приборов;
- $\beta_3 \in \{\cdot, d_j = d\}$;
- $\beta_3 = \cdot$: если заданы директивные сроки d_j, то они являются произвольными;
- $\beta_3 = (d_j = d)$: d_j одинаковы и равны d для всех требований;
- $\beta_4 \in \{\cdot, r_j\}$;
- $\beta_4 = \cdot$: моменты поступления равны нулю для всех требований;
- $\beta_4 = r_j$: заданы произвольные моменты поступления требований r_j;
- $\beta_5 \in \{\cdot, prec, tree, in-tree, out-tree, s\text{-}p, chain\}$;
- $\beta_5 = \cdot$: на множестве требований не заданы отношения предшествования;
- $\beta_5 = prec$: на множестве требований заданы отношения предшествования, которые представляются произвольным ацикличным ориентированным графом;
- $\beta_5 \in \{tree, in-tree, out-tree, chain\}$: граф отношений предшествования является деревом, входящим деревом, выходящим деревом, последовательно-параллельным графом и набором цепей соответственно.

Для отдельных задач могут быть использованы некоторые другие легко интерпретируемые обозначения. Необходимые пояснения даются по ходу изложения.

1.2.3. Третье поле, γ, описывает критерий задачи, который состоит в отыскании расписания, допустимого относительно директивных сроков либо минимизирующего некоторую *целевую функцию*. Значение целевой функции зависит от следующих величин, которые могут быть вычислены для каждого требования j при заданном расписании:

- *момент завершения обслуживания* C_j;
- *временное смещение* $L_j = C_j - d_j$;
- *запаздывание* $T_j = \max\{0, C_j - d_j\}$;

– *единичный штраф* U_j, где $U_j = 1$, если требование j является *запаздывающим* ($C_j > d_j$), и $U_j = 0$, если требование j не является запаздывающим ($C_j \leq d_j$). Не запаздывающие требования будем называть *ранними*.

Расписание *допустимо относительно директивных сроков*, если

$C_j \leq d_j$ для всех j.

Рассматриваются следующие целевые функции:

– *максимальная стоимость* $f_{\max} = \max\{f_j(C_j)\}$;

– *общий момент завершения обслуживания* $C_{\max} = \max\{C_j\}$;

– *максимальное временное смещение* $L_{\max} = \max\{L_j\}$;

– *максимальное запаздывание* $T_{\max} = \max\{T_j\}$;

– *суммарная стоимость* $\sum f_j = \sum f_j(C_j)$;

– *(взвешенная) сумма моментов завершения* $\sum(w_j)C_j$;

– *(взвешенное) число запаздывающих требований* $\sum(w_j)U_j$;

– *(взвешенное) суммарное запаздывание* $\sum(w_j)T_j$.

В приведенных выше обозначениях каждый максимум и сумма берется по всем требованиям j. Предполагается, что функции $f_{\max}$ и $\sum f_j$ являются *регулярными*, т.е. неубывающими от моментов завершения обслуживания требований. Очевидно, что остальные приведенные выше функции также являются регулярными.

Третье поле γ содержит обозначение одного из перечисленных критериев, т.е. $\gamma \in \{C_j \leq d_j, f_{\max}, \ldots, \sum w_j T_j\}$.

1.2.4. Ниже описываются несколько примеров, показывающих, как используются приведенные обозначения.

Задача $1/prec/\sum w_j C_j$: имеется один прибор; на множестве требований определены произвольные отношения предшествования; критерий состоит в минимизации взвешенной суммы моментов завершения. Кроме того, запрещены прерывания и все требования готовы к обслуживанию в момент времени ноль.

Задача $Qm/p_j = p, d_j = d/C_j \leq d_j$: имеется m параллельных приборов разной производительности, где m фиксировано; все значения p_j и все значения d_j одинаковы; критерий состоит в отыскании расписания, допустимого относительно заданных директивных сроков. Запрещены прерывания. Все требования готовы к обслуживанию в момент времени ноль.

Задание. Расшифровать ряд обозначений задач построения расписаний.

1.3. Основы теории сложности вычислений

В теории сложности вычислений вводится понятие трудноразрешимой задачи. Существование эффективного (быстрого) алгоритма решения хотя бы одной из трудноразрешимых задач влечет существование такого рода алгоритма для любой из таких задач. Из этого следует, что для трудноразрешимых задач существование эффективных алгоритмов решения маловероятно.

1.3.1. Задачи распознавания и экстремальные комбинаторные задачи. Под *задачей распознавания (свойств объектов)* будем понимать некоторый общий вопрос, предполагающий ответ "да", если рассматриваемый в этом вопросе объект обладает свойствами, описанными в формулировке вопроса. Объект и его свойства описываются с помощью некоторых параметров, конкретные значения которых в общей формулировке вопроса отсутствуют. Параметрами могут быть множества, графы, числа, функции и т.п.

Пример задачи получается при замене всех имеющихся в общей формулировке задачи параметров их конкретными значениями. Пример называется *да-примером*, если при соответствующих значениях параметров объект обладает указанными в формулировке задачи свойствами.

Экстремальная комбинаторная задача состоит в следующем. На конечном множестве X' определена функция $F(x)$, $x \in X'$. Необходимо в заданном подмножестве X множества X' отыскать такой элемент x^*, что $F(x^*) = \min\{F(x)|x \in X\}$ (задача минимизации), либо такой x^0, что $F(x^0) = \max\{F(x)|x \in X\}$ (задача максимизации).

Сопоставим экстремальной задаче B следующую задачу распознавания B': определить, существует ли такой элемент x' в заданном множестве X, что $F(x') \leq y$ (либо $F(x') \geq y$ в случае задачи максимизации) для данного действительного числа y. Очевидно, если x^* решение задачи минимизации B, то элемент $x' \in X$ такой, что $F(x') \leq y$, существует тогда и только тогда, когда $F(x^*) \leq y$. Таким образом, экстремальная задача не проще соответствующей задачи распознавания.

1.3.2. Разумные схемы кодирования. Поскольку задачи интересуют нас с точки зрения разработки алгоритмов их решения и реализации этих алгоритмов на ЭВМ, входные данные задачи должны быть каким-то образом закодированы. Схема кодирования сопоставляет цепочку символов каждому примеру задачи.

Оценка вычислительной (временной) сложности алгоритма решения задачи – это функция от длины записи входной информации задачи, т.е. она непосредственно зависит от выбранной схемы кодирования.

Можно подобрать схему кодирования, которая дает очень длинные коды примеров задачи, так что почти любой алгоритм будет эффективным относительно длин таких входов и мы не сможем определить, какой алгоритм лучше. Кроме того, схема кодирования может выдавать коды существенно разной длины для приблизительно одинаковых примеров одной и той же задачи. В этом случае непонятно как определять вычислительную сложность алгоритма – на одних примерах он будет эффективным, на других нет.

Эти и другие противоречия будут устранены, если пользоваться так называемыми *разум-*

ными схемами кодирования. Будем говорить, что схема кодирования является разумной, если она удовлетворяет следующим неформальным условиям.

(а) *Код примера задачи должен быть компактным и не должен содержать необязательную информацию или символы.*

(б) *Числа из примера задачи должны быть представлены в двоичной, либо какой-нибудь иной системе счисления с основанием, отличным от единицы.*

Система счисления с основанием, равным 1, называется унарной. В этой системе число 3 представляется в виде 111, число 4 в виде 1111 и т.д. (как зарубки у древних людей). В унарной системе счисления для представления целого числа x требуется x единиц памяти. В двоичной системе счисления все числа представляются в виде последовательностей двух чисел: 0 и 1. Число 3 представляется в виде 11, число 4 в виде 100, число 5 в виде 101 и т.д. Для представления целого числа x требуется $O(\log_2 x)$ нулей и единиц (единиц памяти). Функция $O(x)$ определяется следующим образом:

$$\lim_{x\to\infty} \frac{O(x)}{x} = const.$$

(в) *Схема кодирования должна обладать свойством декодируемости,* т.е. для каждого параметра задачи должен существовать эффективный алгоритм, способный выделить код этого параметра из кода любого примера задачи.

(г) *Схема кодирования должна обеспечивать однородность при кодировании различных примеров одной и той же задачи.* Однородность означает, что схема кодирования должна обеспечивать для любых двух различных примеров задачи получение кодов, "близких" по длине, если эти примеры содержат близкие по величине числа и количество параметров в обоих примерах приблизительно одно и то же.

Выполнение условия однородности может быть обеспечено соблюдением следующего простого правила. Схема кодирования не должна заниматься распознаванием специфических свойств кодируемого примера задачи. При построении кода конкретного примера задачи схема может использовать лишь те свойства, которые непосредственно указаны в формулировке задачи и являются общими для всех примеров данной задачи. Таким образом, *код примера задачи должен содержать лишь ту информацию, которую содержит общая формулировка задачи, дополненная конкретными значениями параметров задачи.*

Условие (г) предотвращает появление слишком коротких кодов для отдельных примеров задачи. С другой стороны, условие (а) не дает возможности появляться слишком длинным кодам.

1.3.3. Вычислительная (временная) сложность алгоритмов. Полиномиальные и псевдополиномиальные алгоритмы. Под временной сложностью алгоритма понимается количество *элементарных операций*, необходимое для его реализации. Для алгоритмов, предназначенных для реализации на ЭВМ, под элементарными понимаются такие машинные операции, как сложение, умножение, сравнение двух чисел, запись либо считывание числа, расположенного по известному адресу, и т. п.

Для любого примера I некоторой задачи введем в рассмотрение две функции:

– функция $Length[I]$, определяющая длину кода примера I при использовании разумной схемы кодирования и

– функция $Max[I]$, определяющая величину наибольшего числового параметра в I. Если задача не содержит чисел, то полагаем $Max[I] = 1$ для любого примера I.

Как уже отмечалось, оценка вычислительной сложности алгоритма решения задачи – это функция от длины записи входной информации задачи.

Алгоритм решения задачи называется *полиномиальным*, если его временная сложность не превосходит некоторого полинома от функции $Length[I]$ для любого примера I этой задачи. Соответствующая задача называется *полиномиально разрешимой*. Полиномиальные алгоритмы принято считать эффективными, поскольку полиномиальная функция при больших значениях аргумента ($Length[I]$) растет не так быстро как экспоненциальная.

Полиномиально разрешимые задачи образуют класс $\mathcal{P}$.

Алгоритм решения задачи называется *псевдополиномиальным*, если его временная сложность не превосходит некоторого полинома от двух функций: $Length[I]$ и $Max[I]$, - для любого примера I этой задачи. Соответствующая задача называется *псевдополиномиально разрешимой*.

1.3.4. Класс $\mathcal{NP}$ задач. NP-полные и NP-трудные задачи. Задача принадлежит классу $\mathcal{NP}$, если она может быть решена за полиномиальное время на так называемой недетерминированной машине Тьюринга, которая реально не существует. На такой машине за полиномиальное время можно реализовать как полиномиальные так и некоторые неполиномиальные алгоритмы. Можно сказать, что на некоторых этапах своей работы, когда есть несколько вариантов дальнейших действий, недетерминированная машина обращается к помощи оракула (предсказателя будущего), который подсказывает дальнейшие действия.

Задача П называется NP-*трудной*, если любая задача из класса $\mathcal{NP}$ *полиномиально сводится* к ней, т.е. имея полиномиальный алгоритм решения NP-трудной задачи, можно получить полиномиальный алгоритм решения любой задачи из класса $\mathcal{NP}$. Задача распознавания П называется NP-*полной*, если она NP-трудна и принадлежит $\mathcal{NP}$.

Понятие полиномиальной сводимости транзитивно. Поэтому для доказательства NP-трудности некоторой задачи достаточно полиномиально свести к ней некоторую NP-трудную задачу. Для этого для любого примера известной NP-трудной задачи за полиномиальное от его размерности время построить пример нашей задачи и показать, что решение NP-трудной задачи существует тогда и только тогда, когда существует решение построенного примера нашей задачи.

Экстремальная задача называется NP-трудной, если соответствующая ей задача распознавания является NP-трудной.

Из существования полиномиального алгоритма решения какой-нибудь NP-трудной задачи следует существование полиномиальных алгоритмов решения для всех задач из класса $\mathcal{NP}$, включая все NP-полные задачи.

Как уже отмечалось, все полиномиально разрешимые задачи полиномиально разрешимы на недетерминированной машине Тьюринга. Поэтому

$$\mathcal{P} \subseteq \mathcal{NP}.$$

Однако неясно $\mathcal{P} = \mathcal{NP}$ или $\mathcal{P} \neq \mathcal{NP}$.

Поскольку ни для одной NP-трудной задачи не разработан полиномиальный алгоритм решения, в настоящее время общепринятой является гипотеза о том, что

$$\mathcal{P} \neq \mathcal{NP}.$$

Для того, чтобы ее опровергнуть, достаточно построить полиномиальный алгоритм решения любой (какой-нибудь одной) NP-трудной задачи. Список таких задач включает десятки тысяч задач из различных областей науки.

За разработку полиномиального алгоритма решения любой NP-трудной задачи установлен приз в 1.000.000 долларов.

1.3.5. NP-трудные в сильном смысле задачи. Наряду с разделением задач на NP-трудные и полиномиально разрешимые, NP-трудные задачи, в свою очередь, подразделяются на NP-трудные в сильном смысле задачи и задачи, имеющие псевдополиномиальные алгоритмы решения. NP-трудные в сильном смысле задачи не имеют псевдополиномиального алгоритма решения.

1.3.6. Примеры NP-трудных задач.

NP-трудная задача, имеющая псевдополиномиальный алгоритм решения:

РАЗБИЕНИЕ:

Заданы целые положительные числа $a_1, a_2, \ldots, a_r$ и A такие, что $\sum_{j=1}^{r} a_j = 2A$. Требуется определить, существует ли множество $X \subset N = \{1, 2, \ldots, r\}$ такое, что $\sum_{j \in X} a_j = A$.

NP-трудная в сильном смысле задача:

3-РАЗБИЕНИЕ:

Заданы целые положительные числа $a_1, a_2, \ldots, a_{3r}$ и A такие, что $A/4 < a_j < A/2$, $j = 1, \ldots, 3r$, и $\sum_{j=1}^{3r} a_j = rA$. Требуется определить, существует ли разбиение множества $\{1, 2, \ldots, 3r\}$ на r непересекающихся подмножеств $X_1, X_2, \ldots, X_r$ такое, что $\sum_{j \in X_l} a_j = A$, $l = 1, \ldots, r$.

Нетрудно заметить, если решение задачи 3-РАЗБИЕНИЕ существует, то каждое из множеств $X_1, \ldots, X_r$ содержит в точности три элемента.

Нетрудно заметить, что если решение задачи 3-РАЗБИЕНИЕ существует, то каждое из множеств $X_1, \ldots, X_r$ содержит в точности три элемента.

Задание 1. Сравнить временную сложность следующих алгоритмов решения задачи РАЗБИЕНИЕ:

Алгоритм 1: Построить всевозможные разбиения и проверить, существует ли искомое подмножество.

Алгоритм 2: Упорядочить числа $a_1, a_2, \ldots, a_r$ по невозрастанию. Назначать в множество X индексы $1, 2, \ldots$ до тех пор, пока сумма соответствующих чисел не превосходит A. (Алгоритм отыскивает приближенное решение задачи)

Временные сложности равны $O(2^r)$ и $O(r \log r)$ соответственно. Вычислить значения функций 2^r и $r \log r$ при $r = 5, 10$.

Задание 2. Доказать, что задача $P2//C_{\max}$ является NP-трудной. В качестве эталонной использовать задачу РАЗБИЕНИЕ.

1.4. Оптимальные последовательности требований и перестановочный прием

1.4.1. Оптимальные последовательности. Во многих задачах оптимального планирования расписания однозначно определяются указанием последовательностей обслуживания требований каждым из приборов. В ряде случаев удается установить, что существуют оптимальные последовательности, в которых требования упорядочены по неубыванию либо невозрастанию значений некоторого параметра a_j, $j = 1, \ldots, n$, сопоставленного требованию.

Ниже приводятся обозначения наиболее распространенных последовательностей.

Последовательность EDD (Earliest Due Date) – требования упорядочены по неубыванию директивных сроков $d_j : d_1 \le \cdots \le d_n$.

Последовательность ERT (Earliest Release Time) – требования упорядочены по неубыванию моментов поступления $r_j : r_1 \le \cdots \le r_n$.

Последовательность SPT (Shortest Processing Time) – требования упорядочены по неубыванию длительностей обслуживания $p_j : p_1 \le \cdots \le p_n$.

Последовательность SWPT (Shortest Weighted Processing Time) – требования упорядочены по неубыванию отношений $p_j/w_j : p_1/w_1 \le \cdots \le p_n/w_n$.

Последовательность LPT (Longest Processing Time) – требования упорядочены по невозрастанию длительностей обслуживания $p_j : p_1 \ge \cdots \ge p_n$.

1.4.2. Перестановочный прием. При доказательстве того, что существует оптимальная последовательность требований, удовлетворяющая определенным условиям, зачастую используется так называемый *перестановочный прием*, который состоит в следующем.

Предполагается, что существует оптимальная последовательность, в которой расположение некоторых требований i и j не удовлетворяет необходимым условиям. Например, предполагается, что в оптимальной последовательности требование i предшествует требованию j, в то время как с точки зрения рассматриваемых условий они должны быть расположены в обратном порядке. Затем рассматривается последовательность, полученная из исходной путем перестановки требований i и j местами. Показывается, что в результате значение целевой функции либо уменьшается (что противоречит предположению об оптимальности исходной последовательности), либо не увеличивается (что свидетельствует об оптимальности и новой последовательности). Применение указанной процедуры конечное число раз приводит к оптимальной

последовательности, удовлетворяющей требуемым условиям.

Зачастую используется обобщение описанной техники, при котором переставляются не отдельные требования, а целые сегменты последовательности.

В дальнейшем, если при доказательстве некоторого утверждения техника применения перестановочного приема не вызывает затруднений, отмечается лишь возможность использования этого приема и само доказательство не приводится.

Задание. Доказать, что последовательности SPT и EDD являются оптимальными для задач $1//\sum C_j$ и $1//L_{\max}$ соответственно.

1.5. Динамическое программирование

Динамическое программирование (ДП) является универсальным методом решения экстремальных задач.

Существует несколько интерпретаций метода ДП. Мы рассмотрим ДП как процесс построения множеств частичных решений исходной задачи и выбора из этих множеств *доминирующих* решений таких, что хотя бы одно из них может быть "достроено" до оптимального.

При таком подходе процесс решения некоторой задачи может быть организован следующим образом. Для определенности, рассмотрим задачу минимизации

$$f(x) \to \min,\ x \in X.$$

Будем формировать множества $X_0, X_1, \ldots, X_n$ частичных решений, где множество X_{k+1} получается из множества X_k путем "достраивания" каждого решения из X_k. Оптимальное решение x^* выбирается из множества X_n.

С каждым частичным решением $x \in X_k$ будем связывать некоторый набор параметров $B = (k, b_1, \ldots, b_i)$, называемых *переменными состояния*. Набор B переменных состояния называется *состоянием*.

Пусть $X_k(B)$ обозначает множество частичных решений $x \in X_k$ в состоянии (соответствующих состоянию) B. Пусть $B(x)$ обозначает состояние частичного решения x.

В методе ДП с каждым состоянием B связывается функция *доминирования* $F(B)$ такая, что частичное решение, находящееся в состоянии B и максимизирующее (либо минимизирующее) $F(B)$, *доминирует* все остальные частичные решения в состоянии B, т.е. это частичное решение может быть достроено до полного допустимого решения с наименьшим значением целевой функции среди всех полных допустимых решений, достроенных из любого частичного решения в состоянии B.

