РОССИЙСКАЯ АКАДЕМИЯ НАУК
СИБИРСКОЕ ОТДЕЛЕНИЕ
ИНСТИТУТ МАТЕМАТИКИ им. С.Л. СОБОЛЕВА

На правах рукописи
УДК 519.854

Кононов Александр Вениаминович

АКТУАЛЬНЫЕ ЗАДАЧИ ТЕОРИИ РАСПИСАНИЙ: ВЫЧИСЛИТЕЛЬНАЯ СЛОЖНОСТЬ И ПРИБЛИЖЕННЫЕ АЛГОРИТМЫ

01.01.09 — дискретная математика и математическая кибернетика

Диссертация на соискание ученой степени
доктора физико-математических наук

Новосибирск — 2014

Оглавление

Введение

Актуальность темы. Объектом исследования настоящей диссертации являются задачи теории расписаний. Это широкий круг, как правило, NP-трудных задач комбинаторной оптимизации, имеющих обширные приложения в организации производства, в бизнесе, в оптимизации компьютерных вычислений, в индустрии сервиса и во многих других областях человеческой деятельности. Предметом исследования являются энергетические задачи теории расписаний, задачи с воспроизводимым ресурсом, задачи оптимизации параллельных вычислений с задержками передачи данных и цеховые задачи с маршрутизацией, для которых исследуются вопросы алгоритмической сложности и построения точных или приближенных алгоритмов.

Задачи теории расписаний и задачи календарного планирования связаны с распределением ограниченных ресурсов для выполнения множества работ и поиском расписания с наилучшим значением заданной целевой функции. Оба направления возникли в 50-х годах прошлого столетия [113, 115, 122, 123, 161] и обусловлены как ростом сложности комбинаторных задач по управлению производственными процессами, так и появлением первых быстродействующих вычислительных устройств, давших надежду решить эти задачи за приемлемое время. Хотя формально все задачи теории расписаний можно отнести к задачам календарного планирования, длительное время теория расписаний развивалась как отдельное направление дискретной оптимизации. В классических задачах теории расписаний фактически не рассматриваются ресурсы, отличные от ресурса "машина", а основной упор делается на различные технологии выполнения множества работ и на разнообразие целевых функций. К середине 80-х годов прошлого столетия были выделены основные модели классической теории расписаний и установлена комбинаторная сложность большинства задач, возникающих в рамках этих моделей. Не удивительно, что примерно в то же время были предприняты первые попытки расширить классические модели теории расписаний внедрением в них таких дополнительных ресурсов, как энергия, деньги, машинная память и многие другие [57, 59, 60]. Новое направление исследований получило название "задачи теории расписаний с ограничениями на ресурсы"[58] и стало одним из наиболее популярных направлений в теории расписаний в последние два десятилетия.

Хотя первые задачи теории расписаний моделировали производственные процессы, вскоре выяснилось, что схожие задачи возникают и в других областях человеческой деятельности, в особенности, в организации оптимальной работы современных вычислительных устройств. Быстрое развитие компьютерных технологий порождает

огромное количество новых задач в теории расписаний.

В частности, одно из популярных направлений развития компьютерных вычислений — это параллельные вычисления. При этом основная сложность при проектировании параллельных программ состоит в том, чтобы обеспечить правильную последовательность взаимодействий между различными вычислительными процессами, а также координацию ресурсов, распределяемых между процессами. В 1987 Рэйвард-Смит [147] предложил рассмотреть однородную коммуникационную модель выполнения множества заданий параллельной программы как задачу теории расписаний с коммуникационными задержками. Идея, предложенная Рэйвард-Смитом, оказалась очень плодотворной, и задачи с коммуникационными задержками интенсивно изучаются уже больше двух десятилетий. Результатам в данном направлении посвящены главы в монографиях [84] и [75] и несколько обзоров [82, 159], каждый из которых содержит большой список открытых проблем.

Другим важным достижением в развитии компьютерных технологий стала возможность современных многопроцессорных компьютеров, таких как Intel SpeedStep или AMD PowerNow, варьировать свою скорость. Высокие скорости вычислений приводят к более быстрому и качественному обслуживанию, но при этом и к большим затратам энергии. Поиск золотой середины между экономией энергии и сохранением качества обслуживания породил новый класс задач, в которых требуется найти энергетически эффективные расписания. Результаты, полученные в этой области, настолько интересны, а открытые проблемы настолько важны, что обзор, посвященный различным задачам минимизации энергии при построении расписаний, был опубликован в одном из самых престижных журналов по проблемам передачи информации "Communications of the ACM"[28].

Еще одним интересным современным направлением исследования является слияние задач теории расписаний с другими классическими задачами дискретной оптимизации в одну общую задачу. Это в некоторой степени относится к задачам теории расписаний с ограничениями на ресурсы, в которых к исходной постановке задачи в терминах теории расписаний добавляются ограничения, характерные для задач об упаковках и задач календарного планирования. Другим интересным примером являются цеховые задачи теории расписаний с маршрутизацией, которые обобщают цеховые задачи с широко известной метрической задачей коммивояжера. Удивительно, что постановки таких проблем независимо появились при рассмотрении задач, возникающих как на производстве [38, 39], так и в индустрии обслуживания [73, 167].

Большинство задач теории расписаний являются NP-трудными. Напомним, что поиск для них точных алгоритмов решения, время работы которых ограничено полиномом от размера входа задачи, в настоящее время представляется бесперспективным. Экспоненциальные алгоритмы переборного типа требуют значительных вычислительных затрат даже при решении примеров средней размерности. Поэтому одним из важных направлений исследований является построение и анализ приближенных алгоритмов с гарантированной оценкой точности для NP-трудных задач. В настоящее время это направление завоевало огромное число сторонников в среде исследователей, занимающихся компьютерной математикой. Перечислим основные причины

такой популярности.

- Существует огромное количество оптимизационных задач, которые требуют решения, и большинство из них NP-трудные. Для многих из них необязательно находить точные ответы, а достаточно построить хорошее разумное решение за короткое время, с чем успешно справляются приближенные алгоритмы.

- На практике большинство оптимизационных задач очень сложны, поскольку включают в себя много дополнительных ограничений, которые не позволяют найти хорошее гарантированное приближенное решение. Но зачастую приближенные алгоритмы для более простых вариантов тех же задач подсказывают новые идеи для эвристик, которые затем успешно применяются и для практических задач.

- Построение приближенных алгоритмов с гарантированной оценкой точности требует анализа поведения алгоритмов и выявления свойств оптимальных решений, то есть обнаружения структурных свойств рассматриваемых задач. В свою очередь лучшее понимание структуры задачи ведет к образованию новых алгоритмических идей.

- С точки зрения нахождения точных решений почти все NP-трудные задачи эквивалентны друг другу. Построение приближенных алгоритмов или доказательство невозможности их построения при условии несовпадения классов P и NP дает возможность сравнить между собой NP-трудные задачи по тому, насколько хорошо они могут быть аппроксимированы.

- Построение приближенных алгоритмов с гарантированной оценкой точности часто основано на новых глубоких и интересных результатах в различных областях математики и способствует развитию математики в целом.

Таким образом, установление вычислительной сложности задач теории расписаний и разработка для NP-трудных задач теории расписаний эффективных приближенных алгоритмов с гарантированной оценкой точности является актуальным направлением в современной математике.

Цель данной работы состоит в изучении комбинаторных свойств задач теории расписаний и установлении их вычислительной сложности. Работа посвящена построению точных полиномиальных алгоритмов для рассматриваемых задач или доказательству их NP-трудности и разработке для NP-трудных задач приближенных алгоритмов с гарантированной оценкой точности.

Методика исследований. В работе использовались методы дискретной оптимизации, такие как полиномиальная и псевдополиномиальная сводимость, методы динамического программирования, специальные методы построения приближенных алгоритмов, в том числе жадных алгоритмов, алгоритмов на основе линейного программирования, округления дробных решений, рандомизированного округления, а

также методы анализа точности алгоритмов, разработанные автором.

Научная новизна. Все представленные в диссертации результаты являются новыми.

- В работе сформулировано понятие воспроизводимого ресурса и установлена его связь с общепринятыми понятиями возобновимого и невозобновимого ресурсов. Для задач с воспроизводимым ресурсом предложены первые приближенные алгоритмы с гарантированной оценкой точности.

- Разработан общий метод построения приближенных алгоритмов для задач составления энергетически эффективных расписаний. Впервые рассмотрены неоднородные задачи построения энергетически эффективных расписаний. Для них построены алгоритмы решения, близкие по качеству к известным алгоритмам для аналогичных однородных задач.

- Построены первые нетривиальные приближенные алгоритмы для задачи с задержками передачи данных в иерархической системе.

- Предложены новые приближенные алгоритмы для различных задач теории расписаний и проведен анализ погрешности этих алгоритмов. На момент написания диссертации все представленные в ней алгоритмы имели наилучшую оценку точности среди алгоритмов, трудоемкость которых ограничена полиномом от размера входных данных задачи.

Практическая и теоретическая ценность. Работа носит теоретический характер. Полученные в ней результаты позволяют лучше понять природу рассматриваемых задач, их сложность и границы возможностей полиномиальных алгоритмов. Для некоторых задач впервые установлена их комбинаторная сложность. Для всех рассмотренных NP-трудных задач найдены новые приближенные алгоритмы с лучшими известными оценками точности.

Отметим, что все задачи, рассматриваемые в диссертации, связаны либо с экономическими приложениями, либо с приложениями в области компьютерных технологий. При этом необходимо подчеркнуть, что все построенные алгоритмы, за исключением алгоритмов в параграфе 3.2 и 5.2, имеют низкую трудоемкость и могут быть использованы для эффективного построения хороших приближенных решений.