Из определения функции $F(B)$ следует, что в каждом множестве $X_k(B)$ достаточно выбрать доминирующее решение для дальнейших построений.

Обозначим через S множество всех возможных состояний. Это множество называется *пространством состояний*.

В методе ДП рекуррентно вычисляют значения $F(B)$ и находят состояние, соответствующее оптимальному решению. Затем восстанавливают само оптимальное решение.

Для применения метода ДП необходимо корректно определить

1) пространство состояний S,

2) функцию доминирования $F(B)$, $B \in S$,

3) способ построения частичных решений $x' \in X_{k+1}$ из частичных решений $x \in X_k$,

4) метод вычисления состояния $B(x')$, используя состояние $B(x)$, если $x' \in X_{k+1}$ "достроен" из $x \in X_k$, и

5) рекуррентную формулу для вычисления значений $F(B)$.

1.5.1. Пример алгоритма ДП. В качестве примера рассмотрим следующую задачу. Служащий должен выполнить n работ к заданному сроку d, начиная с момента времени ноль. Для каждой работы j задана длительность выполнения p_j и штраф w_j, выплачиваемый с случае завершения j после d. Все параметры являются неотрицательными целыми числами. Требуется найти такую последовательность выполнения работ, чтобы суммарный штраф был наименьшим.

При заданной последовательности работ $(i_1, \ldots, i_n)$ легко определить момент завершения выполнения каждой работы i_j: $C_{i_j} = \sum_{k=1}^{j} p_{i_k}$.

Введем в рассмотрение переменные U_j: $U_j = 1$, если работа j является запаздывающей $(C_j > d)$, и $U_j = 0$, если j является ранней $(C_j \leq d)$. Требуется найти такую последовательность работ, что значение $\sum_{j=1}^{n} w_j U_j$ минимально.

Заметим, что на значение целевой функции влияет лишь информация о том, какие работы являются запаздывающими, а какие ранними. Поэтому существует оптимальная последовательность работ, в которой все ранние работы упорядочены произвольно и все запаздывающие работы упорядочены произвольно и расположены после последней ранней работы. Тогда задача может быть сформулирована следующим образом:

$$\sum_{j=1}^{n} w_j U_j \to \min$$

при условиях

$$\sum_{j=1}^{n} p_j (1 - U_j) \leq d,$$

и

$$U_j \in \{0, 1\},\ j = 1, \ldots, n.$$

Эта задача NP-трудна.

Пусть $U^* = (U_1^*, \ldots, U_n^*)$ – оптимальный вектор для этой задачи и W^* – соответствующее значение целевой функции.

Сформулированная задача, в свою очередь, может быть проинтерпретирована в терминах ДП следующим образом.

Будем формировать частичные решения, определяя переменные $U_j = 0$ либо $U_j = 1$ для $j = 1, \ldots, n$. Будем говорить, что частичное решение $(U_1, \ldots, U_k)$ находится в состоянии (k, a), если определены значения переменных $U_1, \ldots, U_k$ и значение ограничения $\sum_{j=1}^{k} p_j (1 - U_j) = a$.

Переменные состояния могут принимать значения $k = 0, 1, \ldots, n$ и $a = 0, 1, \ldots, d$.

Из начального состояния $(0,0)$ можно перейти в состояние $(1,0)$, при котором $U_1 = 1$, либо в состояние $(1,p_1)$, где $U_1 = 0$, если $p_1 \le d$.

Рассмотрим некоторое состояние (k,a), $k \in \{2,\dots,n\}$. В это состояние можно перейти только из состояния $(k-1,a)$, если $U_k = 1$, либо из состояния $(k-1,a-p_k)$, если $U_k = 0$.

Определим функцию доминирования $F(k,a)$ как наименьшее значение целевой функции $\sum_{j=1}^{k} w_j U_j$, вычисленное для решений в состоянии (k,a). Из определения функции $F(k,a)$ следует, что

$$W^* = \min\{F(n,a) | a = 0,1,\dots,d\}.$$

Значения $F(k,a)$ могут быть вычислены рекуррентно. Для инициализации рекуррентных вычислений положим $F(0,0) = 0$ и $F(k,a) = \infty$ для всех $(k,a) \neq (0,0)$.

Из определения функции $F(k,a)$ следует, что общее *рекуррентное соотношение* может быть записано как

$$F(k,a) = \min\{F(k-1,a) + w_k, F(k-1,a-p_k)\}, \tag{1.1}$$

$$k = 1,\dots,n,\ a = 0,1,\dots,d.$$

При этом, если $F(k,a) = F(k-1,a) + w_k$, то в частичном решении, соответствующем состоянию (k,a), переменная $U_k = 1$. Если же $F(k,a) = F(k-1,a-p_k)$, то в этом решении $U_k = 0$.

1.5.2. Перебор с возвратом (backtracking). Оптимальное решение U^* может быть найдено с помощью следующей процедуры, называемой *перебор с возвратом.* Эта процедура использует таблицу размерности $n \times (d+1)$, в каждой ячейке (k,a) которой хранится информация о том, на первой или второй компоненте достигается минимум в (1.1) для указанных значений k и a. На первой итерации полагаем $k = n$ и $a = a^*$, где a^* определяется из $W^* = F(n,a^*)$. Если минимум в (1.1) достигается на первой компоненте, то полагаем $U_k^* = 1$, $k := k-1$, оставляем a без изменений и переходим к следующей итерации процедуры перебора с возвратом. Если минимум достигается на второй компоненте, то полагаем $U_k^* = 0$, $a := a - p_k$, $k := k-1$, и переходим к следующей итерации. После выполнения n итераций будет построен оптимальный вектор $U^* = (U_1^*,\dots,U_n^*)$.

1.5.3. Время работы и требуемая память. Эффективность алгоритма ДП определяется его временной сложностью и объемом требуемой памяти. Анализ алгоритма для рассматриваемой задачи показывает, что вычисление $F(k,a)$ требует выполнения одной операции сложения и одной операции сравнения для каждой пары (k,a). Таким образом, временная сложность этого алгоритма равна $O(nd)$.

Что касается объема памяти, то следует отметить, что для вычисления оптимального значения целевой функции W^* на каждом шаге k достаточно хранить таблицу размерности $2 \times (d+1)$ со значениями $F(k-1,a)$ и $F(k,a)$, $a = 0,1,\dots,d$. Для отыскания оптимального вектора U^* достаточно применить процедуру обратного хода, которая требует $n(d+1)$ дополнительных единиц памяти.

Из приведенного анализа можно сделать вывод, общий для всех алгоритмов ДП: *временная*

сложность и объем требуемой памяти алгоритма ДП непосредственно зависят от мощности соответствующего пространства состояний.

1.5.4. Выбор пространства состояний. Прямые и обратные алгритмы ДП. Как правило, существует несколько вариантов для выбора пространства состояний. Например, для рассматриваемой задачи можно ввести состояния (k, w) такие, что частичное решение находится в состоянии (k, w), если определены значения переменных $U_1, \ldots, U_k$ и $\sum_{i=1}^{k} w_i U_i = w$. В этом случае функция $F(k, w)$ определяется как наименьшее значение ограничения $\sum_{i=1}^{k} p_i(1-U_i)$ для решений в состоянии (k, w). Рекуррентное соотношение можно записать в виде

$$F(k,w) = \min \begin{cases} F(k-1,w) + p_k, & \text{если } F(k-1,w) + p_k \le d, \\ F(k-1, w-w_k) \end{cases}$$

Соответствующий алгоритм ДП требует $O(n \sum_{k=1}^{n} w_k)$ единиц времени и памяти.

Алгоритмы ДП, в которых частичные решения формируются от начала к концу, как это делается в приведенных выше алгоритмах, называются *прямыми*. Альтернативный подход состоит в построении частичных решений от конца к началу. Такие алгоритмы называются *обратными*. Например, рассматриваемая задача может быть решена в пространстве состояний (k, a) следующим образом. Полагаем $F(n+1, 0) = 0$, все остальные начальные значения $F(k, a)$ равными бесконечности и вычисляем

$$F(k,a) = \min\{F(k+1, a-p_k), F(k+1,a) + w_k\}$$

для $k = n, n-1, \ldots, 1$ и $a = 0, 1, \ldots, d$. Оптимальное значение целевой функции равно $W^* = \min\{F(1,a) |\ a = 0, 1, \ldots, d\}$.

1.5.5. Числовой пример решения задачи о рюкзаке методом ДП.

$$F(x) = x_1 + 2x_2 + x_3 + x_4 + 2x_5 + x_6 + 2x_7 \Rightarrow \max,$$

$$B(x) = 2x_1 + x_2 + 2x_3 + 2x_4 + x_5 + x_6 + x_7 \le 5,\ x_i \in \{0,1\}.$$

X_0	(0)
$F(x)$	0
$B(x)$	0

X_1	(0)	(1)
$F(x)$	0	1
$B(x)$	0	2

X_2	(0,0)	(0,1)	(1,0)	(1,1)
$F(x)$	0	2	1	3
$B(x)$	0	1	2	3

X_3	(0,0,0)	(0,0,1)	(0,1,0)	(0,1,1)	(1,0,0)	(1,0,1)	(1,1,0)	(1,1,1)
$F(x)$	0	1	2	3	1	2	3	4
$B(x)$	0	2	1	3	2	4	3	5

3 предпоследние частичные решения решения можно дальше не рассматривать, так как есть доминирующие. Далее достраиваем только доминирующие.

X_4	(0,0,0,0)	(0,0,0,1)	(0,0,1,0)	(0,0,1,1)	(0,1,0,0)	(0,1,0,1)	(0,1,1,0)	(0,1,1,1)	(1,1,1,0)	(1,1,1,1)
$F(x)$	0	1	1	2	2	3	3	4	4	5
$B(x)$	0	2	2	4	1	3	3	5	5	7

X_5	(0,0,0,0,0)	(0,0,0,0,1)	(0,0,0,1,0)	(0,0,0,1,1)	(0,1,0,0,0)	(0,1,0,0,1)	(0,1,1,1,0)	(0,1,1,1,1)
$F(x)$	0	2	1	3	2	4	4	6
$B(x)$	0	1	2	3	1	2	5	6

X_6	(0,0,0,0,0,0)	(0,0,0,0,0,1)	(0,0,0,1,0,0)	(0,0,0,1,0,1)	(0,0,0,1,1,0)	(0,0,0,1,1,1)	(0,1,0,0,1,0)	(0,1,0,0,1,1)
$F(x)$	0	1	2	3	3	4	4	5
$B(x)$	0	1	1	2	3	4	2	3

X_7	(0,0,0,0,0,0,0)	(0,0,0,0,0,0,1)	(0,0,0,0,0,1,0)	(0,0,0,0,0,1,1)	(0,0,0,1,0,0,0)	(0,0,0,1,0,0,1)
$F(x)$	0	2	1	3	2	4
$B(x)$	0	1	1	2	1	2

X_7	(0,0,0,1,0,1,0)	(0,0,0,1,0,1,1)	(0,1,0,0,1,0,0)	(0,1,0,0,1,0,1)	(0,1,0,0,1,1,0)	**(0,1,0,0,1,1,1)**
$F(x)$	3	5	4	6	5	**7**
$B(x)$	2	3	2	3	3	4

Оптимальное решение $x^* = (0, 1, 0, 0, 1, 1, 1)$ со значением целевой функции $F(x^*) = 7$.

Задание. Разработать алгоритмы ДП для решения задач $1//\sum w_j U_j$ (с произвольными директивными сроками) и $P2//C_{\max}$. Для задачи минимизации $\sum w_j U_j$ вначале показать, что в оптимальном расписании ранние требования обслуживаются в последовательности EDD.

1.6. Построение вполне полиномиальных ε-приближенных алгоритмов

В этом разделе описывается метод построения эффективных ε-приближенных алгоритмов для задач математического программирования с сепарабельными целевыми функциями. Метод основан на применении алгоритма динамического программирования к релаксированной задаче, получаемой из исходной задачи путем специального округления значений целевой функции и переменных.

Рассмотрим задачу минимизации либо максимизации функции $F(x)$ при условии $x \in X$, где X - некоторое конечное множество.

Пусть F^* обозначает оптимальное значение целевой функции этой задачи и пусть F^H обозначает значение целевой функции для элемента $x \in X$, полученного при помощи некоторого приближенного алгоритма H. Предположим, что $F^* > 0$. Алгоритм H называется *ε-приближенным алгоритмом*, если для любого заданного числа $\varepsilon > 0$ и любого примера задачи он отыскивает такой элемент $x \in X$, что $|F^* - F^H| \leq \varepsilon F^*$. Нетрудно видеть, что ε – относительная погрешность решения. Если ε-приближенный алгоритм имеет временную сложность, которая ограничена некоторым полиномом от длины записи входных данных задачи в двоичном алфавите, то он называется *полиномиальным ε-приближенным алгоритмом*. Если к тому же его временная сложность ограничена некоторым полиномом от $1/\varepsilon$, то он называется *вполне полиномиальным ε-приближенным алгоритмом*.

Далее формулируется минисумная задача и приводится ε-приближенный алгоритм K_ε ее решения. Алгоритм K_ε является вполне полиномиальным ε-приближенным алгоритмом, если определены подходящие нижние и верхние оценки оптимального значения целевой функции. Аналогичный подход применяется для минимаксной задачи. Описывается также процедура улучшения нижних и верхних оценок оптимального значения целевой функции, основанная на применении некоторого приближенного алгоритма.

В терминах минимаксных и минисумных задач могут быть сформулированы многие задачи оптимального планирования.

1.6.1. Минисумная задача. Пусть на множестве R действительных чисел определены неотрицательные неубывающие функции $f_i(x), i = 1, \ldots, n$; на множестве $R^n = R \times \ldots \times R$ определена функция $Z(x) = Z(x_1, \ldots, x_n)$, неубывающий по каждому аргументу, и $A \in R$.

Минисумная задача KMIN формулируется следующим образом.

$$F(x) = \sum_{i=1}^{n} f_i(x_i) \to \min,$$

при условиях

$$Z(x) \geq A, \tag{1.2}$$

$$x_i \in R_i, i = 1, \ldots, n, \tag{1.3}$$

где $R_i \subset R$, $i = 1, \ldots, n$, – заданные конечные множества.

Эта задача может быть сформулирована в терминах задачи о рюкзаке следующим образом. Предположим, что предметы n типов должны быть упакованы в рюкзак. Предметы каждого типа i упаковываются единой партией. Возможный размер x_i такой партии выбирается из множества R_i. Объем, занимаемый x_i предметами типа i, определяется функциями $f_i(x_i)$. Стоимость всех выбранных предметов определяется функцией $Z(x_1, \ldots, x_n)$. Задача состоит в минимизации суммарного объема выбранных предметов при условии, что их стоимость не ниже заданного значения A.

Обозначим точное решение задачи KMIN через $x^* = (x_1^*, \ldots, x_n^*)$. Очевидно, что x^* существует тогда и только тогда, когда $Z(x') \geq A$, где $x' = (x_1', \ldots, x_n')$, и x_i' – наибольший элемент множества R_i, $i = 1, \ldots, n$. Предположим, что x^* существует и известны числа V и U такие, что $0 < V \leq F(x^*) \leq U$.

Сформулируем *округленную задачу* KR. В постановке этой задачи используется величина $\delta = \varepsilon V/n$. Далее, $\lfloor a \rfloor$ обозначает наибольшее целое число, не превосходящее a. Задача KR состоит в следующем:

$$f(x) = \sum_{i=1}^{n} \lfloor f_i(x_i)/\delta \rfloor \to \min,$$

при условиях (1.2) и

$$x_i \in \{x_i(0), x_i(1), \ldots, x_i(\lfloor U/\delta \rfloor)\}, i = 1, \ldots, n, \tag{1.4}$$

где $x_i(l)$ является наибольшим элементом $x \in R_i$, удовлетворяющим условию $\lfloor f_i(x)/\delta \rfloor = l$, или, что эквивалентно, условию $l\delta \leq f_i(x) < (l+1)\delta$. Если такого элемента не существует, то $x_i(l) = \phi$.

Предположим, что для любого $l \in \{0, 1, \ldots, \lfloor U/\delta \rfloor\}$ за время $O(\tau)$ можно найти $x_i(l)$ либо установить, что его не существует. Тогда задача KR может быть сформулирована за время $O(\tau n U/\delta)$.

Установим взаимосвязь задачи KMIN и округленной задачи KR.

Теорема 1.1. ***Любой точный алгоритм решения задачи KR является ε-приближенным алгоритмом для задачи KMIN.***

Доказательство. Рассмотрим точное решение x^0 задачи KR. По определению оно удовлетворяет ограничениям (1.2), (1.4) и, следовательно, (1.2), (1.3). Остается показать, что из существования x^* следует существование x^0 и $F(x^0) \le (1+\varepsilon)F(x^*)$.

Предположим, что x^* существует. Тогда существует решение x' задачи KMIN такое, что компонента x'_i этого решения – наибольший элемент множества R_i, удовлетворяющий соотношению $\lfloor f_i(x'_i)/\delta \rfloor = \lfloor f_i(x^*_i)/\delta \rfloor$, $i = 1, \ldots, n$. По крайней мере, $x'_i = x^*_i$ для $i = 1, \ldots, n$. Поскольку по определению $x'_i \ge x^*_i$ для $i = 1, \ldots, n$, и функция $Z(x)$ является неубывающей, $Z(x') \ge Z(x^*) \ge A$. Таким образом, если x^* существует, то существует по крайней мере одно решение, удовлетворяющее ограничениям (1.2), (1.4), т.е. существует x^0.

Покажем, что $F(x^0) \le (1+\varepsilon)F(x^*)$. Справедливы соотношения:

$$F(x^0) = \sum_{i=1}^{n} f_i(x^0_i) \le \delta \sum_{i=1}^{n} \lfloor f_i(x^0_i)/\delta \rfloor + n\delta \le$$

$$\delta \sum_{i=1}^{n} \lfloor f_i(x^*_i)/\delta \rfloor + n\delta \le F(x^*) + n\delta \le (1+\varepsilon)F(x^*).$$

■

Ниже приводится алгоритм динамического программирования K_ε решения задачи KR для случая, когда функция $Z(x)$ является сепарабельной. В этом алгоритме используется верхняя оценка оптимального значения $f(x^0)$ целевой функции этой задачи, которая получается следующим образом:

$$f(x^0) \le f(x^*) \le F(x^*)/\delta \le U/\delta.$$

Предположим, что функция Z имеет вид

$$Z(x) = \sum_{i=1}^{n} z_i(x_i),$$

где $z_i(x)$, $i = 1, \ldots, n$, являются некоторыми неубывающими функциями. В алгоритме K_ε рекурсивно вычисляются значения $Z_j(f)$, где $Z_j(f)$ – максимальное значение $Z(x)$ при условии, что определены значения первых j переменных $x_1, \ldots, x_j$, и $\sum_{i=1}^{j} \lfloor f_i(x_i)/\delta \rfloor = f$. Формальное описание алгоритма состоит в следующем.

Алгоритм K_ε.

Шаг 1. (Инициализация). Полагаем $Z_j(f) = 0$, если $f = 0, j = 0$, и $Z_j(f) = -\infty$ в противном случае. Полагаем $j = 1$.

Шаг 2. (Рекурсия). Для $f = 0, 1, \ldots, \lfloor U/\delta \rfloor$ вычисляем следующее:

$$Z_j(f) = \max\{Z_{j-1}(f - \lfloor f_j(x_j)/\delta \rfloor) + z_j(x_j) \mid x_j \in \{x_j(0), \ldots, x_j(f)\}\}.$$

Если $j < n$, то полагаем $j := j + 1$ и повторяем шаг 2. В противном случае переходим к шагу 3.

Шаг 3. (Отыскание x^0). Оптимальное значение целевой функции равно

$$\min\{f|Z_n(f) \geq A, f = 0, 1, \ldots, \lfloor U/\delta \rfloor\}.$$

Соответствующее решение x^0 отыскивается при помощи поиска с возвратом.

Временная сложность алгоритма K_ε равна $O(\sum_{j=1}^{n} U \min\{U/\delta,\ |R_j|\}/\delta)$ или, что эквивалентно, $O(\sum_{j=1}^{n} nU \min\{nU/(\varepsilon V), |R_j|\}/(\varepsilon V))$. Если мощность каждого множества R_j ограничена некоторой константой, как, например, в обычной задаче о рюкзаке с 0-1 переменными, то временная сложность алгоритма равна $O(n^2U/(\varepsilon V))$. В противном случае она равна $O((n^3/\varepsilon^2)(U/V)^2)$. Эти оценки временной сложности могут быть уменьшены до $O(n^2/\varepsilon + n^2 \log\log(U/V))$ и $O(n^3/\varepsilon^2 + n^3 \log\log(U/V))$ соответственно, если использовать процедуру улучшения оценок, приведенную ниже в п. 1.6.3. Таким образом, если значение U/V ограничено некоторой экспонентой от длины записи входных данных задачи в унарном алфавите, то объединение алгоритма K_ε с процедурой улучшения оценок образует вполне полиномиальный ε-приближенный алгоритм для задачи KMIN.

1.6.2. Минимаксная задача. Минимаксная задача PMIN состоит в следующем.

$$T(y) = \max\{\sum_{i=1}^{n} f_{li}(y_{li}) | l = 1, \ldots, m\} \to \min,$$

при условиях

$$y_{li} \in \{0, 1, \ldots, q_i\}, i = 1, \ldots, n, l = 1, \ldots, m, \tag{1.5}$$

$$\sum_{l=1}^{m} y_{li} = q_i, i = 1, \ldots, n, \tag{1.6}$$

где $f_{li}(\cdot)$ – некоторые неотрицательные функции, $i = 1, \ldots, n, l = 1, \ldots, m$.

Эта задача допускает следующую интерпретацию. Предположим, что предметы n типов должны быть упакованы в m контейнеров. Общее количество предметов типа i равно q_i. Если y_{li} предметов типа i упаковываются в контейнер l, то они занимают $f_{li}(y_{li})$ единиц его объема. Задача состоит в минимизации объема наиболее заполненного контейнера. Такая задача является в некотором смысле двойственной к известной *задаче об упаковке в контейнеры*, в которой каждый контейнер имеет ограниченный объем, количество контейнеров неограничено, и задача состоит в том, чтобы упаковать предметы таким образом, чтобы минимизировать количество используемых контейнеров.

Для того чтобы разработать ε-приближенный алгоритм для задачи PMIN, сформулируем следующую округленную задачу PR.

$$t(y) = \max\{\sum_{i=1}^{n} \lfloor f_{li}(y_{li})/\delta \rfloor | l = 1, \ldots, m\} \to \min,$$

при условиях

$$(y_{1i}, \ldots, y_{mi}) \in Q_i, i = 1, \ldots, n, \tag{1.7}$$

где Q_i является множеством решений $(u_1, \ldots, u_m)$ следующих систем уравнений, определенных для всех различных наборов $(r_1, \ldots, r_m), r_l \in \{0, 1, \ldots, \lfloor U/\delta \rfloor\}, l = 1, \ldots, m$.

$$\lfloor f_{li}(u_l)/\delta \rfloor = r_l, l = 1, \ldots, m, \tag{1.8}$$

$$u_l \in \{0, 1, \ldots, q_i\}, l = 1, \ldots, m, \tag{1.9}$$

$$\sum_{l=1}^{m} u_l = q_i. \tag{1.10}$$

Используя соотношение $\lfloor y \rfloor \leq y < \lfloor y \rfloor + 1$, условия (1.8), (1.9) могут быть преобразованы к виду

$$u_l \in \{a_l, a_l + 1, \ldots, b_l\}, l = 1, \ldots, m, \tag{1.11}$$

где a_l и b_l – такие числа, что $f_{li}(a_l) = \min\{f_{li}(u) | f_{li}(u) \geq \delta r_l, u \in \{0, 1, \ldots, q_i\}\}$ и $f_{li}(b_l) = \max\{f_{li}(u) | f_{li}(u) < \delta(r_l + 1), u \in \{0, 1, \ldots, q_i\}\}$.

Предположим, что числа a_l и b_l могут быть найдены за время $O(\tau)$ для любого l. Покажем, что задача PR может быть сформулирована за время $O((m + \tau)n(U/\delta)^m)$. Понятно, что $|Q_i| \leq O((U/\delta)^m)$ для $i = 1, \ldots, n$. Остается показать, что система (1.10), (1.11) может быть решена за время $O(m)$.

Если $\sum_{l=1}^{m} a_l > q_i$ либо $\sum_{l=1}^{m} b_l < q_i$, то (1.10), (1.11) и, следовательно, (1.8)-(1.10) не имеет решения. Предположим, что

$$\sum_{l=1}^{m} a_l \leq q_i \leq \sum_{l=1}^{m} b_l. \tag{1.12}$$

Найдем такой индекс $k, 1 \leq k \leq m$, что

$$q_i \leq \sum_{l=1}^{k-1} a_l + \sum_{l=k}^{m} b_l \text{ и } q_i \geq \sum_{l=1}^{k} a_l + \sum_{l=k+1}^{m} b_l. \tag{1.13}$$

Здесь $\sum_{l=1}^{k-1} a_l = 0$, если $k = 1$, и $\sum_{l=k+1}^{m} b_l = 0$, если $k = m$. В силу предположения (1.12) такое k всегда существует. Зададим значения u_l следующим образом:

$$u_l = a_l, l = 1, \ldots, k-1, u_l = b_l, l = k+1, \ldots, m,$$

$$u_k = q_i - (\sum_{l=1}^{k-1} a_l + \sum_{l=k+1}^{m} b_l).$$

Понятно, что $\sum_{l=1}^{m} u_l = q_i$. Далее, из (1.13) следует $a_l \leq u_l \leq b_l$. Таким образом, $(u_1, \ldots, u_m)$ является решением системы (1.8)–(1.10).

Очевидно, что приведенная процедура решения системы (1.10), (1.11) и, следовательно, системы (1.8)–(1.10) требует время $O(m)$, если значения $\sum_{l=1}^{k} a_l$ и $\sum_{l=k}^{m} b_l$, $k = 1, \ldots, m$, вычислены заранее. Поскольку число различных наборов $(r_1, \ldots, r_m)$ и значений i, для которых необходимо решать систему (1.8)–(1.10), не превосходит $(U/\delta)^m$ и n соответственно, задача PR может быть сформулирована за время $O((m + \tau)n(U/\delta)^m)$.

Установим взаимосвязь исходной задачи PMIN и округленной задачи PR.

Теорема 1.2. ***Любой точный алгоритм решения задачи PR является ε-приближенным алгоритмом для задачи PMIN.***

Доказательство. Обозначим точные решения задач PMIN и PR через y^* и y^0 соответственно. Для доказательства теоремы достаточно показать, что из существования y^* следует существование y^0 и $T(y^0) \leq (1+\varepsilon)T(y^*)$.

Предположим, что y^* существует. Тогда существует матрица y' такая, что для любого $i \in \{1,\ldots,n\}$ вектор $(y'_{1i},\ldots,y'_{mi})$ является решением системы

$$\lfloor f_{li}(y'_{li})/\delta \rfloor = \lfloor f_{li}(y^*_{li})/\delta \rfloor, l = 1,\ldots,m,$$

$$y'_{li} \in \{0,1,\ldots,q_i\}, l = 1,\ldots,m, \sum_{l=1}^{m} y'_{li} = q_i.$$

По крайней мере $(y^*_{1i},\ldots,y^*_{mi})$ является решением этой системы. Таким образом, $Q_i \neq \phi$ для $i = 1,\ldots,n$, т.е. y^0 существует.

Покажем, что $T(y^0) \leq (1+\varepsilon)T(y^*)$. Напомним, что $\delta = \varepsilon V/n$ и $V \leq T(y^*)$. Справедлива следующая цепочка неравенств:

$$T(y^0) \leq \delta \max\{\sum_{i=1}^{n} \lfloor f_{li}(y^0_{li})/\delta \rfloor | l = 1,\ldots,m\} + n\delta \leq$$

$$\delta \max\{\sum_{i=1}^{n} \lfloor f_{li}(y'_{li})/\delta \rfloor | l = 1,\ldots,m\} + n\delta =$$

$$\delta \max\{\sum_{i=1}^{n} \lfloor f_{li}(y^*_{li})/\delta \rfloor | l = 1,\ldots,m\} + n\delta \leq T(y^*) + n\delta \leq (1+\varepsilon)T(y^*).$$

■

Ниже приводится алгоритм динамического программирования решения задачи PR. Нетрудно заметить, что из $T(y^*) \leq U$ следует $t(y^0) \leq U/\delta$. Эта верхняя оценка для $t(y^0)$ используется в приведенном ниже алгоритме P_ε решения задачи PR. В алгоритме P_ε формируется последовательность множеств $S_0, S_1, \ldots, S_n$. Каждое множество S_j включает различные наборы $(T_1,\ldots,T_m)$. Каждый набор из S_j соответствует набору переменных $y_{li}, i = 1,\ldots,j, l = 1,\ldots,m$, для которого выполняются ограничения (1.7) и $T_l = \sum_{i=1}^{j} \lfloor f_{li}(y_{li})/\delta \rfloor \leq U/\delta$, $l = 1,\ldots,m$.

Алгоритм P_ε.

Шаг 1. (Инициализация). Полагаем $S_0 = \{(0,\ldots,0)\}$ и $j = 1$.

Шаг 2. (Формирование $S_1,\ldots,S_n$). Полагаем $S_j = \phi$. Для каждого набора $(T_1,\ldots,T_m) \in S_{j-1}$ и для каждого набора $(y_{1j},\ldots,y_{mj}) \in Q_j$ вычисляем $T'_l = T_l + \lfloor f_{lj}(y_{lj})/\delta \rfloor$. Если $T'_l \leq U/\delta$ для $l = 1,\ldots,m$ и набора $(T'_1,\ldots,T'_m)$ нет в S_j, то добавляем его в S_j. При $j < n$ полагаем $j := j+1$ и повторяем шаг 2. В противном случае переходим к шагу 3.

Шаг 3. (Отыскание y^0). Набор из S_n с наименьшим значением $\max\{T_l | l = 1,\ldots,m\}$ соответствует решению y^0, которое может быть найдено при помощи поиска с возвратом.

Установим временную сложность алгоритма P_ε. Очевидно, что процедура формирования множества S_j требует время $O(m|Q_j||S_{j-1}|)$. Поскольку для каждого набора $(T_1,\ldots,T_m) \in S_j$ выполняется $T_l \le U/\delta$, $l = 1,\ldots,m$, то $|S_j| \le (U/\delta)^m$ для $j = 1,\ldots,n$. Таким образом, временная сложность алгоритма P_ε не превосходит $O(mn(U/\delta)^m)$ или, что эквивалентно, $O((U/V)^m mn^{m+1}/\varepsilon^m)$. Используя процедуру улучшения оценок (см. п. 1.6.3), эта временная сложность может быть понижена до $O(m3^m n^{m+1}/\varepsilon^m + m2^m n^{m+1}\log\log(U/V))$, т.е. алгоритм P_ε с включенной в него процедурой улучшения оценок является вполне полиномиальным ε-приближенным алгоритмом, если m фиксировано, и величина U/V ограничена по крайней мере экспонентой от длины записи входных данных задачи в унарном алфавите.

1.6.3. Процедура улучшения оценок. Процедура улучшения нижней и верхней оценки оптимального значения целевой функции задачи минимизации позволяет отыскивать такое число F^0, что $F^0 \le F^* \le 3F^0$, где F^* – минимальное значение целевой функции. Процедура основана на применении специального приближенного алгоритма. Ниже описываются свойства, которыми этот алгоритм должен обладать для того, чтобы процедура была корректной и эффективной.

Пусть $F^* > 0$ и имеется приближенный алгоритм $X(A,B)$, удовлетворяющий следующим условиям:

(1) для любых положительных чисел A и B и произвольного набора входных данных задачи алгоритм $X(A,B)$ отыскивает решение со значением целевой функции $F^X \le B + A$, если $F^* \le B$;

(2) временная сложность алгоритма $X(A,B)$ ограничена полиномом $P(Length, B/A)$ от двух переменных, где $Length$ – длина записи входных данных задачи в двоичном алфавите.

Предположим, что для рассматриваемой задачи известны нижняя и верхняя оценки такие, что $0 < V \le F^* \le U$. Если $U > 3V$, то предлагается процедура отыскания числа F^0 такого, что $F^0 \le F^* \le 3F^0$.

В этой процедуре применяется модификация *метода двоичного поиска* (в интервале $[V,U]$).

Процедура улучшения оценок.

Шаг 1. Полагаем $F^0 = V$, $j = 1$, $a_j = 0$ и находим целое число b_j такое, что $2^{b_j-1}V < U \le 2^{b_j}V$.

Шаг 2. Вычисляем $k_j = \lceil (a_j + b_j)/2 \rceil$ и $F^{(k_j)} = 2^{k_j-1}V$.

Шаг 3. Применяем алгоритм $X(F^{(k_j)}, 2F^{(k_j)})$. Если он находит решение со значением $F^X \le 3F^{(k_j)}$, то полагаем $F^0 = F^{(k_j)}$, и, если $k_j = b_j$, то процедура завершается. Если $k_j < b_j$, полагаем $a_{j+1} = a_j$ и $b_{j+1} = k_j$. Если алгоритм $X(F^{(k_j)}, 2F^{(k_j)})$ не находит решения со значением $F^X \le 3F^{(k_j)}$, то при $k_j = b_j$ процедура завершается. При $k_j < b_j$ полагаем $a_{j+1} = k_j$ и $b_{j+1} = b_j$. В обоих случаях полагаем $j := j + 1$ и переходим к шагу 2.

В качестве алгоритма X можно использовать ε-приближенный алгоритм, удовлетворяющий условиям (1) и (2), полагая $\varepsilon = 1$. Следует отметить, что многие ε-приближенные алгоритмы, в том числе все приведенные в данном курсе лекций, удовлетворяют этим условиям при $\varepsilon = 1$.

Процедура улучшения оценок и связанные с ней результаты легко адаптируются для задач максимизации.

Задание. Разработать ε-приближенные алгоритмы для задач $P2//C_{\max}$ и $2//\sum w_jC_j$. Для задачи минимизации $\sum w_jC_j$ вначале показать, что в оптимальном расписании каждый прибор обслуживает требования в последовательности SWPT.

1.7. Приближенные алгоритмы с гарантированными оценками точности

Рассмотрим задачу минимизации целевой функции $F(x)$ на множестве X. Обозначим $F^* = \min\{F(x)|x \in X\}$. Предположим, что $F^* > 0$.

Пусть приближенный алгоритм H отыскивает элемент множества X со значением F^H целевой функции. Введем в рассмотрение величину $\Delta_H = F^H/F^*$. Эта величина зависит от алгоритма H и от набора входных данных задачи. В данном разделе под гарантированной оценкой точности решения, получаемого при помощи H, будем понимать верхнюю оценку величины Δ_H при любом наборе входных данных задачи.

Построение эффективных приближенных алгоритмов с гарантированными оценками точности имеет смысл для NP-трудных в сильном смысле задач и NP-трудных задач, для которых не известны быстродействующие ε-приближенные алгоритмы.

В дальнейшем обозначения F^*, F^H и Δ_H используются при рассмотрении конкретных задач и конкретных приближенных алгоритмов. Смысл этих обозначений не изменяется.

1.7.1. Параллельные приборы. Минимизация $\sum w_jC_j$. Рассмотрим NP-трудную в сильном смысле задачу $P//\sum w_jC_j$. В этой задаче расписание однозначно определяется разбиением множества требований на подмножества $N_1, \ldots, N_m$, где требования множества N_l обслуживаются прибором l, и указанием порядка обслуживания требований в каждом подмножестве. При помощи перестановочного приема нетрудно показать, что достаточно ограничиться рассмотрением расписаний, при которых требования каждого подмножества упорядочены по правилу SWPT, т.е. по неубыванию значений p_j/w_j. Таким образом, расписание однозначно определяется разбиением множества требований на m подмножеств.

Опишем приближенный алгоритм A решения задачи $P//\sum w_jC_j$, который состоит в следующем. Перенумеруем требования в порядке SWPT так, что $p_1/w_1 \leq \cdots \leq p_n/w_n$. Будем последовательно, начиная с первого, назначать требования $1, \ldots, n$ на приборы. Пусть требования $0, \ldots, k$, $k < n$, назначены, где 0 – фиктивное требование, соответствующее начальной ситуации, $p_0 = 0$. Требование $k+1$ назначается следующим образом. Обозначим через P_l сумму длительностей обслуживания требований, назначенных на прибор l, $l = 1, \ldots, m$. Находим такой прибор l, что $P_l = \min_{1 \leq h \leq m} P_h$. Назначаем требование $k+1$ на прибор l и, если $k+1 \neq n$, переходим к назначению требования $k+2$.

Теорема 1.3. ***Для любого расписания справедливо*** $\sum w_jC_j \geq \frac{1}{m}F_1^* + \frac{m-1}{2m}F_n^*$.

Из теоремы следует, что $F^* \geq \frac{1}{m}F_1^* + \frac{m-1}{2m}F_n^* \geq F^A - \frac{m-1}{2m}F_n^*$.

Сейчас нетрудно получить оценку сверху величины Δ_A :

$$\Delta_A = F^A/F^* \leq (F^* + \frac{m-1}{2m}F_n^*)/F^* \leq \frac{3m-1}{2m} = \frac{3}{2} - \frac{1}{2m},$$

поскольку $F_n^* \leq F^*$.

1.7.2. Задача об упаковке в контейнеры. Рассмотрим NP-трудную в сильном смысле задачу об упаковке в контейнеры, которая в терминах теории расписаний может быть сформулирована следующим образом. В условиях задачи $P/d_j = d/C_j \leq d_j$ предполагается, что $m = n$, $d \geq \max_j p_j$ и требуется отыскать минимальное число приборов и соответствующее расписание, при котором все требования будут обслужены к заданному директивному сроку d.

Приведем описание четырех приближенных алгоритмов решения задачи об упаковке в контейнеры. В алгоритмах рассматривается некоторая перестановка π требований, требования последовательно выбираются из этой перестановки и назначаются на приборы в соответствии с некоторыми правилами. Резервом времени прибора l, обозначаемым R_l, называется разница между d и суммарной длительностью обслуживания требований, назначенных на прибор l.

В алгоритмах B1 и B2 перестановка π является произвольной. На шаге k алгоритма B1 или B2 выбирается требование, расположенное на месте k в перестановке π. В алгоритме B1 это требование назначается на прибор с наименьшим номером среди приборов с достаточным резервом времени для завершения этого требования с срок. В алгоритме B2 это требование назначается на прибор с наименьшим достаточным резервом времени.

Алгоритмы B3 и B4 отличаются от алгоритмов B1 и B2 соответственно лишь тем, что требования в перестановке π расположены в порядке LPT невозрастания значений p_j.