Апробация работы. Все разделы диссертации прошли апробацию на следующих конференциях в России и за рубежом:

– Международный рабочий семинар по моделям и алгоритмам для задач планирования и теории расписаний (MAPSP): 2001 (Ассуа, Франция), 2003 (Ассуа, Франция), 2005 (Сиена, Италия), 2009 (Ролдюк, Нидерланды), 2011 (Нимбурк, Чехия); 2013 (Понт-а-Муссон, Франция);

– Симпозиум по параллельным алгоритмам и архитектуре (SPAA): 2001 (Гераклион, Греция), 2003 (Сан Диего, США);

– Всероссийская конференция «Проблемы оптимизации и экономические приложения»: 2003, 2006, 2012 (Омск, Россия);

– Российская конференция «Дискретный анализ и исследование операций»: 2004, 2010 (Новосибирск, Россия);

– Международный рабочий семинар по календарному планированию и теории расписаний (PMS): 2004 (Нанси, Франция), 2010 (Тур, Франция), 2012 (Левен, Бельгия);

– Байкальская международная школа-семинар «Методы оптимизации и их приложения»: 2005, 2014 (Иркутск, Россия);

– Международный рабочий семинар по приближенным и он-лайн алгоритмам (WAOA), 2009 (Копенгаген, Дания);

– Балканская конференция по исследованию операций (BalCOR) 2011 (Салоники, Греция);

– Международная конференция по вычислительным методам и комбинаторике (COCOON) 2013 (Гуанчжоу, Китай);

– Международная конференция по основам технологии программного обеспечения и теоретической информатики (FSTTCS) 2013 (Гувахити, Индия).

Результаты диссертации докладывались на семинарах:

– «Математические модели принятия решений», Институт математики им. С.Л. Соболева СО РАН, Новосибирск;

– «Дискретные экстремальные задачи», Институт математики им. С.Л. Соболева СО РАН, Новосибирск;

– семинар лаборатории информатики (LIP 6) университета Пьера и Марии Кюри, Объединенный университет Сорбонны, Париж, Франция;

– семинар факультета информационного менеджмента Национального университета Чиао-Тунг, Хсинчу, Тайвань.

– семинар школы бизнеса и экономики университета Маастрихта, Нидерланды.

Личный вклад. Диссертационная работа представляет собой единый цикл многолетних исследований автора, объединенных не только предметом, но и методами исследований. В совместных работах соискателю принадлежат либо основные идеи анализа приближенных алгоритмов, включая доказательства основных утверждений и теорем, либо идеи новых приближенных алгоритмов. Некоторые доказательства утверждений и теорем выполнены в соавторстве при непосредственном участии соискателя. Конфликт интересов с соавторами отсутствует.

На защиту выносится совокупность результатов по разработке и анализу точных алгоритмов и приближенных алгоритмов с гарантированными оценками точности для задач теории расписаний с ограничениями на ресурсы или порядок выполнения работ и результаты по установлению комбинаторной сложности для задач теории расписаний с воспроизводимым ресурсом.

Публикации. По теме диссертации автором опубликовано 61 работа, в том числе 26 статей, все — в журналах из списка ВАК или включенных в базы данных

Scopus, Web of Science, ZBMath, 37 работ — в трудах российских и международных конференций.

Основные результаты диссертации

1. Решен открытый вопрос о вычислительной сложности задачи построения кратчайшего расписания единичных работ на двух параллельных идентичных машинах при наличии воспроизводимого ресурса, поставленный Амиром и Капланом в 1988 году [31]. Доказано, что данная задача является NP-трудной в сильном смысле. Также установлена NP-трудность в сильном смысле следующих задач с воспроизводимым ресурсом:

– задача построения кратчайшего расписания единичных работ на двух параллельных машинах с фиксированным распределением работ по машинам,

– задача минимизации суммы моментов завершения работ на одной машине,

– задача минимизации взвешенной суммы моментов завершения единичных работ на одной машине.

2. Построены первые приближенные алгоритмы с гарантированными оценками точности для задач построения энергетически эффективных расписаний, в которых запрещены прерывания работ.

3. Разработан общий подход к построению приближенных алгоритмов для задач построения энергетически эффективных расписаний, основанный на решении задач линейного программирования, в которых число переменных не ограничено полиномом от размера входа задачи, и последующим вероятностным округлением полученных решений. Как следствие, впервые получены приближенные алгоритмы с гарантированной оценкой точности для неоднородных задач на минимизацию расхода энергии.

4. Для задачи составления расписания работ на параллельных машинах в иерархической коммуникационной сети с малыми коммуникационными задержками предложен первый алгоритм, который находит приближенное решение с гарантированной оценкой точности меньше меньше тривиальной оценки, равной двум, а, именно, $12(\Phi + 1)/(12\Phi + 1) < 2$, где $\Phi \geq 1$ — отношение длительности минимальной работы к длине максимальной задержки.

5. Построены приближенные алгоритмы с гарантированными оценками точности для различных NP-трудных вариантов цеховой задачи открытого типа с маршрутизацией машин. Все построенные алгоритмы имеют лучшие оценки точности среди известных алгоритмов одинаковой с ними трудоемкости.

6. Построен точный псевдо-полиномиальный алгоритм для двухмашинной цеховой задачи открытого типа с маршрутизацией для NP-трудного случая, когда работы и машины расположены на двухвершинной сети. Как следствие, решен открытый вопрос о вычислительной сложности этой задачи.

Объем и структура диссертации. Диссертация состоит из введения, пяти глав, заключения и списка литературы (168 наименований). Объем диссертации — 196 страниц.

СОДЕРЖАНИЕ РАБОТЫ

Во введении обосновывается актуальность темы диссертации, формулируется цель исследования, дается общее содержание работы, а также приводятся основные результаты диссертации.

В первой главе вводятся основные термины и понятия, принятые в календарном планировании, теории расписаний и в целом в дискретной оптимизации. Обсуждаются подходы к изучению свойств рассматриваемых задач и методы их решения.

В первом разделе приводятся различные подходы к классификации ресурсов, независимо сложившиеся в 60-х и 70-х годах прошлого столетия в русскоязычной и англоязычной литературе по дискретной оптимизации. Обсуждаются достоинства и недостатки принятых классификаций и их последующее развитие.

Второй раздел посвящен описанию классификации задач теории расписаний, первоначально введенной в [99] и позднее развитой в многочисленных англоязычных монографиях [62, 64, 131]. Для обозначения каждой задачи используется $\alpha|\beta|\gamma$-идентификатор, в котором в поле α вносится информация о типе ресурсов задачи, в том числе о конфигурации машин, в поле β описываются характеристики работ и в поле γ — вид целевой функции. Данная классификация адаптируется к задачам, рассматриваемым в диссертации.

В третьем разделе обсуждается разделение задач на простые и сложные. В частности, вводятся базовые понятия, такие как задача распознавания, индивидуальная задача, ее размер входа, алгоритм, его время работы, определяются классы P и NP и дается краткий обзор результатов в теории вычислительной сложности. Большинство задач теории расписаний являются NP-трудными, то есть существование точных эффективных алгоритмов решения таких задач представляется маловероятным. Построение приближенных алгоритмов с априорной оценкой точности является одним из основных подходов к решению NP-трудных задач.

В четвертом разделе даны определения понятий приближенного алгоритма и приближенной схемы. Пятый раздел посвящен краткому описанию развития теории расписаний, ее современного состояния и возможных перспектив будущих исследований.

Вторая глава диссертации посвящена задачам теории расписаний с воспроизводимым ресурсом. Множество работ $\mathcal{J} = \{J_1, \ldots, J_n\}$ должно быть выполнено на

одной или нескольких машинах. Задан общий ресурс объема Ω_0. Перед началом выполнения работа J_i потребляет α_i единиц ресурса и воспроизводит β_i единиц ресурса сразу после своего завершения. Величина α_i называется *расходом* ресурса на работу J_i, величина β_i – *доходом* ресурса от работы J_i, и величина $\delta_i = \beta_i - \alpha_i$ – *прибылью* ресурса от работы J_i. Длительность работы J_i равна p_i. Прерывания в обслуживании работ запрещены. Требуется найти такое расписание σ, в котором выполнены все работы, и в каждый момент времени t остаток общего ресурса неотрицателен.

Понятие воспроизводимого ресурса обобщает понятия как возобновимого, так и невозобновимого ресурсов. Действительно, полагая в определении воспроизводимого ресурса $\alpha_i = \beta_i$ для всех работ, получаем возобновимый ресурс. А в случае, если для всех работ выполнено $\beta_i = 0$, имеем невозобновимый ресурс. Более того, классическая задача об упаковке в контейнеры может быть сформулирована, как задача с возобновимым ресурсом и единичными длительностями работ. Таким образом, задача с воспроизводимым ресурсом и единичными длительностями работ является естественным обобщением этой известной задачи.

В первом разделе изучается задача на одной машине, в которой требуется минимизировать взвешенную сумму моментов завершения работ. Рассмотрены следующие четыре варианта исходной задачи:

- задача с единичными длительностями всех работ и неположительной прибылью ресурса от каждой работы (задача $W1,1|p_i = 1, \delta_i \leq 0|\sum w_i C_i$),
- задача с единичными длительностями всех работ и неотрицательной прибылью ресурса от каждой работы (задача $W1,1|p_i = 1, \delta_i \geq 0|\sum w_i C_i$),
- задача с единичными весами всех работ и неположительной прибылью ресурса от каждой работы (задача $W1,1|\delta_i \leq 0|\sum C_i$),
- задача с единичными весами всех работ и неотрицательной прибылью ресурса от каждой работы (задача $W1,1|\delta_i \geq 0|\sum C_i$).

Установлено, что все четыре подзадачи являются NP-трудными в сильном смысле. Следующие две теоремы — основные результаты раздела 2.1.

Теорема 1 *Задача $W1,1|p_j = 1, \delta_j \leq 0|\sum w_i C_i$ является NP-трудной в сильном смысле.*

Теорема 2 *Задача $W1,1|p_j = 1, \delta_j \geq 0|\sum w_i C_i$ является NP-трудной в сильном смысле.*

В конце раздела представлены 2-приближенные жадные алгоритмы для задач $W1,1|p_j = 1, \delta_i \geq 0|\sum w_i C_i$ и $W1,1|\delta_i \leq 0|\sum C_i$.