Пусть m_i – количество приборов, найденное этим алгоритмом Bi и m^* - наименьшее возможное количество приборов.

Теорема 1.4. ***Справедливы неравенства*** $m_i < \frac{17}{10}m^* + 2$ ***для*** $i = 1, 2$ ***и*** $m_i \leq \frac{11}{9}m^* + 4$ ***для*** $i = 3, 4$.

1.8. Минимизация приоритето-порождающих функций

Пусть Π_r – множество всех перестановок $\pi_r = (i_1, \ldots, i_r)$ элементов множества $N = \{1, \ldots, n\}$, $r = 1, \ldots, n$, $\Pi_0 = \{\pi_0\} = \{(\phi)\}$ и $\Pi = \cup_{r=0}^{n}\Pi_r$.

На множестве Π_n определена функция $F(\pi)$. Эта функция называется *приоритето-порождающей,* если существует функция $\omega(\pi)$, $\pi \in \Pi$, называемая *функцией приоритета,* которая обладает следующими свойствами: для любых перестановок $\pi = (\pi^1, \pi^a, \pi^b, \pi^2) \in \Pi_n$ и $\pi' = (\pi^1, \pi^b, \pi^a, \pi^2) \in \Pi_n$

- из $\omega(\pi^a) > \omega(\pi^b)$ следует $F(\pi) \leq F(\pi')$ и
- из $\omega(\pi^a) = \omega(\pi^b)$ следует $F(\pi) = F(\pi')$.

Многие задачи построения оптимальных расписаний сводятся к минимизации приоритето-порождающих функций на частично упорядоченных множествах требований.

Множество N является *частично упорядоченным,* если на нем задано *отношение предшествования (бинарное, транзитивное, антирефлексивное отношение),* представленное *графом редукции этого отношения* $G = (N, U)$. Граф G называется графом редукции отношения предшествования, если он получен из графа отношения частичного порядка путем удаления всех транзитивных дуг.

В задачах оптимального планирования, если i предшествует j, то обслуживание требования j не может быть начато до завершения обслуживания требования i. Отношения предшествования присутствуют в задачах, где некоторые операции используют результаты других (предшествующих) операций.

1.8.1. Пример приоритето-порождающей функции. Рассмотрим задачу $1/prec/\sum C_j$.

Обозначим через $\sum C_j(\pi)$ значение $\sum C_j$ для частичного расписания, представленного перестановкой $\pi = (i_1, \ldots, i_r) : \sum C_j(\pi) = \sum_{k=1}^{r} \sum_{j=1}^{k} p_{i_j}$.

Введем в рассмотрение функцию $\sum C_j(\pi, K) = \sum_{k=1}^{r}(K + \sum_{j=1}^{k} p_{i_j})$, где K – некоторая константа, представляющая момент времени, начиная с которого начинают обслуживаться требования перестановки π. Обозначим $P(\pi) = \sum_{j\in\{\pi\}} p_j$.

Вычислим

$$\sum C_j(\pi^1,\pi^a,\pi^b,\pi^2)=\sum C_j(\pi^1)+\sum C_j(\pi^a,P(\pi^1))+\sum C_j(\pi^b,P(\pi^1,\pi^a))+\sum C_j(\pi^2,P(\pi^1,\pi^a,\pi^b)),$$

$$\sum C_j(\pi^1,\pi^b,\pi^a,\pi^2)=\sum C_j(\pi^1)+\sum C_j(\pi^b,P(\pi^1))+\sum C_j(\pi^a,P(\pi^1,\pi^b))+\sum C_j(\pi^2,P(\pi^1,\pi^b,\pi^a)).$$

Поскольку $P(\pi^1, \pi^a, \pi^b)) = P(\pi^1, \pi^b, \pi^a))$, неравенство $\sum C_j(\pi^1, \pi^a, \pi^b, \pi^2) \le \sum C_j(\pi^1, \pi^b, \pi^a, \pi^2)$ эквивалентно

$$\sum C_j(\pi^a, P(\pi^1)) + \sum C_j(\pi^b, P(\pi^1, \pi^a)) \le \sum C_j(\pi^b, P(\pi^1)) + \sum C_j(\pi^a, P(\pi^1, \pi^b)),$$

что эквивалентно

$$|\{\pi^b\}|P(\pi^a) \le |\{\pi^a\}|P(\pi^b)|.$$

Из последнего неравенства следует, что в качестве функции приоритета можно взять $\omega(\pi) = |\{\pi\}|/P(\pi)$.

Задание. Найти функцию приоритета для задачи $1/prec/\sum w_j C_j$.

Ответ: $\omega(\pi) = W(\pi)/P(\pi)$, где $W(\pi) = \sum_{j\in\{\pi\}} w_j$.

1.8.2. Методы минимизации приоритето-порождающих функций на частично упорядоченных множествах. Пусть задано частично упорядоченное множество N с графом редукции отношения частичного порядка $G = (N, U)$. Задача состоит в минимизации $F(\pi)$, $\pi \in \Pi_n(G)$, где $\Pi_n(G)$ – множество всех перестановок элементов множества N, допустимых относительно G.

Задача $1|out - tree|F$, F **- приоритето-порождающая функция.** Цепь $(i_1, \ldots, i_k)$, где компоненты i_j являются составными вершинами, называется *w-цепью*, если $w(i_l) \ge w(i_{l+1})$, $l = 1, \ldots, k-1$.

Алгоритм минимизации приоритето-порождающей функции на множестве $\Pi_n(G)$, где G – набор выходящих деревьев состоит в следующем.

1. Вычисляем приоритеты висячих вершин.

2. Если G не есть набор изолированных вершин, то находим в G вершину i^0, называемую *опорной*, все прямые потомки которой являются висячими. Пусть этим потомкам соответствуют w-цепи $C_1, \ldots, C_l$. Построим w-цепь $(i_1, \ldots, i_\nu)$, упорядочив все (составные) вершины цепей $C_1, \ldots, C_l$ по невозрастанию приоритетов. Построим цепь $(i^0, i_1, \ldots, i_\nu)$. Если $w(i^0) > w(i_1)$, то цепь $(i^0, i_1, \ldots, i_\nu)$ является w-цепью. Если $w(i^0) \leq w(i_1)$, то объединяем i^0 и i_1 в составную вершину $[i^0, i_1]$. Далее сравниваем $w(i^0, i_1)$ и $w(i_2)$ и, в случае необходимости, объединяем $[i^0, i_1]$ и i_2. Процесс продолжается до тех пор, пока цепь $(i^0, i_1, \ldots, i_\nu)$ не будет преобразована в некоторую w-цепь $C^0 = ([i^0, i_1, \ldots, i_k], i_{k+1}, \ldots, i_\nu)$. Удаляем из G всех потомков вершины i^0 и ставим ей в соответствие w-цепь C^0.

Повторяем описанный процесс до тех пор, пока не будет построен граф, состоящий из изолированных вершин. Последовательность (составных) вершин соответствующих w-цепей, в которой вершины упорядочены по невозрастанию приоритетов, является оптимальным решением задачи.

В случае, когда G – входящее дерево, в качестве опорной выбирается вершина, все непосредственные предшественники которой $i_1, \ldots, i_\nu$ не имеют предшественников. Формируется цепь $(i_1, \ldots, i_\nu, i^0)$. Она преобразуется в w-цепь путем сравнения $w(i^0)$ и $w(i_\nu)$. Составная вершина $[i_\nu, i^0]$ образуется, если $w(i_\nu) \leq w(i^0)$. Далее процесс аналогичен случаю выходящего дерева.

Задание. Решить задачу $1/out - tree/\sum C_j$, в которой имеется 10 требований. Требование 3 предшествует требованию 4, которое, в свою очередь, предшествует требованиям 1, 7 и 9. Длительности обслуживания p_j заданы в таблице

j	1	2	3	4	5	6	7	8	9	10
p_j	4	2	3	5	7	4	1	2	9	4

Ответ: $(\{2, 8\}, 3, 4, 7, \{1, 6, 10\}, 5, 9)$. В процессе решения образуется составной элемент $[3, 4, 7]$.

Глава 2

Одностадийные системы обслуживания

В данной главе рассматриваются обслуживающие системы, состоящие из одного либо нескольких параллельных приборов. Каждое требование может быть полностью обслужено любым из приборов.

Поскольку функции стоимости неубывающие, случай $r_j = 0$ и $m \geq n$ является тривиальным – достаточно назначить каждое требование на отдельный прибор.

2.1. Один прибор. Максимальный штраф.

Рассмотрим задачу $1/prec/f_{\max}$. Поскольку моменты поступления требований равны нулю и прерывания процесса обслуживания запрещены, расписание однозначно определяется последовательностью требований.

Пусть отношения предшествования заданы ориентированным бесконтурным графом $G_n = (O_n, Y)$, где O_n – исходное множество требований. Следующий алгоритм предложен Е. Лоулером.

Обозначим через $N^-(G_n)$ множество всех висячих вершин графа G_n, т.е. множество всех требований, не имеющих потомков в графе G_n. Вычислим момент завершения обслуживания всех требований $P_n = \sum_{j=1}^{n} p_j$. Очевидно, что обслуживание одного из требований множества $N^-(G_n)$ завершается в момент времени P_n. Если этим требованием является требование i, то стоимость его обслуживания равна $f_i(P_n)$. Поскольку функции стоимости являются неубывающими, нетрудно убедиться, что последним обслуживаемым требованием в оптимальной перестановке является такое требование $i \in N^-(G_n)$, что значение $f_i(P_n)$ минимально. Пусть i_n является таким требованием. Тогда момент завершения обслуживания оставшихся требований равен $P_{n-1} = P_n - p_{i_n}$. Построим граф G_{n-1} путем удаления из графа G_n вершины i_n и всех входящих в нее дуг. В графе G_{n-1} определим вершину (требование) i_{n-1} аналогично тому, как была определена вершина i_n. Процесс повторяется до тех пор, пока не будет построена последовательность $(i_1, \ldots, i_n)$, являющаяся оптимальным решением задачи $1/prec/f_{\max}$.

Задание. Решить задачу $1/prec/L_{\max}$, в которой имеется 10 требований. Требование 3 предшествует требованию 4, которое, в свою очередь, предшествует требованиям 1, 7 и 9. Длительности обслуживания p_j и директивные сроки d_j заданы в таблице

j	1	2	3	4	5	6	7	8	9	10
p_j	4	2	3	5	7	4	1	2	9	4
d_j	14	23	17	25	17	14	10	7	9	24

Ответ: $(8, 3, 4, 9, 7, \{1, 6\}, 5, 2, 10)$.

Теорема 2.1. ***Задача*** $1/r_j/C_j \le d_j$ ***является NP-трудной в сильном смысле.***

Задача $1/r_j/C_j \le d_j$ принадлежит классу $\mathcal{NP}$, поскольку представление оптимального расписания требует $O(n)$ единиц памяти и моменты завершения обслуживания требований могут быть вычислены за время $O(n)$.

Далее воспользуемся NP-полной в сильном смысле задачей 3-РАЗБИЕНИЕ: заданы целые положительные числа $a_1, a_2, \ldots, a_{3r}$ и A такие, что $A/4 < a_j < A/2$, $j = 1, \ldots, 3r$, и $\sum_{j=1}^{3r} a_j = rA$. Требуется определить, существует ли разбиение множества $\{1, 2, \ldots, 3r\}$ на r непересекающихся подмножеств $X_1, X_2, \ldots, X_r$ такое, что $\sum_{j \in X_l} a_j = A$, $l = 1, \ldots, r$.

Для любого примера задачи 3-РАЗБИЕНИЕ построим пример задачи $1/r_j/C_j \le d_j$, в котором имеется $4r - 1$ требование: $3r$ *основных* требований с параметрами $p_j = a_j$, $r_j = 0$ и $d_j = rA + (r - 1)$, $j = 1, \ldots, 3r$, и $r - 1$ *вспомогательное* требование с параметрами $p_j = 1$, $r_j = A(j - 3r) + (j - 3r - 1)$ и $d_j = r_j + 1$, $j = 3r + 1, \ldots, 4r - 1$. Покажем, что задача 3-РАЗБИЕНИЕ имеет решение тогда и только тогда, когда существует решение построенного примера задачи $1/r_j/C_j \le d_j$.

Необходимость. Пусть множества $X_1, \ldots, X_r$ представляют решение задачи 3-РАЗБИЕНИЕ. Построим расписание, в котором вспомогательные требования обслуживаются в интервалах $[r_j, d_j]$, $j = 3r + 1, \ldots, 4r - 1$, и основные требования множества X_1 обслуживаются в произвольном порядке перед вспомогательным требованием $3r + 1$, основные требования множества X_l, $l = 2, \ldots, r - 1$, обслуживаются в произвольном порядке между вспомогательными требованиями $3r + l - 1$ и $3r + l$, основные требования множества X_r обслуживаются в произвольном порядке после вспомогательного требования $4r - 1$.

Нетрудно убедиться, что для построенного расписания $C_j \le d_j$ для всех требований.

Достаточность. Предположим, что существует расписание такое, что $C_j \le d_j$ для всех требований. В этом расписании вспомогательные требования обслуживаются в интервалах $[r_j, d_j]$, $j = 3r + 1, \ldots, 4r - 1$. Обозначим через X_1 множество основных требований, обслуживаемых перед требованием $3r + 1$, через X_l, $l = 2, \ldots, r - 1$, множество основных требований, обслуживаемых между требованиями $3r + l - 1$ и $3r + l$, через X_r множество основных требований, обслуживаемых после требования $4r - 1$. Тогда должно выполняться $\sum_{j \in X_l} p_j = \sum_{j \in X_l} a_j \le A$, $l = 1, \ldots, r$, откуда следует, что $\sum_{j \in X_l} a_j = A$, $l = 1, \ldots, r$, т.е. задача 3-РАЗБИЕНИЕ имеет решение. ■

2.2. Один прибор. Суммарный штраф.

Исследуем задачу $1//\sum f_j$ и ее частные случаи.

2.2.1. Вначале рассмотрим задачу $1//\sum w_j C_j$. Целевая функция этой задачи является

приоритето-порождающим. Кроме того, последовательность SWPT, в которой требования упорядочены по неубыванию отношения p_j/w_j, является оптимальной. Доказательство может быть проведено при помощи перестановочного приема.

2.2.2. Рассмотрим задачу $1//\sum w_jU_j$. Вначале докажем, что эта задача NP-трудна.

Теорема 2.2. ***Задача*** $1//\sum w_jU_j$ ***является NP-трудной даже в случае*** $d_j = d$, $j = 1, \ldots, n$.

Задача распознавания, соответствующая задаче $1/d_j = d/\sum w_jU_j$, принадлежит классу $\mathcal{NP}$, поскольку представление оптимального расписания требует $O(n)$ единиц памяти и значение целевой функции от оптимального расписания может быть вычислено за время $O(n)$.

Далее воспользуемся NP-полной задачей РАЗБИЕНИЕ: заданы целые положительные числа $a_1, a_2, \ldots, a_r$ и A такие, что $\sum_{j=1}^{r} a_j = 2A$. Требуется определить, существует ли множество $X \subset N = \{1, 2, \ldots, r\}$ такое, что $\sum_{j\in X} a_j = A$.

Для любого примера этой задачи построим пример задачи $1/d_j = d/\sum w_jU_j$, в котором имеется r требований с параметрами $p_j = a_j$, $w_j = a_j$ и $d_j = d = A$, $j = 1, \ldots, r$. Покажем, что задача РАЗБИЕНИЕ имеет решение тогда и только тогда, когда существует решение построенного примера задачи $1/d_j = d/\sum w_jU_j$ со значением целевой функции, не превосходящим A.

Необходимость. Пусть множество X является решением задачи РАЗБИЕНИЕ. Построим расписание, в котором вначале обслуживаются требования множества X в произвольном порядке и за ними обслуживаются требования множества $N\backslash X$ в произвольном порядке. Поскольку $\sum_{j\in X} a_j = A$, требования множества X являются ранними и требования множества $N\backslash X$ – запаздывающими. Тогда для построенного расписания $\sum w_jU_j = \sum_{j\in N\backslash X} a_j = A$.

Достаточность. Предположим, что существует расписание со значением целевой функции $\sum w_jU_j \le d$. Обозначим через X множество ранних требований в этом расписании. Имеем $\sum_{j\in X} p_j = \sum_{j\in X} a_j \le d = A$ и $\sum_{j\in N\backslash X} w_j = \sum_{j\in N\backslash X} a_j \le A$. Следовательно, $\sum_{j\in X} a_j = A$, т.е. задача РАЗБИЕНИЕ имеет решение. ■

2.2.3. Ниже приводится алгоритм временной сложности $O(n \log n)$ решения задачи $1//\sum U_j$.

Упорядочим требования в порядке EDD неубывания директивных сроков: $d_1 \le \cdots \le d_n$. Будем назначать требования на обслуживание в порядке $1, 2, \ldots$ до тех пор, пока не будет нарушен директивный срок очередного требования. Пусть нарушается директивный срок d_j. Среди требований $1, \ldots, j$ отыскиваем требование с наибольшей длительностью обслуживания, например, требование k и удаляем его из последовательности $(1, \ldots, j)$. Требование k считается запаздывающим. Все запаздывающие требования обслуживаются в произвольном порядке после последнего раннего требования. Если после удаления выбранного требования обслуживание требования j завершается до директивного срока d_j, то переходим к назначению требования $j+1$. Если директивный срок d_j все еще нарушается, удаляем очередное требование с наибольшей длительностью обслуживания. Удаление продолжается до тех пор, пока директивный срок d_j не будет выполняться либо не удалится j. Процедура продолжается до тех пор, пока не будет

построена оптимальная последовательность ранних требований.

Задание. Решить задачу $1//\sum U_j$, в которой имеется 10 требований. Длительности обслуживания p_j и директивные сроки d_j заданы в таблице из предыдущего раздела.

Один из ответов: $(8, 7, 1, 6, 3, 2, 10, 4)$ – последовательность ранних требований, 9 и 5 – запаздывающие требования.

2.3. Параллельные приборы. Максимальный штраф

В данном разделе рассматриваются задачи $P//f_{\max}$ и $P/pmtn/f_{\max}$ и их частные случаи.

2.3.1. Вначале покажем, что задача $P/pmtn/C_{\max}$ (при разрешении прерываний) может быть решена за время $O(n)$.

Вычислим значение $LB = \max\{\max_j\{p_j\}, \sum_{j=1}^{n} p_j/m\}$. Нетрудно убедиться, что $LB \leq C^*_{\max}$, где $C^*_{\max}$ является оптимальным значением общего момента завершения обслуживания.

Оптимальное расписание формируется следующим образом. Назначаем требования в произвольной последовательности, например, в последовательности $1, 2, \ldots, n$ на приборы $1, 2, \ldots, m$ в интервале времени $[0, LB]$, начиная с момента времени ноль. Если на приборе l достигнут момент времени LB и текущее требование j полностью не обслужено, то его обслуживание прерывается и возобновляется на приборе $l + 1$, начиная с момента времени ноль.

Приведенный алгоритм называется алгоритмом Макнотона или правилом обертки (wrap around rule). Очевидно, что построенное с помощью этого алгоритма расписание является допустимым (ни одно требование не обслуживается двумя приборами одновременно) и для этого расписания $C_{\max} = LB$, т.е. оно является оптимальным.

Задание. Построить пример задачи $P/pmtn/C_{\max}$, в котором количество прерываний в оптимальном расписании равно $m - 1$.

2.3.2. Далее докажем, что задача $P//C_{\max}$ (прерывания запрещены) NP-трудна в сильном смысле.