Второй и третий разделы посвящены задачам минимизации длины расписания множества работ на параллельных идентичных машинах. В задаче $W1,P||C_{max}$ число параллельных машин не фиксировано и может быть различным для каждой индивидуальной задачи. В задаче $W1,Pm||C_{max}$ число параллельных машин фиксировано

и равно m. В этом разделе также изучается задача $W1||C_{max}$, в которой отсутствует ресурс типа "машина", т.е. произвольное число работ может выполняться параллельно.

Во втором разделе исследуется комбинаторная сложность задачи на двух параллельных машинах с единичными длительностями работ. Легко показать, что задачи $W1||C_{max}$, $W1,P||C_{max}$ и $W1,Pm||C_{max}$ при $m \geq 3$ являются NP-трудными в сильном смысле, даже если все работы имеют единичную длительность. Действительно, все эти задачи являются обобщением различных NP-трудных вариантов задачи об упаковке в контейнеры. В [118], Каплан и Амир поставили вопрос о комбинаторной сложности задачи $W1,P2|p_i = 1|C_{max}$. В диссертации показано, что к этой задаче сводится классическая NP-полная в сильном смысле задача ПАРОСОЧЕТАНИЕ С ОГРАНИЧЕНИЯМИ ПО ВЕСУ (задача МР-17 в списке NP-полных задач в [4]), и доказана следующая

Теорема 7 *Задача $W1,P2|p_j = 1,\delta_j \geq 0|C_{max}$ является NP-трудной в сильном смысле.*

Отметим, что задача остается NP-трудной, даже если назначение работ по машинам фиксировано.

В конце раздела рассматривается задача $W1|p_j = 1|C_{max}$, в которой длительности работ единичные и произвольное число работ может выполняться параллельно. Устанавливается, что для нее не существует асимптотического ρ-приближенного алгоритма с $\rho < \frac{4}{3}$ при условии несовпадения классов P и NP. Этот результат подчеркивает, что с точки зрения аппроксимации задача $W1|p_j = 1|C_{max}$ является более сложной, чем классическая задача об упаковке в контейнеры, обобщением которой она является.

Третий раздел посвящен построению и анализу приближенных алгоритмов для задач с воспроизводимым ресурсом, в которых произвольное число работ может выполняться параллельно. В основе этих алгоритмов лежит идея частичного выполнения работ в случае, когда свободного ресурса не хватает для выполнения выбранной работы. При этом получается недопустимое расписание, которое, с одной стороны, дает нижнюю оценку длины оптимального расписания, а с другой стороны, может быть перестроено в допустимое так, что его длина увеличится не более чем в два раза, если длительности всех работ одинаковые. В случае произвольных длительностей та же идея, примененная к округленному примеру, позволяет получить $O(\log n)$-приближенный алгоритм.

Все результаты главы, кроме $O(\log n)$-приближенного алгоритма для задачи $W1||C_{max}$, получены в соавторстве с Б.М.-Т. Лином. Последний результат получен совместно с А. Григорьевым и М. Свириденко.

В третьей главе рассматривается задача построения энергетически эффективных расписаний. Множество работ $\mathcal{J} = \{J_1, \ldots, J_n\}$ должно быть выполнено на одной или нескольких машинах. В однородной модели для каждой работы J_j задан

ее объем W_j и интервал времени, в котором она должна быть выполнена $[r_j, d_j)$. Длительность работы зависит от скорости, с которой она выполняется. Выбор скорости работы каждой машины $S(t)$ определяется при составлении расписания и может меняться с течением времени. Широко известное в инженерных кругах правило кубического корня для устройств, использующих микросхемы CMOS, гласит, что потребляемая мощность пропорциональна третьей степени (кубу) от скорости вычислений. В статьях по теории расписаний, как правило, рассматривается произвольное значение $\alpha > 1$ этого степенного параметра. Расход энергии определяется интегрированием функции мощности на интервале времени работы машины. Требуется выполнить все работы внутри заданных интервалов обслуживания так, чтобы суммарный расход энергии был минимален.

Содержание главы разделено на два больших раздела в зависимости от подхода к построению приближенных алгоритмов.

В первом разделе рассматривается подход, основанный на преобразовании оптимальных решений, в которых есть прерывания работ, в допустимые решения без прерываний. Данный подход является одним из основных для построения приближенных алгоритмов с гарантированной оценкой точности для классических задач теории расписаний [160]. Часто построение оптимального расписания в задаче с прерываниями легче, чем построение оптимального решения в аналогичной задаче без прерываний. Например, как показано в [29, 33], задача $P|pmtn, r_j, d_j|E$ построения энергетически эффективного расписания с прерываниями работ на параллельных идентичных машинах разрешима за полиномиальное время. В свою очередь, аналогичная задача $P|r_j, d_j|E$ без прерываний NP-трудна в сильном смысле [30]. Однако необходимым условием для построения хорошего приближенного решения является близость между собой значений оптимального решения задачи с прерываниями и задачи без прерываний. В разделе 3.1 приведен пример задачи минимизации энергии на одной машине, для которого отношение оптимума расписания без прерываний к оптимуму расписания с прерываниями больше, чем $\frac{n^{\alpha-1}}{2 \cdot 3^{\alpha}}$.

Таким образом, в задачах, рассматриваемых в главе 3, в общем случае оптимум расписания с прерываниями не является хорошей нижней оценкой на оптимум расписания без прерываний. Тем не менее, для частных случаев задачи $1|r_j, d_j|E$ и $P|r_j, d_j|E$ удается преобразовать оптимальное расписание с прерываниями в допустимое расписание без прерываний и оценить в них погрешность расхода энергии в худшем случае.

Основными результатами раздела являются разработка алгоритмов 3.2 и 3.3 и анализ их точности в худшем случае.

Для задачи $1|r_j, d_j|E$ алгоритм 3.2 находит расписание σ с оценкой $E(\sigma) \leq (1 + \frac{W_{\max}}{W_{\min}})^{\alpha} OPT$, которая зависит от отношения максимального объема работы $W_{\max}$ к минимальному $W_{\min}$. Для задачи, в которой все работы имеют одинаковые объемы, алгоритм является 2-приближенным.

Для задачи $P|r_j, d_j|E$, в которой никакие два интервала выполнения работ не вложены друг в друга, предложен 2-приближенный алгоритм 3.3. С использованием оптимального решения задачи с прерываниями, в алгоритме вычисляется длитель-

ность каждой работы, а затем работы упорядочиваются по неубыванию директивных сроков.

Результаты этого раздела получены в соавторстве с Е. Бамписом, Д. Лукарелли, Д. Летсиосом и И. Немпарисом.

Во втором разделе предлагается общий подход к построению приближенных рандомизированных алгоритмов для задач построения энергетически эффективных расписаний, основанный на представлении рассматриваемых задач, как задач целочисленного линейного программирования с большим неполиномиальным числом переменных. По сравнению с релаксациями стандартных задач ЦЛП релаксации задач очень большой размерности дают значительно лучшую нижнюю оценку, однако их решение связано с дополнительными трудностями. Заметим, что если задача линейного программирования имеет экспоненциальное число переменных и полиномиальное число ограничений, то двойственная к ней задача имеет экспоненциальное число ограничений и полиномиальное число переменных. Предположим, что для двойственной задачи можно построить полиномиальный отделяющий оракул, то есть алгоритм, который для заданного решения либо устанавливает его допустимость, либо находит ограничение линейной программы, которое нарушается. Существование полиномиального отделяющего оракула позволяет решить двойственную задачу методом эллипсоидов [102]. В свою очередь, построенное методом эллипсоидов оптимальное решение двойственной задачи позволяет найти оптимальное решение прямой задачи за полиномиальное время, так как значения прямых переменных, соответствующих двойственным ограничениям, которые не были нарушены в ходе решения двойственной задачи, равны нулю. Округление полученного решения задачи ЛП с экспоненциальным числом переменных позволяет находить хорошие приближенные решения для неоднородных задач построения энергетически эффективных расписаний, в которых параметры работ зависят от машин, на которых они выполняются.

Для произвольного действительного параметра $\alpha > 1$ назовем обобщенным числом Белла величину

$$\tilde{B}_\alpha = \sum_{k=0}^{\infty} \frac{k^\alpha e^{-1}}{k!},$$

соответствующую α-му (дробному) моменту случайной величины с распределением Пуассона и математическим ожиданием равным единице. Пусть $\varepsilon > 0$ — произвольное фиксированное число. В разделе 3.2 представлены следующие рандомизированные приближенные алгоритмы для задач построения энергетически эффективных расписаний:

- $(1+\varepsilon)^\alpha \tilde{B}_\alpha$-приближенный алгоритм для неоднородной задачи на параллельных машинах с прерываниями без переноса работ с машины на машину,
- $2^{\alpha-1}(1+\varepsilon)^\alpha \tilde{B}_\alpha$-приближенный алгоритм для задачи на одной машине без прерываний работ,
- $(1+\varepsilon)^\alpha \tilde{B}_\alpha$-приближенный алгоритм для неоднородной цеховой задачи рабочего типа с прерываниями работ.

Для неоднородной задачи на параллельных машинах с прерываниями и переносом работ с машины на машину построен вполне полиномиальный приближенный алгоритм с абсолютной погрешностью ε для любого фиксированного $\varepsilon > 0$.

Результаты раздела 3.2 получены в соавторстве с Е. Бамписом, Д. Лукарелли, Д. Летсиосом и М.Свириденко.

Четвертая глава посвящена задачам построения расписаний с малыми задержками передачи данных. Данные задачи моделируют вычисление на параллельных компьютерах и были впервые описаны в статье Рэйварда-Смита [147] в 1987 г. Множество работ $\mathcal{J} = \{J_1, \ldots, J_n\}$ должно быть выполнено на m идентичных параллельных машинах. На множестве работ задан частичный порядок. Если работы, связанные отношением предшествования, выполняются на разных машинах, то требуется дополнительное время на передачу данных от одной машины к другой. Если обе работы выполняются на одной машине, то считается, что величина задержки несущественна, и она полагается равной нулю. В [108] показано, что задачи минимизации длины расписания (C_{max}) и минимизации взвешенной суммы моментов окончания работ ($\sum w_j C_j$) на неограниченном числе машин ($m > n$) являются NP-трудными, даже если все задержки и длительности всех работ единичные. Поэтому большинство теоретических работ в этом направлении связаны с построением приближенных алгоритмов с гарантированными оценками точности. Основные результаты четвертой главы посвящены задачам с малыми задержками, т.е. когда величина наибольшей задержки не превосходит минимальной длительности работы. Для обозначения задач с малыми задержками во втором поле трехиндексной записи используется сокращение *sct* от термина "small communication time"в английском языке.