Теорема 2.3. ***Задача*** $P//C_{\max}$ ***является NP-трудной в сильном смысле.***

Воспользуемся NP-полной в сильном смысле задачей 3-РАЗБИЕНИЕ. Для любого примера задачи 3-РАЗБИЕНИЕ построим пример задачи $P//C_{\max}$, в котором имеется $3r$ требований с длительностями обслуживания $p_j = a_j$, $j = 1, \ldots, 3r$, и r приборов. Покажем, что задача 3-РАЗБИЕНИЕ имеет решение тогда и только тогда, когда существует решение построенного примера задачи $P||C_{\max}$ со значением $C_{\max} \leq A$.

Необходимость. Пусть множества $X_1, \ldots, X_r$ представляют решение задачи 3-РАЗБИЕНИЕ. Построим расписание, в котором требования множества X_l, $l = 1, \ldots, r$, обслуживаются в произвольном порядке прибором l. Нетрудно убедиться, что для построенного расписания $C_{\max} = A$.

Достаточность. Предположим, что существует расписание такое, что $C_{\max} \leq A$. Обозначим через X_l множество требований, обслуживаемых прибором l, $l = 1, \ldots, r$. Должны выполнять-

ся неравенства $\sum_{j\in X_l} p_j = \sum_{j\in X_l} a_j \le A$, $l = 1,\dots,r$. Поскольку $\sum_{l=1}^{r}\sum_{j\in X_l} a_j = rA$, имеем $\sum_{j\in X_l} a_j = A$, $l = 1,\dots,r$, т.е. задача 3-РАЗБИЕНИЕ имеет решение. ■

2.3.3. Рассмотрим частные случаи задачи без прерываний. Начнем с задачи $P/in{-}tree, p_j = 1/C_{\max}$. Обозначим через N множество требований, через N^+ – множество требований, не имеющих предшественников, и через $h(j)$ – высоту вершины j, т.е. длину (количество вершин) цепи от вершины j до корня дерева.

Следующий алгоритм H формирует оптимальное расписание для задачи $P/in-tree, p_j = 1/C_{\max}$, которое будем называть H-расписанием. В алгоритме H помечаются некоторые требования. Вначале все требования являются непомеченными.

Пронумеруем интервалы времени единичной длины $1, 2, \dots$. На итерации t алгоритма H определенные требования назначаются на обслуживание приборами в интервале времени t единичной длины. Рассмотрим итерацию t. Найдем прибор l, не задействованный в интервале времени t. Среди непомеченных требований множества N^+ выбираем требование j с наибольшей высотой $h(j)$, назначаем его на обслуживание в интервале t прибором l и помечаем требование j. Процесс назначения требований и их пометки прекращается тогда, когда в интервале времени t все приборы заняты или все требования множества N^+ помечены. В этом случае исключаем из N помеченные требования и, если $N \ne \phi$, переходим к итерации $t+1$, т.е. к назначению требований в интервале $t+1$. Если $N = \phi$, то H-расписание построено.

Временная сложность H-алгоритма равна $O(n)$.

Задание 1. Докажите, что H-алгоритм решает задачу в случае, когда имеется несколько компонент связности, каждая из которых является входящим деревом.

Задание 2. Решите задачу $P/in-tree, p_j = 1/C_{\max}$, в которой $m = 3$, $n = 12$, требования 9,10,11 предшествуют 7, требование 12 предшествует 8, требования 7 и 8 предшествуют 4, требования 3 и 4 предшествуют 1 и требования 5 и 6 предшествуют 2.

Ответ: требования, назначенные на приборы 1,2,3 в интервалах времени 1,2,3,4,5 соответственно - (12,11,10), (9,8,6), (7,5,3), (4,2,·), (1,·,·).

Задание 3. Покажите, как использовать H-алгоритм для решения задачи $P/out-tree, p_j = 1/C_{\max}$. (Изменить ориентацию дуг на противоположную, решить задачу для "in-tree"и рассмотреть полученное расписание от конца к началу).

2.3.4. Перейдем к рассмотрению задачи $P/in-tree, p_j = 1/L_{\max}$. Вначале приведем алгоритм решения задачи $P/in-tree, p_j = 1/C_j \le d_j$. В этом алгоритме используется понятие *λ-расписания*, которое формируется в соответствии с некоторым списком требований $\lambda = (j_1,\dots,j_n)$.

Так же, как в H-алгоритме, будем назначать требования в интервалы времени единичной длины $t = 1, 2, \dots$ и помечать требования списка λ. Вначале все требования являются непомеченными.

На итерации t находим прибор l, не занятый в интервале времени t. Находим первое непо-

меченное требование j списка λ, которое принадлежит множеству N^+. Назначаем j на обслуживание в интервале t прибором l и помечаем j. Процесс назначения требований и их пометки прекращается тогда, когда в интервале времени t все приборы заняты или все требования множества N^+ помечены. В этом случае исключаем из λ и N помеченные требования и, если $N \neq \phi$, переходим к итерации $t+1$, т.е. к назначению требований в интервале $t+1$. Если $N = \phi$, то λ-расписание построено.

Единственное отличие от H-алгоритма заключается в том, что в первую очередь выбираются требования не с наибольшей высотой, а с наименьшим номером в списке λ. Иными словами, H-алгоритм является частным случаем приведенного алгоритма, если в списке λ отсортировать требования но невозрастанию высот.

Для решения задачи $P/in-tree, p_j = 1/C_j \leq d_j$ вначале модифицируем директивные сроки. Обозначим модифицированные директивные сроки через d'_j, $j = 1, \ldots, n$. Положим $d'_r = d_r$ для корня r дерева. На каждом следующем шаге выбираем такое требование j, для которого не найдено d'_j, но для его прямого потомка k значение d'_k найдено. Полагаем $d'_j = \min\{d_j, d'_k - 1\}$. В списке λ требования упорядочиваются по неубыванию модифицированных дирктивных сроков.

Задание. Докажите утверждение о том, что расписание допустимо относительно модифицированных директивных сроков тогда и только тогда, когда оно допустимо относительно исходных директивных сроков.

Теорема 2.4. ***Задача*** $P/in - tree, p_j = 1/C_j \leq d_j$ ***имеет решение тогда и только тогда, когда допустимо относительно директивных сроков*** λ***-расписание, соответствующее списку*** $\lambda = (j_1, \ldots, j_n)$, $d'_{j_1} \leq \cdots \leq d'_{j_n}$.

Задание. Решите задачу $P/in - tree, p_j = 1/C_j \leq d_j$, в которой $m = 3$, $n = 10$, 1 предшествует 2, 2 предшествует 3, 3 и 4 предшествуют 7, 5 и 6 предшествуют 9, 7, 8 и 9 предшествуют 10, $(d_1, \ldots, d_{10}) = (6, 5, 6, 4, 9, 8, 7, 3, 2, 8)$.

Ответ: последовательности требований на приборах 1,2,3 в интервалах времени 1,2,3,4,5 соответственно - (8,6,5), (4,1,9), (2,·,·), (3,·,·), (7,·,·), (10,·,·). Модифицированные директивные сроки: $(d'_1, \ldots, d'_{10}) = (4, 5, 6, 4, 1, 1, 7, 3, 2, 8)$.

Возвратимся к рассмотрению задачи $P/in - tree, p_j = 1/L_{\max}$. Пусть $L^*_{\max}$ обозначает оптимальное значение максимального временного смещения. Покажем, что решение задачи $P/in - tree, p_j = 1/C_j \leq d_j + M$, где M – любое число, удовлетворяющее $M \geq L^*_{\max}$, является решением задачи $P/in - tree, p_j = 1/L_{\max}$, например, $M = \sum p_j$.

Действительно, не существует расписания, допустимого относительно директивных сроков $d_j + \tau$, $j = 1, \ldots, n$, если $\tau < L^*_{\max}$. С другой стороны, λ-расписания, построенные алгоритмом H для директивных сроков $d_j + M$, $j = 1, \ldots, n$, совпадают при любых значениях $M \geq L^*_{\max}$.

2.4. Параллельные приборы. Суммарный штраф

В данном разделе рассматриваются задачи построения оптимальных расписаний для обслу-

живающих систем, состоящих из параллельных приборов. Критерием оптимальности является минимизация суммарного штрафа $\sum f_j$.

2.4.1. Рассмотрим задачу $R//\sum C_j$. В этой задаче длительность обслуживания требования j прибором l равна p_{lj}. Если требование j назначено на прибор l и обслуживается этим прибором k-ым с конца (после него прибор l обслуживает $k-1$ требование), то вклад этого требования в целевую функцию равен kp_{lj}.

Введем 0-1 переменные $x_{j(lk)}$ такие, что $x_{j(lk)} = 1$, если требование j обслуживается прибором l k-ым с конца, и $x_{j(lk)} = 0$, в противном случае. Задача $R//\sum C_j$ сводится к *задаче о назначениях*

$$\sum_j \sum_{(lk)} kp_{lj} x_{j(lk)} \to \min,$$

при условиях

$$\sum_{j=1}^{n} x_{j(lk)} \le 1,\ l = 1, \ldots, m,\ k = 1, \ldots, n,$$

$$\sum_{(l,k)} x_{j(lk)} = 1,\ j = 1, \ldots, n,$$

$$x_{j(lk)} \in \{0, 1\},\ j = 1, \ldots, n,\ l = 1, \ldots, m,\ k = 1, \ldots, n.$$

Известно, что задача о назначениях может быть решена за время $O(n^3)$.

Для решения задачи $Q//\sum C_j$ существует более эффективный алгоритм. В условиях этой задачи, если требование j обслуживается прибором l k-ым с конца, то его вклад в целевую функцию равен kp_j/v_l, где v_l – производительность прибора l. Упорядочим значения k/v_l по неубыванию. Задача сводится к выбору n значений k/v_l и назначению каждому из выбранных значений некоторого требования j таким образом, что сумма произведений $p_j \cdot (k/v_l)$ минимальна. Известно, что для этого достаточно выбрать n наименьших чисел k/v_l и назначать наибольшему из выбранных чисел k/v_l наименьшее значение p_j, второму наибольшему из выбранных чисел k/v_l второе наименьшее значение p_j и так далее. Если значение p_j назначено значению k/v_l, то требование j обслуживается k-ым с конца прибором l. Задача $Q//\sum C_j$ может быть решена за время $O(n \log n)$.

Задание. Решить задачу $Q//\sum C_j$, в которой $m = 3$, $v_1 = 1, v_2 = 2, v_3 = 3$. Количество требований и длительности обслуживания p_j заданы в таблице

j	1	2	3	4	5	6	7	8	9	10
p_j	6	18	24	12	6	12	12	24	6	18

Решение: наименьшие значения $k/v_l = \{1/3, 2/3, 3/3, 4/3, 5/3, 1/2, 2/2, 3/2, 4/2, 1/1, 2/1\}$. Упорядочим 10 наименьших значений k/v_l по неубыванию: $(1/3, 1/2, 2/3, 3/3, 2/2, 1/1, 4/3, 3/2, 5/3, 2/1)$. Упорядочим p_j по невозрастанию: $(p_3, p_8, p_2, p_{10}, p_4, p_6, p_7, p_1, p_5, p_9)$. Оптимальное расписание: требование 3 обслуживается прибором 3 первым с конца, требование 8 обслуживается прибором 2 первым с конца и т.д. Последовательность на приборе 1: (9,6), на приборе 2: (1,4,8), на приборе 3: (5,7,10,2,3).

Оптимальное расписание для задачи $P//\sum C_j$ может быть построено при помощи следующего алгоритма. Упорядочим длительности обслуживания требований по правилу SPT: $p_1 \le \cdots \le p_n$. Будем назначать требования на приборы в последовательности $1, \ldots, n$. Требование j назначается на тот прибор, где его обслуживание завершится раньше.

2.4.2. Перейдем к рассмотрению задач $Q/p_j = p/\sum f_j$ и $Q/p_j = p/f_{\max}$. Поскольку длительности обслуживания одинаковы, момент завершения обслуживания требования j прибором l может быть представлен в виде ip/v_l. При этом $f_j(C_j) = f_j(ip/v_l) := b_{j(il)}$. Как и ранее, введем в рассмотрение 0-1 переменные $x_{j(il)}$ такие, что $x_{j(il)} = 1$, если требование j обслуживается прибором l i-ым по порядку, и $x_{j(il)} = 0$, в противном случае. Задача $Q/p_j = p/\sum f_j$ сводится к задаче о назначениях

$$\sum_j \sum_{(il)} b_{j(il)} x_{j(il)} \to \min,$$

при условиях

$$\sum_{j=1}^{n} x_{j(il)} \le 1,\ l = 1, \ldots, m,\ i = 1, \ldots, n,$$

$$\sum_{(i,l)} x_{j(il)} = 1,\ j = 1, \ldots, n,$$

$$x_{j(il)} \in \{0, 1\},\ j = 1, \ldots, n,\ l = 1, \ldots, m,\ i = 1, \ldots, n.$$

Задача $Q/p_j = p/f_{\max}$ сводится к максиминной задаче о назначениях

$$\max_{j,(il)} b_{j(il)} x_{j(il)} \to \min$$

при тех же условиях. Обе задачи могут быть решены за время $O(n^3)$.

Задание. Постройте сведение задачи $P/r_j, p_j = 1/\sum f_j$ к задаче о назначениях.

2.4.3. Появление весов требований значительно усложняет задачу.

Теорема 2.5. ***Задача $P//\sum w_j C_j$ является NP-трудной в сильном смысле.***

Доказательство. Воспользуемся NP-полной в сильном смысле задачей 3-РАЗБИЕНИЕ. Для любого примера задачи 3-РАЗБИЕНИЕ построим пример задачи $P//\sum w_j C_j$, в котором имеется $3r$ требований с длительностями обслуживания $p_j = a_j$ и весами $w_j = a_j$, $j = 1, \ldots, r$, и r приборов. Покажем, что задача 3-РАЗБИЕНИЕ имеет решение тогда и только тогда, когда существует решение построенного примера задачи $P//\sum w_j C_j$ со значением $\sum w_j C_j \le y = \sum_{j=1}^{3r} a_j^2/2 + mA^2/2$.

Необходимость. Пусть множества $X_1, \ldots, X_r$ представляют решение задачи 3-РАЗБИЕНИЕ. Построим расписание, в котором требования множества X_l, $l = 1, \ldots, r$, обслуживаются в произвольном порядке прибором l. Пусть $(i_1, \ldots, i_k)$ – последовательность требований на приборе l. Тогда

$$\sum_{j \in X_l} w_j C_j = a_{i_1}^2 + a_{i_2}(a_{i_1} + a_{i_2}) + \cdots$$

$$+ a_{i_k}(a_{i_1} + \cdots + a_{i_k}) =$$

$$\sum_{j\in X_l} a_j^2 + \sum_{1\le\nu<\mu\le k} a_{i_\nu} a_{i_\mu}.$$

Поскольку

$$(\sum_{j\in X_l} a_j)^2 = \sum_{j\in X_l} a_j^2 + 2\sum_{1\le\nu<\mu\le k} a_{i_\nu} a_{i_\mu},$$

имеем

$$\sum_{1\le\nu<\mu\le k} a_{i_\nu} a_{i_\mu} = (\sum_{j\in X_l} a_j)^2/2 - \sum_{j\in X_l} a_j^2/2.$$

Поэтому

$$\sum_{j\in X_l} w_j C_j = \sum_{j\in X_l} a_j^2/2 + (\sum_{j\in X_l} a_j)^2/2.$$

Для построенного расписания

$$\sum w_j C_j = \sum_{l=1}^{m}\sum_{j\in X_l} w_j C_j = \sum_{j=1}^{3r} a_j^2/2 + \sum_{l=1}^{m}(\sum_{j\in X_l} a_j)^2/2 = \sum_{j=1}^{3r} a_j^2/2 + mA^2/2 = y. \qquad (2.1)$$

Достаточность. Предположим, что существует расписание такое, что $\sum w_j C_j \le y$. Обозначим через X_l множество требований, обслуживаемых прибором l, $l = 1, \ldots, r$. Аналогично тому, как была получена формула (2.1), получаем

$$\sum w_j C_j = \sum_{j=1}^{3r} a_j^2/2 + \sum_{l=1}^{m}(\sum_{j\in X_l} a_j)^2/2 \le y = \sum_{j=1}^{3r} a_j^2/2 + mA^2/2.$$

Из последнего неравенства следует, что

$$\sum_{l=1}^{m}(\sum_{j\in X_l} a_j)^2 \le mA^2.$$

Предположим, что $\sum_{j\in X_l} a_j = A + \delta_l$, где $-A \le \delta_l \le (m-1)A$, $l = 1, \ldots, m$, и $\sum_{l=1}^{m} \delta_l = 0$. Тогда

$$\sum_{l=1}^{m}(\sum_{j\in X_l} a_j)^2 = \sum_{l=1}^{m}(A+\delta_l)^2 = mA^2 + 2A\sum_{l=1}^{m}\delta_l + \sum_{l=1}^{m}\delta_l^2 = mA^2 + \sum_{l=1}^{m}\delta_l^2.$$

Нетрудно заметить, что

$$\sum_{l=1}^{m}(\sum_{j\in X_l} a_j)^2 \le mA^2$$

тогда и только тогда, когда $\delta_l = 0$, $l = 1, \ldots, m$. Следовательно, $\sum_{j\in X_l} a_j = A$, $l = 1, \ldots, r$, т.е. задача 3-РАЗБИЕНИЕ имеет решение. ■

Глава 3

Многостадийные системы обслуживания

В данном разделе рассматриваются обслуживающие системы flow-, open- и job-shop.

3.1. Обслуживающая система flow-shop

В системе flow-shop каждое требование обслуживается приборами $1, 2, \ldots, m$ в этой последовательности. Обслуживание требования прибором $l+1$ может быть начато не ранее завершения его обслуживания прибором l.

Рассмотрим задачу $F//C_{\max}$.

Теорема 3.1. ***Для задачи*** $F//C_{\max}$ ***существует оптимальное расписание, при котором приборы 1 и 2 обслуживают требования в одной и той же последовательности.***

Доказательство. При заданном расписании S обозначим через $C^0_{lj}(S)$ и $C_{lj}(S)$ моменты начала и завершения обслуживания требования j прибором l соответственно.

Пусть S – расписание, для которого утверждение теоремы не выполняется. При этом расписании прибор 1 обслуживает требования в последовательности $(i_1, \ldots, i_p, \ldots, i_r, i_{r+1}, \ldots, i_n)$, а прибор 2 – в последовательности $(i_1, \ldots, i_p, i_{r+1}, \pi_1, i_r, \pi_2)$, где $\pi_1 = (j_1, \ldots, j_v)$ и π_2 – некоторые подпоследовательности. Для расписания S выполняются соотношения

$$C^0_{2i_{r+1}}(S) = \max\{\sum_{k=1}^{r+1} p_{1i_k}, C_{2i_p}(S)\},$$

$$C^0_{2i_r}(S) = \max\{\sum_{k=1}^{r} p_{1i_k}, C_{2j_v}(S)\} = C_{2j_v}(S) \geq$$

$$C_{2i_{r+1}}(S) \geq C_{1i_{r+1}}(S) = \sum_{k=1}^{r+1} p_{1i_k}.$$

Построим расписание S', отличающееся от S тем, что на первом приборе требования i_r и i_{r+1} обслуживаются в последовательности i_{r+1}, i_r. Для расписания S' выполняются соотношения

$$C^0_{2i_{r+1}}(S') = \max\{\sum_{k=1}^{r-1} p_{1i_k} + p_{1i_{r+1}}, C_{2i_p}(S')\},$$

$$C^0_{2i_r}(S) = \max\{\sum_{k=1}^{r+1} p_{1i_k}, C_{2j_v}(S')\}.$$

Поскольку $C_{2i_p}(S') = C_{2i_p}(S)$, то $C^0_{2i_{r+1}}(S') \le C^0_{2i_{r+1}}(S)$. Из этого следует $C_{2j_v}(S') \le C_{2j_v}(S)$ и далее $C^0_{2i_r}(S') \le C^0_{2i_r}(S)$.