- Задача $P\infty|sct|C_{\max}, \sum w_i C_i$ — двухкритериальная задача построения расписания работ на неограниченном множестве параллельных машин с малыми задержками передачи данных.

- Задача $P\infty(P2)|sct|C_{\max}$ — задача минимизации длины расписания работ на неограниченном множестве параллельных двухпроцессорных машин с малыми задержками передачи данных.

В первом разделе изучается задача $P\infty|sct|C_{\max}, \sum w_i C_i$, в которой требуется найти "хорошее" расписание относительно двух критериев C_{max} и $\sum w_j C_j$. В [162] Штайн и Вайн для большого класса задач доказали существование расписаний, для которых значения целевых функций C_{max} и $\sum w_j C_j$ не более чем в два раза отличаются от оптимальных значений этих целевых функций в однокритериальных версиях рассматриваемых задач. В последующем Аслам, Рассала, Штайн и Юнг [37] улучшили эту оценку до значения 1.806.

Допустимое расписание σ относительно критериев f_0 и f_1 называется (α, β)-*расписанием*, если $f_0(\sigma) \leq \alpha f_0(\sigma_0)$ и $f_1(\sigma) \leq \beta f_1(\sigma_1)$, где σ_0 – оптимальное расписание для критерия f_0, и σ_1 – оптимальное расписание для критерия f_1. Алгоритм, строящий такое расписание для любого примера двухкритериальной задачи, называется (α, β)-*приближенным алгоритмом*.

Основным результатом раздела является семейство (α, β)-приближенных алгоритмов для задачи $P\infty|sct|C_{\max}, \sum w_i C_i$ (алгоритм 4.2).

Теорема 20 *Для любого* $\phi \in [0.721, 1.5]$ *алгоритм* 4.2 *является* (α, β)*-приближенным алгоритмом для задачи* $P\infty|sct|C_{max}, \sum w_j C_j$, *где* $\alpha = \frac{4}{3} + \frac{4}{9}\phi$ *и* $\beta = \frac{4}{3-3(1-\frac{2}{3}\phi)^{\frac{3}{2}}}$.

В частности, для $\phi = 0.927$ алгоритм 4.2 находит $(1.746, 1.746)$-расписание.

Полученные результаты обобщаются для задачи, в которой для каждой работы J_j дополнительно задано время доставки q_j и требуется одновременно минимизировать величины $C_j + q_j$ и $\sum w_j(C_j + q_j)$.

Заметим, что рассматриваемые в этом разделе задачи не попадают в список задач Штайна и Вайна, для которых доказано существование (α, β)-расписаний. Следующий результат показывает, что для задачи $P\infty|uct, p_j = 1|C_{max}, \sum w_j C_j$, в которой длительности всех работ и задержки равны между собой, существует (α, β)-расписание лучше чем то, что находит алгоритм 4.2.

Теорема 22 *Для любого* $\phi \in [0, 1]$ *существует* $(1 + \frac{\phi}{2}, \frac{4}{\phi(4-\phi)})$*-приближенное решение для задачи* $P\infty|uct, p_j = 1|C_{max}, \sum w_j C_j$.

Из теоремы 22 следует, что для любого примера задачи $P\infty|uct, p_j = 1|C_{max}, \sum w_j C_j$ существует $(1.445, 1.445)$-расписание.

В конце раздела приближенное решение задачи с неограниченным числом машин используется для получения приближенного решения задачи с заданным (фиксированным) числом машин. В частности, для задачи с единичными задержками предложен (α, β)-приближенный алгоритм 4.3.

Теорема 23 *Для любого* $\phi \in [0, 1.5]$ *алгоритм 4.3 является* (α, β)*-приближенным алгоритмом для задачи* $P|uct, p_j = 1|C_{\max}, \sum w_j C_j$ *со значениями* $\alpha = \frac{7}{3} + \frac{4}{9}\phi$ *и* $\beta = \frac{10}{3} + \frac{4}{3}((1 - \frac{2}{3}\phi)^{-\frac{3}{2}} - 1)^{-1}$.

Результаты этого раздела получены в соавторстве с Е. Бамписом.

Во втором разделе рассматривается задача $P\infty(P2)|sct|C_{\max}$, в которой каждая машина может обрабатывать две работы одновременно. Обозначим через Ф отношение минимальной длительности работы к максимальной задержке. Как и в предыдущем разделе, предполагается, что задержки малы, то есть $\Phi \geq 1$. В [43] показано, что существование ρ-приближенного алгоритма с $\rho < \frac{5}{4}$ для этой задачи влечет совпадение классов P и NP, даже если длительности всех работ и все задержки равны 1. В разделе 4.2 представлен $12(\Phi + 1)/(12\Phi + 1)$-приближенный алгоритм для задачи $P\infty(P2)|sct|C_{\max}$. Алгоритм основан на округлении решения соответствующей задачи линейного программирования. Интересно отметить, что целочисленная постановка этой задачи (ЦЛП) не эквивалентна задаче $P\infty(P2)|sct|C_{\max}$, и оптимальное решение задачи ЦЛП может быть хуже, чем оптимальное решение задачи $P\infty(P2)|sct|C_{\max}$.

Результаты этого раздела получены в соавторстве с Е. Бамписом и Р. Жиродо.

В пятой главе рассматриваются цеховые задачи открытого типа с маршрути-

зацией. Данные задачи являются обобщением двух классических NP-трудных задач дискретной оптимизации: цеховой задачи открытого типа и метрической задачи коммивояжера. Множество работ $\mathcal{J} = \{J_1, \ldots, J_n\}$ должно быть выполнено на m специализированных машинах $M_1, \ldots, M_m$. Каждая работа J_j состоит из m операций $O_{1j}, O_{2,j}, \ldots, O_{m,j}$. Операция O_{ij} выполняется на машине M_i, ее длительность составляет $p_{ij} \in \mathrm{Z}^+$ единиц времени. Работы и машины расположены в узлах транспортной сети $G = (V, E)$, которая задается полным реберно-взвешенным графом. Веса ребер удовлетворяют неравенству треугольника и соответствуют времени перемещения машины из одной вершины в другую. Для выполнения любой работы каждая машина должна переместиться в вершину, в которой эта работа находится. Изначально все машины расположены в одной вершине v_0 и должны вернуться в нее после выполнения всех работ. Все машины перемещаются с одинаковой скоростью, то есть затрачивают на перемещение из вершины v_j в вершину v_k время τ_{jk}. Операции каждой работы могут выполняться в произвольном порядке. Прерывания в процессе выполнения операций запрещены. Различные машины не могут выполнять операции одной работы одновременно, и каждая машина обрабатывает не более одной работы в каждый момент времени. Требуется минимизировать длину расписания, т.е. момент возвращения последней машины в вершину v_0 после выполнения всех работ.

Цеховые задачи, в которых машины перемещаются между работами, расположенными в узлах транспортной сети, впервые были рассмотрены в [38, 39, 40, 41]. Примеры задач, в которых машины должны перемещаться между деталями, возникают при обработке тяжелых или громоздких деталей или при построении расписания работы роботов, которые осуществляют ежедневное техническое обслуживание неподвижных объектов, расположенных в различных частях цеха [39]. Цеховая задача открытого типа с маршрутизацией также возникла при автоматическом планировании маршрутов экскурсий в Национальном Дворцовом музее в городе Тайбэй, входящим в пятерку крупнейших музеев мира [73, 167].

Цеховая задача открытого типа с маршрутизацией является NP-трудной в сильном смысле, даже если работы требуется выполнить одной машиной или все работы лежат в одной вершине. Более того, как показано в [41], эта задача NP-трудна для случая двух машин, когда все работы расположены в узлах сети, состоящей из двух вершин. Поэтому данная глава посвящена построению приближенных алгоритмов для различных вариантов цеховой задачи открытого типа с маршрутизацией.

В первых двух разделах приводятся приближенные алгоритмы для задачи с произвольным числом машин и произвольной транспортной сетью. Первый алгоритм (алгоритм 5.3) основан на простых комбинаторных свойствах рассматриваемой задачи. В нем строится гамильтонов цикл, длина которого не более чем в полтора раза больше длины минимального гамильтонова цикла. Затем n исходных работ преобразуется в $O(\sqrt{m})$ новых работ. Новый пример решается жадным алгоритмом, и полученное расписание перестраивается в допустимое расписание исходной задачи. Алгоритм 5.3 находит расписание, длина которого не более чем в $\left((\sqrt{3}+\sqrt{2})\sqrt{m}+3.5\right)$ раз больше нижней оценки, за время $O(n^3)$. Трудоемкость алгоритма может быть

понижена до $O(n^2)$ за счет увеличения константы перед $\sqrt{m}$.

Алгоритм 5.4, рассматриваемый во втором разделе, является $O(\log m)$-приближенным алгоритмом и имеет асимптотически лучшую точность по сравнению с алгоритмом 5.3. Алгоритм 5.4 основан на сведении цеховой задачи открытого типа с маршрутизацией к цеховой задаче рабочего типа с единичными длительностями работ. Для последней задачи все известные алгоритмы основаны на применении локальной леммы Ловаша и, хотя теоретически их трудоемкость ограничена полиномом от размера входа задачи, их реализация сопряжена со значительными трудностями.

В третьем разделе рассматривается задача с двумя машинами и произвольной транспортной сетью. В [41] для этой задачи предложен $\frac{7}{4}$-приближенный алгоритм. В разделе 5.3 предложен новый приближенный алгоритм (алгоритм 5.7), который имеет ту же трудоемкость, но лучшую оценку качества получаемых решений.

Теорема 28 *Алгоритм* 5.7 *является* 1.625*-приближенным алгоритмом для задачи* $RO2||\tilde{C}_{\max}$*, и время его работы равно* $O(n^3)$.

В следующем разделе рассматривается задача с двумя машинами в предположении, что кратчайший обход вершин транспортной сети может быть найден за полиномиальное время. Для этой задачи предложен (4/3)-приближенный алгоритм, который имеет линейную трудоемкость, если оптимальный обход сети уже известен.

Результаты разделов 5.1, 5.3 и 5.4 получены в соавторстве с С. Севастьяновым и И. Черных.

Основными и наиболее интересными результатами пятой главы являются точный алгоритм динамического программирования и вполне полиномиальная приближенная схема для задачи $RO2||V| = 2|\tilde{C}_{\max}$ с двумя машинами на двухвершинной сети, представленные в разделе 5.5.

Авербах, Берман и Черных [41] доказали, что задача $RO2||V| = 2|\tilde{C}_{\max}$ является NP-трудной в обычным смысле и предложили для ее решения (6/5)-приближенный алгоритм, оставив открытым вопрос о существовании для нее точного псевдополиномиального алгоритма. Построению такого алгоритма и посвящен последний раздел пятой главы.

Первым шагом к построению алгоритма динамического программирования для задачи маршрутизации с двумя машинами на двухвершинной сети является разбиение множества примеров этой задачи на простые и сложные. Обозначим через N_0 — множество работ в вершине v_0 и через N_1 — множество работ в вершине v_1. Напомним, что изначально обе машины расположены в вершине v_0 и должны вернуться в нее после выполнения всех работ. Пусть $l_{\max}$ обозначает максимальную нагрузку машины. Напомним, что каждая работа состоит из двух операций. Обозначим через J_0 работу с максимальной меньшей операцией. Пусть d_0 — сумма длительностей двух операций работы J_0, а λ_R — тривиальная нижняя оценка длины расписания. Тогда следующие примеры задачи $RO2||V| = 2|\tilde{C}_{\max}$ разрешимы за линейное от числа работ время.

Теорема 30 *Если в примере I задачи* $RO2||V| = 2|\tilde{C}_{\max}$ *выполнено одно из двух следующих условий:*

a) $J_0 \in N_0$,

b) $J_0 \in N_1$ *и* $d_0 \geq l_{max}$,

то оптимальное расписание имеет длину λ_R *и может быть построено за время* $O(n)$.

Таким образом, только примеры, в которых работа J_0 лежит в удаленной вершине и ее длина меньше $l_{\max}$, являются NP-трудными. Расписание называется *каноническим*, если множество работ может быть разбито на не более чем восемь непересекающихся подмножеств, для которых выполнены следующие условия:
1) все работы, принадлежащие одному подмножеству, расположены в одной вершине;
2) все работы, принадлежащие одному подмножеству, выполняются блоком на каждой машине;
3) все работы, принадлежащие одному подмножеству, выполняются либо сначала на машине A, потом на машине B, либо наоборот;
4) машины A и B выполняют работы каждого подмножества в одном и том же порядке, более того, этот порядок совпадает с порядком Джонсона для соответствующей цеховой задачи потокового типа на двух машинах.

Показано, что для любого трудного примера существует каноническое оптимальное расписание. Причем, длина канонического расписания зависит только от разбиения исходного множества работ на непересекающиеся подмножества и может быть вычислена за полиномиальное от числа работ время. Используя свойства канонических расписаний, трудный пример задачи $RO2||V| = 2|\tilde{C}_{\max}$ можно решить алгоритмом динамического программирования за время $O(n\Delta^{24})$, где $\Delta = \sum_{J_j \in \mathcal{J}} d_j$.

Используя стандартную технику округления [109], можно трансформировать алгоритм динамического программирования во вполне полиномиальную приближенную схему, т.е. для любого фиксированного $\varepsilon > 0$ построить $(1+\varepsilon)$-приближенный алгоритм, время работы которого ограничено полиномом от числа работ и величины $\frac{1}{\varepsilon}$.

В заключении перечисляются основные результаты диссертации.

Глава 1

Задачи и алгоритмы в теории расписаний

В этой главе вводятся основные термины и понятия, принятые в дискретной оптимизации и теории расписаний, а также обсуждаются подходы к изучению свойств рассматриваемых задач и методы их решения.

1.1 Классификация ресурсов

Задачи теории расписаний и календарного планирования связаны с распределением ограниченных ресурсов для выполнения множества работ и поиском расписания с наилучшим значением заданной целевой функции. Оба направления возникли в 50-х годах прошлого столетия [113], [115], [122], [123], [161] и связаны как с появлением сложных комбинаторных задач по управлению производственными процессами, так и с появлением первых быстродействующих вычислительных устройств, давших надежду решить эти задачи за приемлемое время.

Первые работы [26], [122], [138] по задачам календарного планирования в основном были посвящены построению сетевых графиков, допустимых относительно заданного частичного порядка работ и директивных сроков, и поиску критических путей в них. Однако вскоре основным направлением исследования стали задачи календарного планирования с ограниченными ресурсами [116], [163].

Первоначально в англоязычной литературе различали два основных типа ресурсов: *возобновимые* и *невозобновимые* [66]. Суммарное количество *возобновимого* ресурса, потребляемого в момент времени t, не должно превышать количество ресурса, имеющегося в наличии в этот момент. Под возобновимыми ресурсами можно понимать станки, вычислительные машины, людские ресурсы, производственные помещения и многое другое, что после завершения одной работы может быть использовано для выполнения следующей. *Невозобновимым* ресурсом называется ресурс, который ограничен для всего проекта в целом, и, будучи использованным для выполнения одной работы, не может быть использован для выполнения других работ. Как правило, в задачах календарного планирования с невозобновимым ресурсом предполагается

существование нескольких вариантов его потребления одной работой. Бóльшие затраты ресурса увеличивают скорость выполнения работы и, таким образом, уменьшают ее длительность. К невозобновимым ресурсам принято относить бюджет проекта, строительные материалы и электроэнергию.

Отметим, что разделение ресурсов на возобновимые и невозобновимые удобно с точки зрения приложений, но неудобно с точки зрения классификации задач и типов ресурса. Многие задачи календарного планирования с ограниченными ресурсами не попадают под указанные типы. Неудивительно, что в последующих работах стали появляться новые типы ресурсов. Так, в конце 90-х годов в англоязычной литературе появилось понятие частично возобновимых ресурсов [63], то есть ресурсов, которые доступны в заданном множестве временны́х интервалов. Понятия возобновимых, невозобновимых и частично возобновимых ресурсов стали основой в классификации задач, используемых в англоязычных монографиях [58], [104].

В русскоязычной литературе [2], [3], [13] была принята другая классификация ресурсов. Они делились на складируемые и нескладируемые. Понятие нескладируемого ресурса ничем не отличается от понятия возобновимого ресурса. Ресурсные ограничения складируемого типа формулируются в виде баланса между потребляемым и выделяемым количеством ресурса. Для каждого складируемого ресурса k задана функция $q_k(t) \geq 0$ интенсивности его выделения в заданном интервале времени. В свою очередь для каждой работы j задана интенсивность $r_{jk}(t) \geq 0$ потребления этого ресурса в каждый момент времени ее выполнения. Построенное расписание является допустимым, если для каждого момента времени суммарный по всем работам объем ресурса, потребляемого к этому моменту, не превосходит объема ресурса, выделенного на этот момент времени. Таким образом, понятие складируемого ресурса является обобщением понятия невозобновимого ресурса и позволяет учитывать дополнительные поступления ресурса в течение интервала планирования.

Снятие в последнем определении ограничения на неотрицательность функций $q_k(t)$ и $r_{jk}(t)$, по-видимому, позволяет покрыть все типы ресурсов, встречающиеся в литературе по календарному планированию. Более того, накладывая различные ограничения на функции $q_k(t)$ и $r_{jk}(t)$, можно описать все известные типы ресурсов. Например, пусть p_j – длительность работы j, тогда возобновимый ресурс k может быть описан функциями $q_k(t)$ и $r_{jk}(t)$, равными нулю всюду в их области определения, кроме точек $q_k(0) = \Omega$, $r_{jk}(0) = \alpha$, $r_{jk}(p_j) = -\alpha$ для некоторых положительных Ω и α. Отметим, что отрицательная величина $-\alpha$ указывает на то, что после завершения выполнения работы ресурс, ею потребленный, возвращается и может быть использован для выполнения других работ. Такое альтернативное описание задач календарного планирования с ограниченными ресурсами предложено в частной беседе профессором С. В. Севастьяновым.

Рассмотрим теперь специальный вид ресурса, который будем называть *воспроизводимым* ресурсом. Пусть Ω, α_j и β_j $j = 1, \ldots, n$, — целые положительные числа. Для воспроизводимого ресурса будем считать, что $q(0) = \Omega$ и $q(t) = 0$ при $t > 0$. Пусть для любой работы j интенсивность потребления ресурса задана как $r_j(0) = \alpha_j$, $r_j(p_j) = -\beta_j$ и $r_j(t) = 0$ при $0 < t < p_j$. Неформально это означает, что перед на-

чалом выполнения работа j поглощает α_j единиц ресурса и возвращает β_j единиц ресурса сразу после ее завершения. Если $\beta_j = 0$ для всех работ, то имеем дело с классическим невозобновимым ресурсом. В случае $\alpha_j = \beta_j$ для всех работ получаем классический возобновимый, или нескладируемый ресурс. Таким образом, понятие воспроизводимого ресурса обобщает понятия как возобновимого, так и невозобновимого ресурсов. Основным примером воспроизводимого ресурса являются деньги, или капитал. Вкладывая денежные средства в проект, инвестор надеется получить доход или, хотя бы покрыть часть издержек после выполнения проекта.

1.2 Классификация задач теории расписаний

Полагая в определении возобновимого ресурса $\alpha = 1$, получаем специальный возобновимый ресурс, который принято называть *машиной*. Понятие "машина" играет важную роль, так как огромное число задач дискретной оптимизации может быть представлено как выполнение множества работ на заданном множестве машин.