Применяя приведенные рассуждения необходимое количество раз (не более $O(n^2)$), получаем расписание S'', при котором приборы 1 и 2 обслуживают требования в одной и той же последовательности, причем выполняются соотношения $C^0_{2i}(S'') \le C^0_{2i}(S)$, $i = 1, \ldots, n$. Поэтому $C_{\max}(S'') \le C_{\max}(S)$. ■

Задание. Используя перестановочный прием, докажите следующую теорему.

Теорема 3.2. ***Для задачи*** $F//C_{\max}$ ***существует оптимальное расписание, при котором приборы*** $m-1$ ***и*** m ***обслуживают требования в одной и той же последовательности.***

Из двух последних теорем следует, что при $m \le 3$ оптимальное расписание для задачи $F//C_{\max}$ можно отыскивать в классе *перестановочных расписаний*, то есть расписаний, при которых все приборы обслуживают требования в одной и той же последовательности. При $m \ge 4$ это утверждение не справедливо.

Перейдем к рассмотрению задачи $F2//C_{\max}$. Обозначим $a_j = p_{1j}$ и $b_j = p_{2j}$.

Алгоритм:

Сформируем два множества $N_1 = \{j | a_j \le b_j\}$ и $N_2 = \{j | a_j > b_j\}$. В оптимальной последовательности вначале обслуживаются требования множества N_1 в порядке неубывания значений a_j, а затем требования множества N_2 в порядке невозрастания b_j.

Задание. Решить задачу $F2//C_{\max}$, в которой имеется 10 требований. Построить диаграмму Ганнта для оптимального расписания. Длительности обслуживания a_j и b_j заданы в таблице

j	1	2	3	4	5	6	7	8	9	10
a_j	4	2	3	5	17	40	10	20	9	4
b_j	4	3	17	25	7	14	10	7	9	24

Ответ: $(2, 3, \{1, 10\}, 4, 9, 6, \{5, 8\})$. Множество $N_2 = \{5, 6, 8\}$.

3.2. Обслуживающая система open-shop

В системе open-shop каждое требование обслуживается приборами $1, 2, \ldots, m$ в произвольной последовательности. Напомним, что в любой момент времени каждое требование обслуживается не более чем одним прибором и любой прибор обслуживает не более одного требования.

Задачи для системы open-shop обладают симметрией – приборы и требования можно поменять ролями без изменения существа задачи.

3.2.1. Рассмотрим задачу $O2//C_{\max}$. Как и ранее, обозначим $a_j = p_{1j}$ и $b_j = p_{2j}$.

Для любого расписания справедлива следующая оценка снизу значения $C_{\max}$:

$$C_{\max} \ge \max\{\sum_j a_j, \sum_j b_j, \max_j\{a_j + b_j\}\}.$$

Построим расписание, для которого приведенная оценка достижима. Следовательно, такое расписание является оптимальным.

Алгоритм:

Назначаем на первый освободившийся прибор требование с максимальной длительностью обслуживания на другом приборе. При этом завершившие обслуживание на другом приборе обслуживаются на данном приборе последними в произвольном порядке. Начинаем с назначения требования с максимальной длительностью обслуживания на каком-либо приборе.

Задание. Каждое из 6 транспортных средств должно выполнить два типа работ A и B в любом порядке. Требуется построить расписание, минимизирующее момент завершения последней работы. Длительности a_j и b_j выполнения работ типа A и B соответственно заданы в таблице

j	1	2	3	4	5	6
a_j	4	12	4	6	6	12
b_j	8	6	2	10	8	8

Ответ: $\pi_A = (4, 1, 5, 3, 6, 2)$, $\pi_B = (2, 4, 1, 5, 3, 6)$, $C_{\max} = 44$.

3.2.2. Рассмотрим задачу $O/p_j = 1/\sum C_j$. Для ее решение необходимо ввести понятие *латинского квадрата (прямоугольника).*

Квадратная (прямоугольная) матрица $||\lambda_{lj}||$, $\lambda_{lj} \in N = \{1, \ldots, n\}$, называется латинским квадратом (прямоугольником), построенном на множестве N, если каждая ее строка представляет собой перестановку элементов множества N и каждый элемент N встречается равно один раз (не более одного раза) в каждом столбце матрицы.

Например, матрица с элементами $\lambda_{1j} = j$, $j = 1, \ldots, n$, $\lambda_{l1} = \lambda_{l-1,n}$, $\lambda_{l,j} = \lambda_{l-1,j-1}$, $j = 2, \ldots, n$, $l = 2, \ldots, m$, такая как приведенная ниже

1	2	3	4	5	6	7
7	1	2	3	4	5	6
6	7	1	2	3	4	5

является латинским прямоугольником.

Пусть $n = km + r$, $k \geq 0$, $0 \leq r < m$, для некоторых k и r. В оптимальном расписании для задачи $O/p_j = 1/\sum C_j$ каждый из m приборов выполняет km операций в интервале $(0, km]$ и, дополнительно, каждый из r приборов выполняет m операций в интервале $(km, (k+1)m]$.

Для такого расписания оценим снизу значение целевой функции:

$$\sum C_j \geq (k+1)mr + m^2k(k+1)/2.$$

Построим (оптимальное) расписание, для которого приведенная оценка достижима.

Дополним множество требований $m - r$ фиктивными требованиями $n+1, \ldots, (k+1)m$. Разобъем новое множество требований на $k+1$ наборов по m требований, начиная с первого. Набор N_j включает требования $(j-1)m + 1, \ldots, jm$. Для каждого набора N_i построим соответствующий латинский квадрат Λ_i, располагая элементы первой строки по возрастанию. Построим

матрицу Λ путем "склеивания" латинских квадратов Λ_i в порядке возрастания их номеров. Полученная матрица Λ соответствует оптимальному расписанию S_1^* для задачи $O/p_j = 1/\sum C_j$. При этом элемент матрицы Λ, расположенный в столбце t и строке l определяет требование, обслуживаемое прибором l в единичном интервале t.

Незначительная модификация алгоритма построения расписания S_1^* позволяет решать задачу $O/p_j = 1/\sum w_j C_j$. Единственное отличие заключается в том, что перед применением алгоритма требования необходимо пронумеровать так, чтобы $w_1 \geq \cdots \geq w_n$.

Покажем, как построить расписание S_2^* оптимальное одновременно для задач $O3/p_j = 1/\sum C_j$ и $O3/p_j = 1/C_{\max}$.

Предположим, что наборы N_i, $i = 1, \ldots, k+1$, построены. Из набора N_k переместим $m - r$ наибольших элементов в набор N_{k+1} и из набора N_{k+1} переместим $m - r$ фиктивных элементов в набор N_k. Оставим для наборов прежние обозначения. Для каждого набора N_i построим соответствующий латинский квадрат Λ_i, располагая элементы первой строки по возрастанию. Построим матрицу Λ путем "склеивания" латинских квадратов Λ_i в порядке возрастания их номеров. Далее в матрице Λ каждое фиктивное требование j поменяем местами с требованием, расположенным в той же строке и в столбце j. При этом все фиктивные требования оказываются в последних столбцах.

Полученная матрица Λ' соответствует расписанию S_2^* оптимальному для задач $O/p_j = 1/\sum C_j$ и $O/p_j = 1/C_{\max}$.

Задание. Каждый из 8 студентов должен пройти медицинский осмотр в поликлинике у трех врачей A, B и C в любом порядке. Все студенты пришли в поликлинику одновременно (в момент времени ноль) и каждый студент покидает поликлинику сразу после завершения осмотра. Длительность осмотра любого студента любым врачом равна 15 минутам. Требуется построить расписание, минимизирующее среднее время пребывания студентов в поликлинике и момент завершения осмотра последнего студента.

Ответ:

1	2	3	4	5	8	6	7
3	1	2	7	4	5	8	6
2	3	1	5	6	4	7	8

$\frac{1}{8}\sum C_j = \frac{3\cdot 3+2\cdot 6+3\cdot 8}{8} \cdot 15$ минут (около 84), $C_{\max} = 8 \cdot 15 = 120$ минут.

3.3. Обслуживающая система job-shop

В системе job-shop для каждого требования j задана последовательность приборов $(l_1(j), \ldots, l_{r_j}(j))$, которые его обслуживают. В этой последовательности приборы могут повторяться. Процесс обслуживания требования j прибором $l_i(j)$ называется i-й стадией обслуживания требования j. Обозначим через p_{ij}, $i = 1, \ldots, r_j$, длительность обслуживания j на стадии i.

Задача $J/pmtn/f$ в случае, когда имеется два требования и целевая функция $f =$

$f(C_1, C_2)$ **является регулярной, т.е. неубывающей от** C_1 **и** C_2. В этом случае допустимое расписание задачи может быть представлено графически.

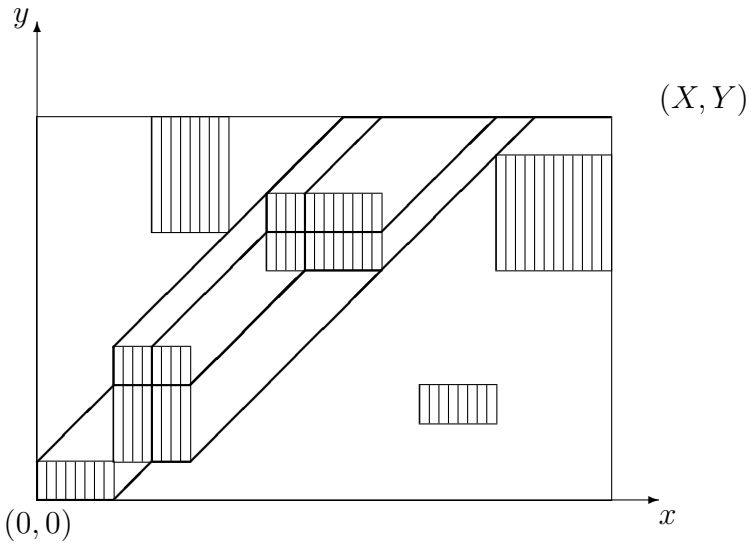

Рис. 1. Графическое решение задач $J//f$ и $J/pmtn/f$ при $n = 2$.

Введем в рассмотрение прямоугольную систему координат. Отложим на оси $(0, x)$ точку $X = \sum_{i=1}^{r_1} p_{i1}$ и на оси $(0, y)$ точку $Y = \sum_{i=1}^{r_2} p_{i2}$. Обозначим прямоугольник, определяемый точками $(0, 0)$ и (X, Y) через W. В этом прямоугольнике определим длину отрезка, соединяющего любые две точки (x, y) и (x', y') как $\rho((x, y), (x', y')) = \max\{|x' - x|, |y' - y|\}$, а длину ломаной линии – как сумму длин составляющих ее отрезков.

Расписание будем представлять в виде непрерывной ломаной линии (траектории), соединяющей точки $(0, 0)$ и (X, Y). Для каждой точки (x, y) траектории, величина x (y) равна суммарному времени работы приборов по обслуживанию требования 1 (2), начиная с момента времени ноль. Составляющими траектории могут быть отрезки трех типов: горизонтальные (обслуживается только требование 1), вертикальные (обслуживается только требование 2) и наклонные под углом 45 градусов (обслуживаются оба требования).

Пусть x_i (y_i) обозначает сумму длительностей обслуживания требования 1 (2) до стадии i включительно. В прямоугольнике W введем в рассмотрение так называемые *запрещенные* прямоугольники с вершинами (x_{u-1}, y_{v-1}) (юго-западный угол), (x_u, y_{v-1}) (юго-восточный угол), (x_{u-1}, y_v) (северо-западный угол) и (x_u, y_v) (северо-восточный угол) такими, что один и тот прибор используется требованием 1 на стадии u (в интервале $(x_{u-1}, x_u]$) и требованием 2 на стадии v (в интервале $(y_{v-1}, y_v]$). Внутренние точки запрещенных прямоугольников соответствуют одновременному обслуживанию требований 1 и 2 одним и тем же прибором.

Поскольку любой прибор в каждый момент времени может обслуживать не более одного требования, то наклонные отрезки *допустимой* траектории не могут находиться внутри запрещенных прямоугольников. В случае запрещения прерываний, любые отрезки допустимой траектории не должны находиться внутри запрещенных прямоугольников.

Алгоритм решения задачи $J/pmtn/f(C_1, C_2)$:

Построим ориентированный граф $G = (Q, R)$ следующим образом. Вначале считаем, что множество Q включает лишь точку $(0, 0)$. Для каждой вершины $(x, y) \in Q$ выполняем следующие действия. Из (x, y) проводим дугу под углом 45 градусов до ее пересечения с границей запрещенного прямоугольника или с границей прямоугольника W. Точку пересечения включаем в Q, а полученную дугу в R. Если указанную дугу провести невозможно, то из точки (x, y) проводим две дуги – вертикальную и горизонтальную или, если точка на границе прямоугольника W, – одну дугу, до тех точек, откуда возможно провести дугу под углом 45 градусов, либо до точки (X, Y). Соответствующие дуги и их конечные точки включаем в G.

Задача отыскания пути из вершины $(0, 0)$ в вершину (X, Y), соответствующего оптимальному расписанию для задачи $J/pmtn/f(C_1, C_2)$, сводится к следующему.

1) Построению графа G и выделению всех вершин вида (X, y) и (x, Y).

2) Нахождению путей наименьшей длины из вершины $(0, 0)$ в каждую из выделенных вершин. Пусть $C(x, y)$ – длина кратчайшего пути $\sigma(x, y)$ из вершины $(0, 0)$ в выделенную вершину (x, y). Обозначим через $S(x, y)$ расписание, соответствующее пути, являющемуся объединением кратчайшего пути $\sigma(x, y)$ и дуги, соединяющей (x, y) с (X, Y).

3) Вычислению для каждого расписания $S(x, y)$, где (x, y) – выделенная вершина, соответствующих значений C_1 и C_2 по формулам:

$C_1 = C(x, y)$, $C_2 = C(x, y) + Y - y$, если $x = X$, $y \neq Y$,

$C_2 = C(x, y)$, $C_1 = C(x, y) + X - x$, если $y = Y$, $x \neq X$,

$C_1 = C_2 = C(x, y)$, если $x = X$, $y = Y$.

4) Вычислению для всех расписаний $S(x, y)$, где (x, y) – выделенная вершина, соответствующих значений $f(C_1, C_2)$ и выбору среди них наименьшего, которое и определяет оптимальное расписание для задачи $J/pmtn/f(C_1, C_2)$.

Задание. Решить задачу $J/pmtn/30C_1 + C_2^2$ в случае, когда (A, B, C, D, A) является последовательностью приборов для требования 1 и (A, B, A, C, A, C) – последовательностью приборов для требования 2. Последовательности длительностей обслуживания равны $(2, 2, 8, 3, 6)$ и $(1, 3, 2, 4, 4, 2)$ соответственно.

Ответ: оптимальный путь

$$\sigma^* = ((0, 0), (2, 0), (3, 1), (4, 1), (9, 6), (9, 10), (15, 16), (21, 16)),$$

$C_1^* = 19 + (21 - 15) = 25$, $C_2^* = 19$, $f(C_1^*, C_2^*) = 30 \cdot 25 + 19^2 = 1111$.

Задача $J//f(C_1, C_2)$ в случае, когда прерывания запрещены. Как и в предыдущем случае, задача сводится к отысканию пути минимальной длины в графе, построенном в прямоугольнике W. Однако в этом случае допустимая траектория может проходить лишь вне запрещенных прямоугольников или по их границам так, что соответствующая сторона прямоугольника целиком принадлежит траектории.

Для задачи $J//f(C_1, C_2)$ соответствующий граф $G' = (Q', R')$ строится следующим образом. Вначале считаем, что множество Q' включает лишь точку $(0, 0)$ и, если $(0, 0)$ является (юго-

западной) вершиной запрещенного прямоугольника, северо-западный и юго-восточный углы этого прямоугольника. Для каждой вершины $(x, y) \in Q'$ выполняем следующие действия. Если эта вершина принадлежит границе прямоугольника W, то включаем в Q' точку (X, Y). Иначе из (x, y) проводим дугу под углом 45 градусов до ее пересечения с границей запрещенного прямоугольника или с границей прямоугольника W. Включаем в Q' северо-западный и юго-восточный углы этого прямоугольника или точку (X, Y) (во втором случае). Дуги, соединяющие точку (x, y) и точки, только что включенные в Q', включаем в R'.

В полученном графе заменяем наклонные дуги с углом наклона, отличным от 45 градусов, двумя дугами, одна из которых имеет угол наклона 45 градусов, а вторая является горизонтальной или вертикальной. Эти новые две дуги соединяют те же точки, что и исходная дуга.

Далее решение задачи $J//f(C_1, C_2)$ сводится к выполнению действий 1), 2), 3) и 4), приведенных для решения задачи $J/pmtn/f(C_1, C_2)$.

Глава 4

Групповые технологии обслуживания

В данном разделе изучаются задачи построения оптимальных расписаний обслуживания требований партиями.

В разд. 4.1 приводятся наиболее общие постановки задач обслуживания требований партиями, дается обзор существующих типов обслуживающих систем и отмечаются практические ситуации, в которых возникают соответствующие задачи. В разд. 4.2 вводятся необходимые обозначения. В разд. 4.3 рассматривается ряд характерных задач и алгоритмов их решения.

4.1. Постановка задач

Многие современные производственные системы обслуживают схожие либо идентичные требования партиями. Требования одной и той же партии обслуживаются прибором непосредственно друг за другом либо одновременно. Такая технология обслуживания приводит к значительному снижению затрат на переналадки.

4.1.1. Общая постановка задачи обслуживания требований партиями состоит в следующем. Имеется множество, состоящее из n требований, которое разбито на g попарно непересекающихся *групп*, исходя из некоторых технологических соображений. Эти требования должны быть объединены в *партии* и обслужены m приборами. Образование партий – это разбиение множества требований на попарно непересекающиеся подмножества, каждое из которых может включать лишь требования одной и той же группы. Партия – это подмножество в таком разбиении. Требования одной и той же партии обслуживаются прибором непосредственно друг за другом либо одновременно. Предполагается, что разбиение множества требований на партии одинаково для всех приборов. Кроме того, обслуживание требования прибором происходит *без прерываний*. Такие предположения являются типичными для большинства практических ситуаций обслуживания требований партиями, и известно лишь незначительное число результатов для задач, где допускается формирование различных партий на разных приборах либо допускаются прерывания обслуживания требований.

Непосредственно перед началом обслуживания каждой партии необходима *переналадка* прибора. Возможны два типа переналадок. Если переналадка может начаться лишь после того, как хотя бы одно требование соответствующей партии поступило на прибор, то переналадка явля-

ется *неупреждающей*. В противном случае она является *упреждающей*. В дальнейшем, если не оговорено особо, предполагается, что все переналадки являются упреждающими.

Каждый прибор может обслуживать не более одной партии в каждый момент времени и не может обслуживать требования во время выполнения переналадки. Каждое требование может обслуживаться не более чем одним прибором в каждый момент времени, если не оговорено обратное.