Например, в 1954 году С.М. Джонсон [115] рассмотрел следующую задачу. Задано множество работ и две машины. Для каждой работы определены ее длительности на первой и второй машинах. Каждая работа сначала должна быть выполнена на первой, а затем на второй машине, и каждая машина не может обслуживать две работы одновременно. Прерывания в процессе выполнения работ на каждой из машин запрещены. Требуется построить расписание работ минимальной длины. Предполагается, что все работы могут начать обслуживание в нулевой момент времени, и длиной расписания называется момент окончания выполнения последней работы. Как показано в [115], оптимальная перестановка работ в задаче Джонсона может быть найдена за полиномиальное от числа работ время.

Область дискретной оптимизации, в которой для выполнения работ используются машины, получила название — теория машинных расписаний (machine scheduling) в англоязычной литературе [131, 132], а в русскоязычной литературе — теория расписаний [24].

В задачах теории расписаний задано множество работ $\mathcal{J} = \{J_1, \ldots, J_n\}$ и множество машин $\mathcal{M} = \{M_1, \ldots, M_m\}$. Для каждой работы J_j, $j = 1, \ldots, n$, и машины M_i, $i = 1, \ldots, m$, *расписание* определяет один или несколько времéнных интервалов, в течение которых работа J_j выполняется на машине M_i. Расписание называется *допустимым*, если оно удовлетворяет различным требованиям рассматриваемой задачи. Каждое допустимое расписание σ однозначно определяет для каждой работы J_i момент начала $s_i(\sigma)$ и момент завершения $C_i(\sigma)$ ее обслуживания. В дальнейшем в записи этих моментов будем опускать символ σ, если из контекста понятно, о каком расписании идет речь. Допустимое расписание называется *активным*, если ни одна работа не может быть выполнена раньше без изменения расписания для других работ и нарушения допустимости расписания. Далее во всех задачах будет предполагаться, что машина не может выполнять две работы одновременно.

Другие требования связаны с заданным составом машин (конфигурацией машин)

и условиями обслуживания ими работ, с характеристиками самих работ и с различными критериями оценки качества расписаний. Сочетание различных условий определяет огромное число задач, возникающих в теории расписаний.

Ниже будут описаны конфигурации машин, характеристики работ и виды целевых функций, которые будут рассмотрены в последующих разделах, и адаптирована система обозначений, введенная в [131].

1.2.1 Конфигурации машин

Рассмотрим два принципиально разных типа конфигураций машин: параллельные и специализированные. В задачах первого типа каждая работа должна быть выполнена на одной из идентичных параллельных машин. Другими словами, для выполнения множества работ задается m единиц возобновимого ресурса, и каждая работа использует единицу этого ресурса в течение всего интервала, когда она выполняется. В случае $m = 1$ получим задачу на одной машине. Задачи такого типа относятся к одностадийным задачам теории расписаний [21].

В свою очередь задачи на специализированных машинах делятся на два типа: многостадийные и многопроцессорные. В многопроцессорных задачах каждая работа требует одновременного обслуживания данным фиксированным подмножеством машин. В некоторых задачах рассматриваются различные способы выполнения одной работы. Каждый способ определяет набор машин, требуемых для выполнения данной работы, и ее длительность. В параграфе 2.2.3 будет рассмотрен частный случай, когда для каждой работы фиксирована ровно одна машина, на которой она должна быть выполнена.

В многостадийных задачах теории расписаний работы состоят из множества операций [23]. В англоязычной литературе такие задачи называют *цеховыми задачами* (shop scheduling problem) [62]. Каждая работа J_j состоит из ν_j операций $O_{1,j}, \dots O_{\nu_j,j}$. Каждая операция $O_{i,j}$ должна быть выполнена на определенной машине $M \in \mathcal{M}$. При этом две операции одной работы не могут выполняться одновременно. Нетрудно заметить, что в такой постановке операции играют роль работ, для выполнения которых требуется по единице двух однотипных возобновимых ресурсов, соответствующих данной операции, один из них называется "машиной", а другой — "работой". Несмотря на некоторую путаницу в терминологии, будем придерживаться терминов, исторически сложившихся для цеховых задач теории расписаний.

Дальнейшее различие в цеховых задачах основано на ограничениях на порядок выполнения операций одной работы. Исторически в теории расписаний рассматриваются две следующие основные модели. В *цеховой задаче открытого типа* (OPEN SHOP) порядок операций может быть выбран произвольным. Будем называть такие работы *работами открытого типа*. В классической первоначальной постановке предполагается, что каждая работа состоит ровно из одной операции на каждой машине. Цеховую задачу, в которой работы открытого типа состоят из произвольного числа операций на каждой машине, называют обобщенной цеховой задачей открытого типа. В *цеховой задаче рабочего типа* (JOB SHOP) операции каждой работы

должны быть выполнены в заданном линейном порядке, такие работы будем называть *работами рабочего типа*. В литературе часто рассматривается ее подзадача — *цеховая задача потокового типа* (FLOW SHOP), в которой число операций каждой работы совпадает с числом машин, операция $O_{i,j}$ работы J_j должна быть выполнена на машине M_i и операция $O_{i,j}$ предшествует операции $O_{k,j}$, если $i < k$.

1.2.2 Характеристики работ

Основной характеристикой работы J_j является ее объем W_j. Время выполнения (длительность) работы J_j зависит от ее объема и скорости S, с которой она выполняется, и равно $p_j = W_j/S$. В третьей главе рассматриваются задачи, в которых объем работы зависит от того на какой машине она выполняется, и скорость выполнения может изменяться в каждый момент времени. Однако для большинства задач теории расписаний объем работы и скорость ее выполнения — величины постоянные, и без ограничения общности предполагается, что $p_j = W_j$. Аналогично определяются объем $W_{i,j}$ и длительность операций $p_{i,j}$ в многостадийных задачах теории расписаний.

Кроме объема и длительности, каждая работа характеризуется интервалом времени, в котором она может выполняться, различными зависимостями от других работ и дополнительными условиями на ее выполнение. Совокупность всех характеристик или условий можно назвать технологией выполнения работы.

Пусть $(r_j, d_j]$ — интервал времени, в котором работа J_j должна быть выполнена, тогда момент r_j называют временем (моментом) поступления работы J_j, а момент d_j — ее директивным сроком. Заметим, что в некоторых задачах эти величины задают желаемые моменты начала и завершения работы, и отклонение от них в ту или иную сторону не нарушает допустимости расписания, а лишь увеличивает значение соответствующей целевой функции.

Если прерывания в процессе выполнения работ или операций запрещены, то в любом допустимом расписании работа (операция) должна целиком выполняться на одной машине в одном интервале времени, и момент ее завершения определяется как сумма момента начала ее обслуживания и ее длительности. Кроме того будем различать два способа выполнения работ, если прерывания работ разрешены. В первом случае работа (операция) должна целиком выполняться на одной машине, но ее обработка может быть прервана и продолжена позднее. Во втором случае после прерывания работа может быть продолжена на любой другой машине. Конечно, эти условия эквивалентны, если за каждой работой или операцией закреплена одна машина, на которой она должна выполняться.

Предположим, что множество работ $\mathcal{J}$ разбито на набор непересекающихся подмножеств (групп) $\mathcal{J}_1, \ldots \mathcal{J}_K$. После выполнения некоторой работы из группы $\mathcal{J}_j$ машине M_i требуется τ_{ijk} времени, чтобы начать выполнение любой работы из группы $\mathcal{J}_k$. Величина τ_{ijk} называется временем переналадки машины. Обычно предполагается, что машине не требуется переналадка, если она выполняет работы из одной группы, то есть $\tau_{ijj} = 0$ для всех $i = 1, \ldots, m$, $j = 1, \ldots, n$. Дополнительно пред-

положим, что для каждой машины задано исходное (нулевое) состояние, с которого она начинает работу и в которое она должна вернуться после завершения всех работ. Специальный класс задач с переналадкой машин будет рассмотрен в последней главе диссертации.

Зависимости между работами также возникают, когда на множестве работ задан частичный порядок их выполнения. Если работа J_i предшествует работе J_j, то работа J_j не может стартовать раньше, чем через δ времени после окончания работы J_i. Величина δ называется задержкой и может зависеть как от работ J_i и J_j, так и от пары машин, на которых они выполнялись. Такая ситуация будет рассмотрена в четвертой главе диссертации.

1.2.3 Целевые функции

В рассматриваемых задачах обычно существует много допустимых расписаний. В каждом из них можно вычислить момент завершения каждой работы или операции. В цеховых задачах моментом завершения работы называют момент завершения ее последней операции. Наиболее популярным критерием качества в теории расписаний является минимизация момента завершения последней работы (длина расписания), $C_{max}(\sigma) = \max\limits_{J_i \in \mathcal{J}} C_j(\sigma)$. Как и критерий C_{max}, другие общепринятые критерии в теории расписаний зависят от моментов завершения работ.

В диссертации будут рассматриваться следующие критерии качества расписания: минимизация максимального запаздывания, минимизация взвешенной суммы моментов окончания работ, минимизация суммарного запаздывания. Первый критерий обозначается через L_{max}, и $L_{max}(\sigma) = \max_{J_i \in \mathcal{J}}\{(C_j(\sigma) - d_j)\}$. Второй критерий обозначается через $\sum w_j C_j$, и его значение на расписании σ равно $\sum\limits_{J_i \in \mathcal{J}} w_j C_j(\sigma)$, где w_j — вес работы J_j. Если $w_j = 1$ для всех работ, то второй критерий принято называть суммой моментов окончания работ и обозначать через $\sum C_j$. Соответственно, третий критерий обозначается через $\sum w_j L_j$, и его значение определяется как $\sum_{J_i \in \mathcal{J}} w_j (C_j(\sigma) - d_j)$.

В последней главе каждой машине M_i после завершения выполнения всех работ требуется время, чтобы вернуться в исходное состояние. Обозначим через $\widetilde{C}_i(\sigma)$ момент времени, после которого машина M_i не совершает никаких действий, а именно не выполняет работы, переналадку или возвращение в исходное состояние. Назовем величину $\widetilde{C}_{max}(\sigma) = \max\{\widetilde{C}_i(\sigma) | i = 1, \ldots, m\}$ *длиной расписания*, и критерий минимизации длины расписания обозначим через $\widetilde{C}_{max}$.