Под *моментом завершения обслуживания* требования прибором понимается момент времени, когда требование покидает данный прибор. Этот момент, вообще говоря, отличается от момента, когда прибор завершает работу по обслуживанию данного требования, так как требование может задержаться на приборе в силу ряда обстоятельств. Например, если требования – это детали, размещенные на палете и обрабатываемые некоторым станком, то детали удаляются со станка вместе с палетой. Поэтому после завершения обработки данной детали до ее удаления из системы может пройти некоторое время, необходимое для завершения обработки всех остальных деталей, размещенных на той же палете.

На множестве групп, также как и на множестве требований, может быть определено отношение предшествования (порядка). Под моментом завершения обслуживания группы понимается момент завершения обслуживания всех ее требований.

Предполагается, что группа J состоит из q_J требований, $J = 1, \ldots, g$. Длительность переналадки для партии группы J равна $s_{IJ}^{(l)}$, если эта партия обслуживается непосредственно после партии группы I на приборе l. Если партия группы J является первой партией, обслуживаемой прибором l, то длительность соответствующей переналадки (наладки) равна $s_{0J}^{(l)}$. Предполагается, что длительности переналадок удовлетворяют *неравенству треугольника* для каждого прибора, т.е. $s_{IJ}^{(l)} \le s_{IK}^{(l)} + s_{KJ}^{(l)}$, $l = 1, \ldots, m$, для всех групп I, K и J, включая случай $I = 0$.

Переналадки называются *не зависящими от прибора*, если $s_{IJ}^{(l)} = s_{IJ}$ для $l = 1, \ldots, m$, и *зависящими от прибора* в противном случае. Переналадки называются *не зависящими от последовательности*, если $s_{IJ}^{(l)} = s_J^{(l)}$ для $I, J = 0, 1, \ldots, g$, $l = 1, \ldots, m$, и *зависящими от последовательности* в противном случае.

Под *расписанием* понимается пара “разбиение – функция”. Здесь разбиение – это разбиение множества требований на партии, а функция – это отображение, которое каждому прибору l и моменту времени t сопоставляет требования партии, обслуживаемой прибором l в момент времени t, либо указывает, что прибор l в момент t простаивает или выполняется его переналадка. Для ряда задач расписание полностью определяется разбиением множества требований на партии, последовательностями этих партий на приборах и порядком обслуживания требований каждой партии.

Критерием оптимальности расписания является соблюдение требованиями заданных директивных сроков, либо минимизация некоторой функции стоимости, зависящей от моментов завершения обслуживания требований.

4.1.2. Задачи построения оптимальных расписаний обслуживания требований партиями могут быть условно разделены на несколько классов, вообще говоря, пересекающихся, в зави-

симости от того, покидают ли требования одной и той же партии прибор одновременно или нет, как происходит обслуживание партии и имеется ли ограничение на максимальный размер партии. Изучаются следующие классы задач.

Индивидуальное завершение обслуживания. В задачах этого класса предполагается, что требование покидает прибор в момент, когда прибор завершает работу по его обслуживанию.

Отдельно рассматривается важный подкласс таких задач, в которых предполагается, что все требования одной и той же группы всегда образуют одну партию (см. ниже класс задач "фиксированные партии"). Следует отметить, что в общем случае обслуживание требований каждой группы единой партией не всегда является наилучшей стратегией решения задачи. Разбиение групп часто приводит к существенному улучшению качества обслуживания. Например, рассмотрим линию для разлива безалкогольных напитков, где в соответствии с заявками заказчиков заполняются различные типы бутылок. Предполагается, что заполненные бутылки сразу же покидают линию разлива. Для переключения линии с одного типа бутылок на другой требуется переналадка (в силу различных размеров, объема, расположения этикетки и т.д.). Бутылки могут рассматриваться как требования, а заявки, включающие бутылки одного типа, как группы требований. В напряженный летний период может возникнуть ситуация, когда все заявки не могут быть выполнены в запланированные сроки. Однако если продукция не поступит на рынок вовремя, это приведет к значительному снижению количества продаж. Если заполнение бутылок по заявке некоторого заказчика производится до тех пор, пока не будет выполнена вся заявка, то другие заказчики могут исчерпать запасы в ожидании поставки. Качество обслуживания заказчиков значительно улучшится, если часть заявок будет произведена в ближайшем будущем с целью покрытия немедленных запросов, и оставшаяся часть будет удовлетворена в более позднее время. Таким образом, разбиение групп на партии приводит к лучшему расписанию с точки зрения качества обслуживания.

Еще одним примером практической ситуации, в которой возникает задача рассматриваемого класса, является трансляция набора заданий для ЭВМ, написанных на различных языках программирования. В этом случае задания, написанные на одном и том же языке, образуют группу. Длительность переналадки связана с загрузкой соответствующего транслятора. В случае однопроцессорной вычислительной системы задача состоит в определении последовательности, в которой необходимо загружать трансляторы и выполнять соответствующие задания.

Фиксированные партии. Это подкласс класса задач с индивидуальным завершением обслуживания требований, который характеризуется дополнительным ограничением, что требования одной и той же группы всегда образуют одну партию.

В качестве примера рассмотрим линию для производства цветного пластика, где должен быть выполнен ряд заявок, включающих пластики различных цветовых оттенков. Эти заявки могут быть разбиты на основные цветовые группы такие, как красные, синие и т.д. Длительности переналадок между цветами одной и той же группы являются небольшими, поскольку для производства естественно переходить от более светлых к более темным оттенкам. Однако более

значительные переналадки необходимы при переключении на новую цветовую группу, так как между цветами различных групп необходима тщательная очистка производственной линии. Поэтому максимальная эффективность работы линии достигается при непрерывном производстве пластика, относящегося к одной и той же цветовой группе, с последующим переходом к группе другого цвета. Таким образом, разбиение групп на более мелкие партии не происходит, т.е. каждая группа образует одну партию.

Одновременное завершение обслуживания. В задачах данного класса предполагается, что все требования партии покидают прибор одновременно в момент, когда прибор завершает работу по обслуживанию всех требований партии. Таким образом, момент завершения обслуживания требования некоторой партии прибором совпадает с моментом, когда прибор завершает работу по обслуживанию "последнего" требования этой партии.

Задачи данного класса возникают при планировании работы производственных систем, в которых изделия перемещаются между обрабатывающими устройствами в контейнерах таких, как ящики, палеты или тележки. Множество изделий, назначенных в один и тот же контейнер, рассматривается как партия. Устройство не начинает обслуживание новой партии до тех пор, пока не завершено обслуживание предыдущей партии. Все изделия одной и той же партии покидают устройство одновременно в момент завершения обработки последнего изделия этой партии. Переналадка необходима для удаления предшествующей партии и установки новой.

Последовательное обслуживание. В задачах этого класса предполагается, что требования каждой партии обслуживаются последовательно, и длительность обслуживания партии равна сумме длительностей обслуживания входящих в нее требований.

Параллельное обслуживание. Требования одной и той же партии обслуживаются прибором одновременно (параллельно), и длительность обслуживания партии равна максимуму из длительностей обслуживания входящих в нее требований.

Последовательное обслуживание требований является естественным для систем, в которых прибор не может обслуживать более одного требования в каждый момент времени. Печи и химические ванны являются примерами приборов, которые способны обслуживать несколько требований одновременно. Если требования обслуживаются такими приборами партиями, как это, например, происходит при фотолитографической обработке интегральных схем либо при их температурных испытаниях, то длительность обслуживания партии равна максимальной из длительностей обслуживания требований этой партии.

Неограниченный размер партий. Каждая партия может включать произвольное количество требований.

Ограниченный размер партий. Каждая партия может включать не более чем b, $b < n$, требований.

Задачи с ограниченными размерами партий возникают, когда верхний предел размера партии не может быть превышен по технологическим соображениям, например, в силу ограниченной емкости контейнеров. Размеры партий не ограничены, если емкость контейнеров не является узким местом.

Для систем flow- и open-shop партия может обслуживаться одновременно несколькими приборами, если в соответствующей задаче каждое требование покидает прибор в момент, когда прибор завершает работу по его обслуживанию. Партия не может обслуживаться одновременно несколькими приборами, если все ее требования покидают прибор одновременно.

Если не оговорено особо, порядок обслуживания требований каждой партии может быть различным на различных приборах.

4.2. Обозначения

Для представления задач построения оптимальных расписаний обслуживания требований партиями будем использовать с соответствующей адаптацией традиционное для теории расписаний обозначение $\alpha/\beta/\gamma$, состоящее из трех полей.

Первое поле $\alpha = \alpha_1\alpha_2$ описывает обслуживающую систему. Как и ранее предполагается, что $\alpha_1 \in \{\cdot, P, Q, R, O, F\}$, где $\cdot$ обозначает пустой символ. Символ α_2 служит для описания количества приборов.

Характеристики партий и требований описываются во втором поле. В этом поле могут присутствовать следующие обозначения:

GT : требования каждой группы всегда образуют одну партию и обслуживаются последовательно (фиксированные партии);

$sum{-}job$: требования обслуживаются последовательно и момент завершения обслуживания требования прибором (момент, когда требование покидает прибор) совпадает с моментом, когда прибор прекращает работу по его обслуживанию (индивидуальное завершение обслуживания);

$sum{-}batch$: требования обслуживаются последовательно и момент завершения обслуживания требования любой партии прибором совпадает с моментом, когда прибор завершает работу по обслуживанию всех требований этой партии (одновременное завершение обслуживания);

$max{-}batch$: отличается от предыдущего тем, что требования каждой партии обслуживаются параллельно, и длительность обслуживания партии равна максимальной из длительностей обслуживания составляющих ее требований (параллельное обслуживание);

$b{=}n$: верхний предел для размера партий отсутствует, каждая партия может включать до n требований (неограниченный размер партий);

$b{<}n$: каждая партия может включать не более чем b, $b < n$, требований (ограниченный размер партий);

$b{=}a$: каждая партия может включать не более чем a требований, где a – заданная фиксированная величина (например, $b{=}2$);

$g{=}c$: количество групп фиксировано и равно c (иначе количество групп произвольно) ;

$s_{IJ}^{(l)}, s_{IJ}, s_J^{(l)}, s_J, s_J = s$: переналадки зависят от прибора и последовательности групп, не зависят от прибора, зависят от прибора и не зависят от последовательности, не зависят от прибора и последовательности и одинаковые соответственно.

Если не оговорено особо, то все переналадки являются упреждающими.

$q_J{=}q$: каждая группа включает одно и то же количество требований q (иначе количества

требований различны);

p_J, d_J, w_J : соответствующие параметры p_j, d_j и w_j одинаковы и равны p_J, d_J и w_J для всех требований группы J, $J = 1, \ldots, g$.

Для отдельных задач могут быть использованы некоторые другие легко интерпретируемые обозначения. Необходимые пояснения даются по ходу изложения.

Третье поле, γ, описывает критерий задачи.

4.2.1. Ниже описываются несколько примеров, показывающих, как используются приведенные обозначения.

Задача $1/GT, s_J, prec/\sum w_j C_j$: имеется один прибор, несколько групп требований, каждая из которых должна образовывать одну партию; на множестве требований и на множестве групп определены отношения предшествования, каждое требование партии имеет индивидуальный момент завершения обслуживания, длительности переналадок не зависят от последовательности обслуживания требований, критерий состоит в минимизации взвешенной суммы моментов завершения.

Задача $Qm/sum-job, s_J = cp, p_j = p, q_J = q, d_j = d/C_j \leq d_j$: имеется m параллельных приборов разной производительности, где m фиксировано, несколько групп требований, каждая из которых может быть разбита на несколько партий, каждое требование партии имеет индивидуальный момент завершения обслуживания, каждая группа включает одинаковое количество идентичных требований с одинаковыми значениями $p_j = p$ и одинаковыми директивными сроками, длительности переналадок одинаковы и кратны cp, критерий состоит в отыскании расписания, допустимого относительно заданных директивных сроков.

Задача $1/sum-batch, g=1, s_J = s/\sum w_j U_j$: имеется один прибор и одна группа требований, которая может быть разбита на партии, все требования партии покидают прибор одновременно в момент времени, когда прибор завершает работу по обслуживанию "последнего" требования этой партии, длительности всех переналадок одинаковы, критерий состоит в минимизации взвешенного числа запаздывающих требований.

В приведенных выше примерах требования обслуживаются последовательно и размеры партий неограничены.

Задача $1/max-batch, b<n, g = 1, s_J = 0/L_{\max}$: имеется один прибор и одна группа требований, требования одной и той же партии должны обслуживаться параллельно так, что длительность обслуживания партии равна максимальной из длительностей обслуживания требований этой партии. Каждая партия может включать не более чем b, $b < n$, требований. Длительности всех переналадок равны нулю. Требуется минимизировать максимальное временное смещение.

Задание. Расшифровать обозначение $1/sum-batch, s_J, p_J, d_J, chain/L_{\max}$.

Далее приводятся наиболее характерные задачи обслуживания требований партиями и подходы к их решению.

4.3. Фиксированные партии

Предполагается, что разбиение требований на партии задано априори и не может быть изменено. Таким образом, требования каждой группы всегда образуют одну партию. Поскольку в этом случае понятия “партия” и “группа” совпадают, термин “группа” используется для обозначения обоих этих понятий.

Задача $1/GT, s_J, prec/f_{\max}$ может быть сформулирована следующим образом. Требования множества $\{1, \ldots, n\}$ обслуживаются одним прибором, начиная с момента $t = 0$. Для каждого требования j заданы длительность его обслуживания $p_j > 0$ и неубывающая функция $f_j(t)$ стоимости завершения обслуживания j в момент времени t, $f_j(t) \geq 0$ при $t \geq 0$. Задано разбиение множества требований на g групп. На множестве требований группы J заданы поизвольные отношения предшествования $\overset{J}{\to}$, $J = 1, 2, \ldots, g$, а на множестве групп – произвольные отношения предшествования $\overset{0}{\to}$. Перед началом обслуживания каждой группы J должна быть выполнена переналадка (наладка) прибора длительности s_J. Задача состоит в отыскании такой допустимой последовательности обслуживания требований, которая минимизирует функцию

$$f_{\max}(C_1, \ldots, C_n) = \max\{f_j(C_j) | j = 1, \ldots, n\}.$$

Напомним, что для множества Q и заданного на нем отношения предшествования элемент $x^0 \in Q$ называется минимальным элементом этого множества, если не существует элемента $x \in Q$, которому x^0 предшествует. Множество всех минимальных элементов множества Q обозначим через Q^-.

Рассмотрим задачу $1/\overset{J}{\to}/f_{\max}$, в которой имеется только одна группа $J = \{1, \ldots, n\}$, на множестве требований этой группы задано отношение предшествования $\overset{J}{\to}$ и соответствующая переналадка равна нулю. В этом случае алгоритм сложности $O(n^2)$, предложенный Е. Лоулером и описанный в разделе 2.1 (алгоритм А1), строит оптимальную последовательность требований $(i_1, \ldots, i_n)$. Этот алгоритм обобщается для задачи $1/GT, s_J, prec/f_{\max}$ следующим образом.

Алгоритм А2 *(для g групп)*

Шаг 1. (Инициализация). Полагаем $U = g$, $H_U = \{1, \ldots, U\}$, $R_U = \sum_{j=1}^{n} p_j + \sum_{I=1}^{U} s_I$.

Шаг 2. (Отыскание элемента I_U оптимальной последовательности групп). Строим множество минимальных элементов H_U^- относительно $\overset{0}{\to}$. Для каждой группы $J \in H_U^-$ применяем алгоритм А1, в котором $O_n = \{J\}$ и $P_n = R_U$. Если $\pi_J = (i_1, \ldots, i_k)$ – последовательность, найденная этим алгоритмом, то вычисляем значение

$$\Phi_J(R_U) = \max\{f_{i_l}(C_{i_l}) | l = 1, \ldots, k, C_{i_k} = R_U\}, \tag{4.1}$$

где C_{i_l} – момент завершения обслуживания требования i_l в последовательности π_J при условии, что $C_{i_k} = R_U$. Отыскиваем группу I_U такую, что $\Phi_{I_U}(R_U) = \min\{\Phi_J(R_U) | J \in H_U^-\}$. Если $U = 1$, то алгоритм прекращает работу: оптимальная последовательность групп $(\pi_{I_1}, \ldots, \pi_{I_g})$ построена. В противном случае полагаем $H_{U-1} = H_U \backslash \{I_U\}, R_{U-1} = R_U - (s_{I_U} + \sum_{j \in I_U} p_j)$ и повторяем шаг 2 при $U = U - 1$.

Алгоритм A2 решает задачу $1/GT, s_J, prec/f_{\max}$ и имеет временную сложность $O(gn^2)$.

4.4. Индивидуальное завершение обслуживания

Для данной модели допускается разбиение любой группы требований на несколько партий. Каждое требование покидает прибор в момент, когда прибор прекращает работу по его обслуживанию.

4.4.1. Один прибор. Идентичные требования в группах. Для ряда задач доказано, что существует оптимальное GT-расписание, при котором требования каждой группы образуют одну партию.

Теорема 4.1 *Существует оптимальное решение задачи* $1/sum-job, s_J, p_J, d_J/L_{\max}$, *при котором все требования каждой группы образуют одну партию и группы упорядочены по неубыванию значений* d_J.

Теорема 4.2 *Существует оптимальное решение задачи* $1/sum-job, s_J, p_J, w_J/\sum w_jC_j$, *при котором все требования каждой группы образуют одну партию и группы упорядочены по неубыванию значений* $(s_J + p_Jq_J)/(w_Jq_J)$.

Далее исследуем задачу $R/sum-job, s_J^{(l)}, p_J, d_J/C_j \le d_j$, в которой критерием является соблюдение директивных сроков.

Рассмотрим некоторое допустимое относительно директивных сроков расписание. Если две или более партии одной и той же группы обслуживаются на одном приборе, то без нарушения допустимости расписания эти партии могут быть объединены в одну партию, и обслуживаться этим же прибором. Следовательно, можно ограничиться рассмотрением расписаний, при которых каждый прибор обслуживает не более одной партии каждой группы. Эти партии могут обслуживаться каждым прибором в порядке EDD неубывания директивных сроков их групп. Таким образом, задача сводится к отысканию разбиения каждой группы на не более чем m партий и распределению этих партий по различным приборам.

Перейдем к изучению частных случаев задачи.

4.4.2. NP-трудные задачи. Ниже устанавливается вычислительная сложность ряда частных случаев.

Теорема 4.1. ***Задачи*** $P2/sum-job, s_J = 1, p_j = 1, d_j = d/C_j \le d_j$ ***и*** $P2/sum-job, s_J = 1, q_J = 1, d_j = d/C_j \le d_j$ ***являются NP-трудными, а задачи*** $P/sum-job, s_J = 1, p_j = 1, d_j = d/C_j \le d_j$ ***и*** $P/sum-job, s_J = 1, q_J = 1, d_j = d/C_j \le d_j$ – ***NP-трудными в сильном смысле.***

Доказательство. При доказательстве NP-трудности задачи $P2/sum-job, s_J = 1, p_j = 1, d_j = d/C_j \le d_j$ в качестве эталонной воспользуемся NP-полной задачей РАЗБИЕНИЕ. Имея пример задачи РАЗБИЕНИЕ, пример рассматриваемой задачи построим следующим образом. Обслуживающая система состоит из двух идентичных приборов, которые должны обслужить r групп

с параметрами $p_j = s_j = 1$, $d_j = d = A$ и $q_j = a_j - 1$ для группы j, $j = 1, \ldots, r$. Покажем, что множество $X \subset N$, для которого $\sum_{j \in X} a_j = A$, существует тогда и только тогда, когда для построенного примера имеется расписание, при котором $C_j \leq d$, $j = 1, \ldots, r$, где C_j – момент завершения обслуживания последнего требования группы j.