Заметим, что для всех задач, кроме рассматриваемых в последней главе, в любом расписании σ величина $\widetilde{C}_{max}(\sigma)$ совпадает с $C_{max}(\sigma)$.

В третьей главе рассматриваются задачи, в которых требуется минимизировать общий расход энергии E, т.е. суммарную энергию, потраченную на выполнение всех работ. Этот критерий принципиально отличается от критериев, описанных выше. Он зависит не от моментов завершения работ, а от скорости их выполнения. Конкретная формула вычисления общего расхода энергии будет приведена в главе 3.

1.2.4 Задачи теории расписаний с ограничениями на ресурсы

Появившись в 50-х годах прошлого столетия, теория расписаний и календарное планирование длительное время фактически развивались параллельно. Задачи календарного планирования близки к реальным задачам, возникающим в приложениях, и в их постановках рассматривалось много однородных или разнородных ресурсов. При этом технология выполнения работ, как правило, ограничивалась заданием общего интервала планирования и отношением предшествования между работами. В свою очередь в классических задачах теории расписаний фактически не рассматривались ресурсы, отличные от ресурса "машина", а основной упор делался на различные технологии выполнения множества работ и на разнообразие целевых функций. В частности, эту ситуацию отражает система обозначений для задач теории расписаний, введенная в конце 70-х годов [99].

1.2.5 Система обозначений

В конце 70-х годов Грехем, Лоуле, Ленстра и Ринной Кан [99] предприняли попытку ввести общие обозначения для всех задач теории расписаний. Для обозначения каждой задачи они предложили использовать $\alpha|\beta|\gamma$-идентификатор, в котором в поле α вносится информация о типе ресурсов задачи, в том числе о конфигурации машин, в поле β описываются характеристики работ, и в поле γ — вид целевой функции.

Пусть $\circ$ обозначает пустой символ. Первое поле имеет вид $\alpha = \alpha_1\alpha_2\alpha_3$, где α_1, α_2 и α_3 обозначают следующее.

- $\alpha_1 \in \{\circ, W1\}$:
  - $\alpha_1 = \circ$: нет ресурсов отличных от машин;
  - $\alpha_1 = W1$: общий воспроизводимый ресурс.
- $\alpha_2 \in \{\circ, P, D, O, \hat{O}, F, J, JO\}$:
  - $\alpha_2 = \circ$: одна машина;
  - $\alpha_2 = P$: идентичные параллельные машины;
  - $\alpha_2 = D$: специализированные машины;
  - $\alpha_2 = O$: цеховая задача открытого типа;
  - $\alpha_2 = \widehat{O}$: обобщенная цеховая задача открытого типа;
  - $\alpha_2 = F$: цеховая задача поточного типа;
  - $\alpha_2 = J$: цеховая задача рабочего типа;
- $\alpha_3 \in \{\circ, m\}$:
  - $\alpha_3 = \circ$: число машин произвольно и является частью входа задачи;
  - $\alpha_3 = m$: число машин фиксировано и равно m;

– $\alpha_3 = \infty$: число машин больше чем число работ.

Если рассматривается задача на одной машине, то $\alpha_2 = \circ$ и $\alpha_3 = 1$. Если в задаче с воспроизводимым ресурсом машины отсутствуют, то $\alpha_2 = \circ$, и $\alpha_3 = \circ$.

Второе поле указывает на следующие характеристики работ.

- $\beta_1 \in \{\circ, s - batch\}$:
  - $\beta_1 = \circ$: множество работ независимо;
  - $\beta_1 = s - batch$: множество работ разбито на группы, и последовательное выполнение на одной машине работ из разных групп требует переналадки машины.
- $\beta_2 \in \{\circ, prec, ct\}$:
  - $\beta_2 = \circ$: порядок выполнения работ произвольный;
  - $\beta_2 = prec$: на множестве работ задан частичный порядок, и $\delta_{ij} = 0$ для всех работ.
  - $\beta_2 = ct$: на множестве работ задан частичный порядок, и $\delta_{ij} = 0$, если работы J_i и J_j выполняются на одной машине.
- $\beta_3 \in \{\circ, pmtn, pmtn^*\}$:
  - $\beta_3 = \circ$: работы должны быть выполнены без прерываний;
  - $\beta_3 = pmtn$: разрешены прерывания в процессе выполнения любой работы и перенос работы с одной машины на другую;
  - $\beta_3 = pmtn^*$: разрешены прерывания в процессе выполнения любой работы, но каждая работа должна обслуживаться только одной машиной.
- $\beta_4 \in \{\circ, r_j\}$:
  - $\beta_4 = \circ$: все работы доступны для выполнения в начальный момент времени;
  - $\beta_4 = r_j$: для каждой работы задан момент ее поступления.
- $\beta_5 \in \{\circ, d_j\}$:
  - $\beta_5 = \circ$: директивные сроки не заданы;
  - $\beta_5 = d_j$: заданы директивные сроки.

  Если $\beta_4 = r_j$ и $\beta_5 = d_j$, то $\beta_6 = \uparrow\uparrow$ означает, что среди интервалов $[r_j, d_j]$ нет целиком вложенных друг в друга, то есть $r_i < r_j$ влечет $d_i \leq d_j$.
- $\beta_7 \in \{\circ, p_j = 1, p_{ij} = 1\}$:
  - $\beta_7 = \circ$: произвольные длительности работ;

- $\beta_7 = p_j = 1$: все работы имеют единичную длительность;
- $\beta_7 = p_{i,j} = 1$: все операции всех работ имеют единичную длительность.

Для задач с воспроизводимым ресурсом

- $\beta_8 \in \{\circ, \delta_j \leq 0, \delta_j \geq 0\}$:
  - $\beta_8 = \circ$: произвольные прибыли работ;
  - $\beta_8 = \delta_j \leq 0$: все работы имеют отрицательную прибыль;
  - $\beta_8 = \delta_j \geq 0$: все работы имеют положительную прибыль.

  И наконец, третье поле γ включает одну или несколько целевых функций из множества $\{C_{max}, \widetilde{C}_{max}, \sum C_j, \sum w_j C_j, L_{max}, \sum w_j L_j\}$.

1.3 Вычислительная сложность

Как и в других разделах математики, задачи комбинаторной оптимизации можно разделить на простые и сложные. Однако, в отличие от других областей сложность каждой задачи определяется не суммарным количеством времени, потраченным математиками на ее решение, а конкретным количеством секунд, минут или часов, за которое удается решить каждую индивидуальную задачу определенного размера на компьютере. Например, задача Джонсона $F2||C_{max}$ на двух машинах с десятью тысячами работ решается на современном компьютере за секунды, а цеховая задача рабочего типа $J||C_{max}$ с 30 работами и 30 машинами требует для своего решения нескольких часов. Конечно, причина такой ситуации может заключаться в том, что для первой задачи известен хороший алгоритм ее решения, а для второй таких алгоритмов неизвестно. Однако теория вычислительной сложности дает математическое обоснование того, что такое различие между задачами неслучайно, не зависит от талантов математического сообщества и обосновывает разделение задач на простые и сложные.

В теории вычислительной сложности рассматриваются задачи распознавания, то есть задачи, в которых для каждого ее примера требуется дать ответ ДА или НЕТ. Поскольку любую задачу дискретной оптимизации, используя бинарный поиск, можно представить как относительно короткую последовательность задач распознавания, то сложности задачи оптимизации и соответствующей ей задачи распознавания как правило совпадают.

Пусть П — некоторая задача распознавания. Обычно задача П содержит несколько параметров, значения которых не определены. *Индивидуальная задача* (*пример*) получается из задачи П, если всем параметрам задачи П присвоить конкретные значения.

Каждый вход индивидуальной задачи можно представить как последовательность целых чисел, для которых будем различать две основные схемы кодировки. Первая схема называется унарной кодировкой и требует $O(k)$ бит для кодировки

числа k. Вторая схема называется бинарной или стандартной кодировкой и требует $O(\log k)$ бит для кодировки числа k. Она используется для представления чисел в любом стандартном компьютере. Обозначим через x вход некоторого примера рассматриваемой оптимизационной задачи. Тогда длина этого входа $|x|$ определяется как общее число бит, используемых в кодировке входа x при заданной схеме кодирования.

Алгоритмом называется пошаговая процедура, решающая вычислительную задачу. Для данного входа x он вычисляет правильный ответ $f(x)$ после конечного числа шагов. *Временной сложностью алгоритма* называется функция, которая каждой входной длине $|x|$ ставит в соответствие максимальное (по всем примерам) время, то есть общее число элементарных операций, затрачиваемое алгоритмом на решение примеров этой длины. *Полиномиальным алгоритмом* называется алгоритм, временная сложность которого ограничена полиномом от длины входа. Задачи, для которых известен точный полиномиальный алгоритм, называются *полиномиально разрешимыми*. Класс, содержащий все такие задачи, называется классом P.

Если задача оптимизации формулируется как задача распознавания, то для тех примеров, в которых ответ ДА, часто существует короткий сертификат, который удостоверяет этот ответ. Например, ответом на вопрос, существует ли в задаче Джонсона расписание длины не больше заданного числа, служит само расписание. Класс NP содержит все задачи распознавания, которые удовлетворяют двум следующим условиям.

1. Для каждого ДА-примера x существует сертификат y, размер которого полиномиально ограничен от $|x|$.

2. Существует полиномиальный алгоритм, который проверяет, действительно ли y является сертификатом для данного примера x.

Очевидно, что класс NP содержит класс P. Вопрос о несовпадении классов P и NP является одной из важнейших открытых проблем современной математики. Большинство исследователей в данной области придерживаются мнения, что класс NP шире класса P.