Если существует указанное множество X, для которого $\sum_{j \in X} a_j = A$, то объединим все требования каждой группы в одну партию, и будем обслуживать группы множества X первым прибором, а все остальные группы – вторым прибором. При этом время завершения обслуживания всех требований равно в точности $A = d$.

Если существует расписание, при котором $C_j \leq d$, $j = 1, \ldots, r$, то при этом расписании имеется в точности r партий, поскольку в противном случае сумма длительностей обслуживания и переналадок была бы не меньше чем $2A+1$, и время завершения обслуживания всех требований было бы не меньше чем $A + 1/2 > d$. Поэтому каждая группа обслуживается единой партией, и множество X соответствует множеству групп, обслуживаемому одним из приборов.

Для доказательства NP-трудности в сильном смысле задачи $P/sum{-}job, s_J = 1, p_j = 1, d_j = d/C_j \leq d_j$ используется сведение NP-трудной в сильном смысле задачи 3-РАЗБИЕНИЕ. Для примера этой задачи строится пример задачи обслуживания требований партиями, в котором обслуживающая система состоит из r параллельных идентичных приборов. Количество групп равно $3r$. Группа j имеет параметры $p_j = s_j = 1$, $d_j = d = A$, $q_j = a_j - 1$. Доказательство аналогично приведенному выше для случая двух приборов.

Доказательства NP-трудности задачи $P2/sum-job, s_J = 1, q_J = 1, d_j = d/C_j \leq d_j$ и NP-трудности в сильном смысле задачи $P/sum{-}job, s_J = 1, q_J = 1, d_j = d/C_j \leq d_j$ не представляют затруднений, поскольку в этих случаях каждая группа содержит единственное требование. ■

Задание. Показать, что задачи $P2/sum-job, s_J = 1, q_J = 1, d_j = d/C_j \leq d_j$, $P2/sum-job, s_J = 1, p_j = 1, d_j = d/C_j \leq d_j$ и $P/sum-job, s_J = 1, q_J = 1, d_j = d/C_j \leq d_j$ принадлежат классу $\mathcal{NP}$. Объяснить, в чем состоит сложность определения принадлежности классу $\mathcal{NP}$ задачи $P/sum{-}job, s_J = 1, p_j = 1, d_j = d/C_j \leq d_j$.

4.4.3. Полиномиально разрешимые задачи. Для решения задачи $Qm/sum{-}job, s_J = cp, p_j = p, q_J = q, d_j = d/C_j \leq d_j$, модифицируем правило Макнотона или обертки (см. раздел 2.3).

Назовем партию *неполной*, если она включает меньше чем q требований. Для каждой фиксированной последовательности приборов $(\pi_1, \ldots, \pi_m)$ при помощи правила Макнотона сформируем расписание следующей структуры. Максимальное количество целых групп обслуживается в интервале времени $[0, d]$ прибором π_1. За ними, возможно, следует неполная партия, включающая наибольшее число требований, скажем y_1, которое может быть обслужено в оставшемся временном интервале прибором π_1. Неполная партия, включающая $z_2 = q - y_1$ требований, обслуживается первой прибором π_2. Допустим, что ее обслуживание завершается в момент времени d_2. Тогда максимальное количество целых групп обслуживается в интервале времени $[d_2, d]$ прибором π_2. За ними следует неполная партия, включающая наибольшее число требований, скажем y_2, которое может быть обслужено в оставшемся временном интервале прибором

π_2. Процесс повторяется для приборов $\pi_3, \ldots, \pi_m$ до тех пор, пока не будет построено допустимое расписание либо не возникнет ситуация, когда не существует допустимого расписания для последовательности приборов $(\pi_1, \ldots, \pi_m)$. В последнем случае рассматривается новая последовательность приборов.

Приведенный алгоритм решает задачу $Qm/sum-job, s_J = cp, p_j = p, q_J = q, d_j = d/C_j \leq d_j$ и имеет временную сложность $O(mm!)$.

Отметим, что если приборы идентичны, то все их последовательности неразличимы, и задача $P/sum-job, s_J = cp, p_j = p, q_J = q, d_j = d/C_j \leq d_j$ может быть решена за время $O(m)$ при помощи правила Макнотона. Однако функция $O(m)$ не может рассматриваться как полином от длины записи входных данных этой задачи, поскольку длина записи равна $O(\log m + \log g + \log p + \log c + \log q + \log d)$.

Ниже для решения этой задачи предлагается алгоритм сложности $O(\log m)$. Вначале покажем, что задача $P/sum-job, s_J = cp, p_j = p, q_J = q, d_j = d/C_j \leq d_j$ эквивалентна задаче $P/sum-job, s_J = s, p_j = 1, q_J = q, d_j = d/C_j \leq d_j$ с единичными длительностями обслуживания. Легко проверить, что расписание является допустимым для задачи с параметрами $p_j = p$, $s_J = cp$ и $d_j = d$ тогда и только тогда, когда оно допустимо для задачи с параметрами $p_j = p$, $s_J = cp$ и $d_j = rp$, где $r = \lfloor d/p \rfloor$. Последняя задача, очевидно, эквивалентна задаче с параметрами $p_j = 1$, $s_J = c$ и $d_j = r$.

Предлагаемый алгоритм решения задачи $P/sum-job, s_J = s, p_j = 1, q_J = q, d_j = d/C_j \leq d_j$ формирует следующее расписание. В интервале времени $[0, D]$, где $D = (s+q)\lfloor d/(s+q) \rfloor$, каждый прибор обслуживает наибольшее количество целых групп. Если при этом обслужены все группы, то допустимое расписание построено. Пусть g' означает число необслуженных групп и $g' > 0$. Каждая необслуженная группа разбивается на партии, содержащие $d-D-s$ требований и, возможно, одну партию, содержащую V, $V < d - D - s$, требований. Партии, содержащие $d-D-s$ требований каждая, обслуживаются m приборами. Если все такие партии обслужены, и $V = 0$, то допустимое расписание построено. Если количество r таких партий больше, чем m, или $r = m$ и $V > 0$, то допустимого расписания не существует. В противном случае полагаем $m' = m - r, d' = d - D, s' = s, q' = V$ и применяем приведенную выше процедуру рекурсивно для задачи, определяемой параметрами g', m', d', s', q'. Можно считать, что $r > m/2$, поскольку в противном случае все партии, включающие V требований каждая, могут быть обслужены в интервале $[D, d]$ оставшимися m' приборами, и допустимое расписание будет построено. Таким образом, $m' < m/2$, и количество рекурсивных применений процедуры не превосходит $O(\log m)$. На каждом шаге рекурсии соответствующее частичное расписание может быть построено за время, равное константе.

4.5. Одновременное завершение обслуживания

Рассматриваются задачи, в которых требования одной и той же партии покидают прибор одновременно в момент, когда прибор завершает работу по обслуживанию всех требований этой партии. В каждой партии требования обслуживаются прибором либо последовательно, и то-

гда длительность обслуживания партии равна сумме длительностей обслуживания входящих в нее требований, либо параллельно, и тогда длительность обслуживания партии равна максимальной из длительностей обслуживания входящих в нее требований. Предполагается, что все требования принадлежат одной группе, т.е. $g = 1$.

4.5.1. Последовательное обслуживание. Вначале предположим, что требования каждой партии обслуживаются прибором непосредственно друг за другом.

Ниже приводятся алгоритмы решения задач $1/sum-batch, g = 1, s_J = s/f$, в которых $f \in \{C_j \le d_j, L_{\max}, f_{\max}\}$. При разработке этих алгоритмов использовано следующее утверждение.

Лемма 4.1. ***Существует оптимальное для задачи*** $1/sum-batch, g = 1, s_J = s/L_{\max}$ ***расписание, при котором***

а) требования обслуживаются в порядке неубывания EDD директивных сроков;

б) требования с одинаковыми директивными сроками расположены в одной партии.

Задание. Используя перестановочный прием, доказать лемму 4.1.

Нетрудно видеть, что для задач минимизации $T_{\max}$ и отыскания расписания, допустимого относительно директивных сроков, также существуют оптимальные расписания, удовлетворяющие условиям а) и б) леммы 4.1.

На основании приведенной леммы размерность указанных задач можно сократить, объединяя требования с одинаковыми директивными сроками в одно *составное требование* с тем же директивным сроком и длительностью обслуживания, равной сумме длительностей обслуживания образующих его исходных требований.

Не ограничивая общности, предположим, что (составные) требования пронумерованы так, что $d_1 < \ldots < d_n$.

Опишем алгоритм динамического программирования для задачи минимизации $L_{\max}$. В этом алгоритме требования добавляются в начало текущих расписаний целыми партиями, содержащими последовательно пронумерованные требования.

Определим расписание обслуживания требований $j, \ldots, n$ как соответствующее состоянию j. Если расписания, соответствующие этому состоянию, достроить до полных расписаний одинаковыми партиями, образованными требованиями $1, \ldots, j-1$, то для всех полученных расписаний значения $L_1, \ldots, L_{j-1}$ и добавки к значениям $L_j, \ldots, L_n$ будут одинаковы.

Пусть F_j – наименьшее значение $L_{\max}$ среди частичных расписаний, соответствующих состоянию j. Очевидно, что расписание со значением F_j доминирует все остальные расписания, соответствующие состоянию j, т.е. это расписание может быть достроено до полного расписания с наименьшим значением целевой функции среди всех полных расписаний, достроенных из частичных расписаний, соответствующих тому же состоянию.

Наименьшее значение $L_{\max}$ может быть найдено путем рекурсивного вычисления значений F_j. Для инициализации рекурсии полагаем $F_{n+1} = -\infty$.

Рекурсия для $j = n, \ldots, 1$ состоит в следующем:

$$F_j = \min_{j<i\le n+1}\{F_j(i)\},\ \text{где}\ F_j(i) = s + \sum_{l=j}^{i-1} p_l + \max\{-d_j, F_i\}.$$

Величина $F_j(i)$ преставляет собой наименьшее значение $L_{\max}$ среди расписаний, соответствующих состоянию j, при которых первая партия включает требования $j, j+1, \ldots, i-1$. Наименьшее значение $L_{\max}$ равно F_1.

Алгоритм требует время и память в объеме $O(n^2)$, если значения $\sum_{l=j}^{i} p_l$, $j, i = 1, \ldots, n$, вычислены на предварительном шаге. Алгоритм также решает задачу минимизации $T_{\max}$ и задачу отыскания расписания, допустимого относительно директивных сроков.

Опишем алгоритм временной сложности $O(n \log n)$ для задачи отыскания расписания, допустимого относительно директивных сроков, $1/sum-batch, g = 1, s_J = s/C_j \le d_j$. Пусть (составные) требования по-прежнему пронумерованы по возрастанию директивных сроков.

Рассмотрим последовательность требований $(1, 2, \ldots, n)$. Включим в первую партию максимальное количество требований с наименьшими номерами, которое может быть обслужено до директивного срока d_1. Пусть j – номер последнего требования первой партии. Включим во вторую партию максимальное количество требований с наименьшими номерами (за исключением требований первой партии), которое может быть обслужено до директивного срока d_{j+1}. Продолжаем приведенный процесс формирования партий до включения в некоторую партию требования n либо возникновения ситуации, когда очередную партию невозможно сформировать без нарушения соответствующего директивного срока. В первом случае допустимое расписание построено. Второй случай означает, что такое расписание не существует.

Описанная процедура имеет временную сложность $O(n)$ при условии, что требования уже пронумерованы нужным образом. В противном случае ее временная сложность – $O(n \log n)$.

В заключение данного раздела покажем, как построить полиномиальный алгоритм для задачи минимизации $f_{\max}$, используя в качестве вспомогательного алгоритм временной сложности $O(n)$ для задачи отыскания допустимого относительно директивных сроков расписания.

Вначале отметим, что решение задачи минимизации $f_{\max}$ может быть сведено к решению последовательности задач распознавания вида “Справедливо ли $f_{\max} \le k$?”, где k является изменяющейся величиной, подбираемой с помощью двоичного поиска в подходящем интервале значений. Поэтому если указанная задача распознавания разрешима за полиномиальное время, то задача минимизации $f_{\max}$ также разрешима за полиномиальное время, если оптимальное значение целевой функции этой задачи является целым числом, чей логарифм ограничен полиномом от размерности входных данных задачи. Предположим, что последнее условие выполняется.

На вопрос: “Справедливо ли $f_{\max} \le k$?”, можно ответить за полиномиальное время следующим образом. Величина k порождает директивный срок $\bar{d}_j$ для каждого требования j, $j = 1, \ldots, n$. Директивный срок $\bar{d}_j$ может быть найден за время $O(\log P)$ при помощи двоичного поиска в интервале $[s+p_j, P]$ возможных моментов завершения обслуживания требования j, где $P = ns + \sum_{j=1}^{n} p_j$. Величина $\bar{d}_j$ находится из условия $\bar{d}_j = \max\{d | f_j(d) \le k, s + p_j \le d \le P\}$. Как только указанные директивные сроки определены, можно использовать алгоритм временной

сложности $O(n)$ для выяснения вопроса о существовании расписания, допустимого относительно этих директивных сроков. Поэтому ответ на вопрос: "Справедливо ли $f_{\max} \leq k$?", может быть получен за полиномиальное время $O(n+n\log P)$. Здесь не учитывается сложность вычисления значений функций $f_j(x)$. Таким образом, задача минимизации $f_{\max}$ может быть решена за полиномиальное время.

4.5.2. Параллельное обслуживание. Перейдем к рассмотрению задач, в которых требования партии обслуживаются прибором параллельно и длительность обслуживания партии равна максимуму длительностей обслуживания входящих в нее требований.

В задаче $1/max-batch, g = 1, s_J = 0/f$, где f – некоторая регулярная функция, предполагается, что размер партий неограничен ($b = n$). Для этой задачи расписание однозначно определяется последовательностью партий $(B_1, \ldots, B_k)$. Обозначим длительность обслуживания партии B_l через $p(B_l) = \max_{j\in B_l}\{p_j\}$, а момент завершения ее обслуживания при заданной последовательности партий – через $C(B_l) = \sum_{q=1}^{l} p(B_q)$. Напомним, что при заданной последовательности партий момент завершения обслуживания требования j равен $C_j = C(B_l)$ для каждого $j \in B_l$, $l = 1, \ldots, k$.

Вначале отметим, что при неограниченных размерах партий задача минимизации $C_{\max}$ решается тривиально путем назначения всех требований в единственную партию B_1. Тогда минимальное значение $C_{\max}$ равно $p(B_1) = \max_{1\leq j\leq n}\{p_j\}$.

Далее в этом разделе предполагается, что требования пронумерованы так, что $p_1 \leq \ldots \leq p_n$ (по правилу SPT).

Лемма 4.2. ***Для задачи*** $1/max-batch, g = 1, s_J = 0/f$ ***существует оптимальная последовательность партий*** $(B_1, \ldots, B_k)$ ***такая, что*** $B_1 = \{1, 2, \ldots, i_1\}$, $B_2 = \{i_1+1, i_1+2, \ldots, i_2\}$, $\ldots$, $B_k = \{i_{k-1}+1, i_{k-1}+2, \ldots, n\}$.

Задание. Используя перестановочный прием, доказать лемму 4.2.

Из этой леммы следует существование расписания, которое определяется последовательностью SPT требований и подходящим разбиением этой последовательности на партии. Расписание такого вида будем называть *SPT-расписанием.*

Перейдем к описанию алгоритмов динамического программирования. Рассмотрим произвольное SPT-расписание для требований $1, \ldots, j$ такое, что обслуживание последней партии завершается в момент времени t. Будем говорить, что такое расписание соответствует состоянию (j, t). Очевидно, что расписание с минимальным значением целевой функции $\sum f_j$ либо $f_{\max}$ доминирует все остальные расписания, соответствующие одному и тому же состоянию.

Пусть σ – некоторая SPT-последовательность с наименьшим значением целевой функции среди последовательностей, соответствующих состоянию (j, t). Для того чтобы получить данную последовательность, необходимо добавить партию $\{i+1, \ldots, j\}$, где $0 \leq i < j$, в конец последовательности партий для требований $1, \ldots, i$, соответствующей некоторому предыдущему состоянию. Поскольку партия $\{i+1, \ldots, j\}$ имеет длительность обслуживания p_j, предыдущим состоянием является $(i, t-p_j)$. Добавление этой партии увеличивает суммарную стоимость на

$\sum_{k=i+1}^{j} f_k(t)$. Максимальная стоимость для требований $i+1, \ldots, j$ равна $\max_{i+1 \le k \le j}\{f_k(t)\}$.

На этих рассуждениях основан следующий прямой алгоритм динамического программирования для задачи минимизации $\sum f_j$.

Пусть $F_j(t)$ – минимальное значение целевой функции, вычисленное для всех SPT-расписаний обслуживания требований $1, \ldots, j$ при условии, что обслуживание последней партии завершается в момент времени t. Начальные значения равны

$$F_0(t) = \begin{cases} 0, & \text{если } t = 0, \\ \infty & \text{в противном случае.} \end{cases}$$

Рекурсия для $j = 1, \ldots, n$ и $t = p_j, \ldots, \sum_{k=1}^{j} p_k$ состоит в следующем:

$$F_j(t) = \min_{0 \le i \le j-1} \Big\{ F_i(t - p_j) + \sum_{k=i+1}^{j} f_k(t) \Big\}.$$

Минимальное значение целевой функции равно $\min_{p_n \le t \le P} \{F_n(t)\}$, где P – сумма всех длительностей обслуживания требований. Соответствующее оптимальное SPT-расписание отыскивается при помощи обратного хода алгоритма.

Поскольку переменные состояния принимают значения $j = 0, \ldots, n$ и $t = 0, \ldots, P$ и имеется j возможных партий, которые могут быть добавлены для того, чтобы получить состояние (j, t), временная сложность алгоритма не превосходит $O(n^3 P)$, и требуемый объем памяти не превосходит $O(nP)$. Временная сложность снижается в n раз, если частичные суммы $\sum_{k=1}^{j} f_k(t)$ для $j = 1, \ldots, n$ и $t = p_j, \ldots, \sum_{k=1}^{j} p_k$ вычисляются заранее (это можно сделать за время $O(nP)$).

В случае, когда размер каждой партии ограничен сверху числом b, $b < n$, соответствующие задачи являются по крайней мере такими же трудными, как и их традиционные аналоги, поскольку при $b = 1$ прибор может обслуживать не более одного требования в каждый момент времени. Эти задачи намного сложнее, чем соответствующие задачи с неограниченными размерами партий, поскольку поиск оптимального расписания не может быть ограничен рассмотрением SPT-расписаний.

Единственным исключением является задача минимизации $C_{\max}$, для которой существует оптимальное SPT-расписание, в котором все партии являются полными (включают b требований), кроме одной, которая включает требования с наименьшими длительностями обслуживания.

Литература

[1] Беллман Р. Динамическое программирование. – М.: ИЛ, 1960.

[2] Гэри М., Джонсон Д. Вычислительные машины и труднорешаемые задачи. – М.: Мир, 1982.

[3] Р.В. Конвей, В.Л. Максвелл и Л.В. Миллер. Теория расписаний. – М.: Наука, 1975.

[4] Танаев В.С., Гордон В.С., Шафранский Я.М. Теория расписаний. Одностадийные системы. – М.: Наука, 1984.

[5] Танаев В.С., Ковалев М.Я., Шафранский Я.М. Теория расписаний. Групповые технологии. – Минск: ИТК НАН Беларуси, 1998.

[6] Танаев В.С., Шкурба В.В. Введение в теорию расписаний. – М.: Наука, 1975.