Наиболее трудными задачами в классе NP являются NP-полные задачи. Неформально говоря, задача П является NP-полной, если любая другая задача в NP может быть решена за полиномиальное время алгоритмом, который делает полиномиальное число обращений к процедуре решения задачи П. Последнее означает, что если существует полиномиальный алгоритм, решающий NP-полную задачу, то существуют полиномиальные алгоритмы для всех задач из NP. В настоящее время доказано, что огромное число задач распознавания являются NP-полными, в том числе упомянутая выше задача $J||C_{max}$. Заметим, что концепция полиномиальной разрешимости и NP-полноты зависит от используемой схемы кодирования. Если заменить схему кодирования с бинарной на унарную, то задача может стать легче, так как размер входа увеличится, и ограничения на время работы алгоритма станут менее жесткими. Задача, которая остается NP-полной и в унарной кодировке,

называется сильно NP-полной, или NP-полной в сильном смысле. Задача, которая может быть решена за полиномиальное время от ее размера в унарной кодировке, называется псевдо-полиномиально разрешимой, а алгоритм ее решения называется псевдо-полиномиальным алгоритмом.

Подробнее с теорией вычислительной сложности можно ознакомиться по монографиям [4], [15], [93]. Задачи, к решению которых полиномиально сводится некоторая NP-полная задача, называются NP-трудными.

В диссертации используется общепринятая техника доказательства NP-трудности задач дискретной оптимизации. Пусть в множестве X допустимых расписаний требуется найти расписание σ^*, которому соответствует наименьшее значение целевой функции $F(\sigma)$, $\sigma \in X$. Сформулируем соответствующую задачу распознавания: существует ли допустимое расписание σ' такое, что $F(\sigma') \leq y$ для заданного числа y. Очевидно, что указанное расписание σ' существует тогда и только тогда, когда $F(\sigma^*) \leq y$. Следовательно, задача оптимизации не легче, чем соответствующая ей задача распознавания, и из NP-трудности (NP-полноты) задачи распознавания следует NP-трудность исходной экстремальной задачи. Для доказательства NP-трудности задачи распознавания достаточно построить полиномиальное сведение к ней известной NP-полной или NP-трудной задачи. Если эталонная задача является NP-полной в сильном смысле, то достаточно построить псевдо-полиномиальное сведение и тем самым убедиться в NP-трудности в сильном смысле исходной задачи.

1.4 Приближенные алгоритмы

Одним из подходов, связанных с решением труднорешаемых задач, является разработка приближенных алгоритмов. Методы, применяемые для построения алгоритмов, сильно зависят как от специфики задачи, так и от выбранного критерия оценки качества работы приближенного алгоритма. Широко известны эвристические алгоритмы, качество работы которых оценивается и сравнивается на основе сочетания эмпирических данных и аргументов, опирающихся на здравый смысл. Другой критерий связан с требованием доказать, что решения, получаемые приближенным алгоритмом, всегда (или в среднем) отличаются от оптимального не более чем на определенную величину. В диссертации будут рассматриваться только приближенные алгоритмы, для которых удается априори оценить качество получаемого ими решения для наихудшего из возможных примера.

Для задачи теории расписаний, в которой требуется минимизировать целевую функцию $F(\sigma) \geq 0$, алгоритм A называется *ρ-приближенным алгоритмом*, если $F(\sigma_H(I)) \leq \rho F(\sigma^*(I))$ для всех примеров I. Другими словами, алгоритм A гарантированно находит решение не более чем в ρ раз хуже оптимального. Если существует константа c такая, что $F(\sigma_H(I)) \leq \rho F(\sigma^*(I)) + c$ для всех примеров I, то алгоритм A называется *асимптотическим ρ-приближенным алгоритмом.* Величину ρ будем называть оценкой точности алгоритма.

Семейство алгоритмов A_ε для задачи минимизации называется *полиномиальной*

приближенной схемой, если для каждого $\varepsilon > 0$ алгоритм A_ε является $(1+\varepsilon)$-приближенным алгоритмом и время его работы ограничено полиномом от размера входа индивидуальной задачи.

Семейство алгоритмов A_ε для задачи минимизации называется *вполне полиномиальной приближенной схемой*, если для каждого $\varepsilon > 0$ алгоритм A_ε является $(1+\varepsilon)$-приближенным алгоритмом и время его работы полиномиально от размера входа и величины $\frac{1}{\varepsilon}$.

1.5 История и основные этапы развития теории расписаний

Развитие теории расписаний шло параллельно развитию комбинаторной оптимизации, частью которой она является. Первые работы по теории расписаний появились в середине 50-х годов и были посвящены определению правил, которые позволяли находить оптимальные расписания для некоторых простых моделей. Например, в 1955 г. Джексон [112] показал, что оптимальное решение задачи минимизации максимального запаздывания работ, выполняемых на одной машине, получается упорядочением работ согласно их директивным срокам. В 1956 г. Смит [161] установил, что упорядочение работ по невозрастанию отношения длительности работы к ее весу ведет к оптимальной перестановке в задаче минимизации взвешенной суммы моментов окончания работ на одной машине. В то же время формировались постановки задач, включая и выбор критериев оценки качества расписания. С середины 50-х годов до середины 70-х годов прошлого столетия основные усилия исследователей были сосредоточены на построении точных эффективных алгоритмов, то есть алгоритмов со временем работы, ограниченным некоторым полиномом от длины бинарной записи входной информации задачи.

Для сравнительно небольшого количества задач комбинаторной оптимизации эти усилия увенчались успехом. В частности, были решены задача нахождения кратчайшего пути между двумя вершинами в произвольном графе, задача построения остовного дерева минимального веса, задача нахождения максимального потока в ориентированной и неориентированной сети и некоторые другие. В 1965 г. Д. Эдмондс построил красивый полиномиальный алгоритм для решения задачи нахождения совершенного паросочетания в произвольном (не двудольном) графе [85]. В теории расписаний в эти годы были построены полиномиальные алгоритмы для цеховых задач на двух машинах в системах потокового [115] и открытого типа [97] и для ряда одностадийных задач. В середине 60-х годов двадцатого столетия Конвей, Максвелл и Миллер опубликовали первую монографию, посвященную задачам теории расписаний [79], а в середине 70-х вышла первая монография по теории расписаний на русском языке, авторами которой стали Танаев и Шкурба [24]. К концу 70-х поток статей, посвященных задачам теории расписаний, значительно вырос. Во многих сформулированных ранее задачах возникали дополнительные ограничения, которые драматически влияли на их сложность и требовали новых алгоритмов решения. В

1979 году в обзорной статье, посвященной задачам теории расписаний, Грехем, Лоуле, Ленстра и Ринной Кан [99] ввели классификацию задач теории расписаний и представили новую систему обозначений этих задач, которая, несмотря на многочисленные последующие модификации, сохранилась по сегодняшний день.

В уже упомянутой статье [85] Эдмондс подчеркнул важность различия между задачами, для которых известны точные полиномиальные алгоритмы и для которых известны только точные экспоненциальные алгоритмы, основанные на методах неявного перебора. Более того, он выдвинул гипотезу, что для большинства труднорешаемых задач нельзя построить полиномиальные алгоритмы. В 1971 году Кук формально определил класс NP и ввел понятие NP-полноты [80]. Он показал, что задача о выполнимости является NP-полной, то есть она так же трудна как и труднейшая задача в этом классе. Спустя некоторое время Карп и Левин независимо доказали NP-полноту ряда комбинаторных проблем, таких как задача о клике, задача о вершинном покрытии, задача коммивояжера, задача о покрытии множествами, задача об упаковке в контейнеры, задача о разбиении и многие другие. Эти результаты породили лавину доказательств NP-полноты как известных на тот момент, так и новых задач комбинаторной оптимизации. Почти все задачи распознавания, для которых не были найдены полиномиальные алгоритмы, оказались NP-полными. В конце 70-х Гэри и Джонсон опубликовали монографию [93], которая содержала список из более чем трехсот NP-полных задач. Целый параграф в этом списке посвящен исключительно NP-полным задачам теории расписаний. Авторы монографии упомянули около двадцати наиболее известных задач в этой области. С тех пор число известных NP-полных задач выросло многократно. Большинство задач теории расписаний, сформулированных как задачи распознавания, также оказались NP-полными. В конце 80-х Лоуле, Ленстра, Ринной Кан и Шмойс опубликовали обзор по вычислительной сложности задач теории расписаний [131]. Еще более обширные монографии по сложности задач теории расписаний были опубликованы в разные годы на русском языке группой математиков из Института технической кибернетики АН БССР под руководством академика Танаева [21, 23, 22].

Первый приближенный алгоритм с гарантированной оценкой точности для задачи теории расписаний появился еще в 1966 году [98], когда Грехем проанализировал простую жадную эвристику для задачи $P||C_{max}$. С появлением теории NP-полноты выяснилось, что большинство задач оптимизации являются NP-трудными. Однако, хотя актуальность в построении приближенных алгоритмов для задач теории расписаний стала очевидной, число публикаций, посвященных таким алгоритмам, в 70х - 80х годах прошлого столетия было сравнительно невелико. Приведем несколько наиболее интересных примеров по аппроксимации задач теории расписаний, полученных в эти два десятилетия. Поттс построил 3/2-приближенный алгоритм для задачи минимизации максимального запаздывания работ, выполняемых на одной машине и имеющих различные моменты поступления [146]. В конце 70-х годов Сани [149] и Генс и Левнер [94, 95] разработали первые вполне полиномиальные приближенные схемы для различных задач комбинаторной оптимизации, в том числе Генс и Левнер предложили вполне полиномиальную приближенную схему для задачи $1||\sum w_j U_j$.

Ибарра и Ким [110] построили вполне полиномиальную приближенную схему для задачи $1||\sum U_j$, в которой на множестве работ задан древесный частичный порядок. Коффман, Гэри и Джонсон предложили оригинальный приближенный алгоритм для задачи минимизации длины расписания множества независимых работ на параллельных машинах (задача $P||C_{\max}$). Их алгоритм использует двойственность задачи $P||C_{\max}$ и задачи об упаковке в контейнеры. Используя бинарный поиск, алгоритм ищет минимальный размер для m контейнеров, в которые можно упаковать n предметов. Коффман, Гэри и Джонсон [77] показали, что если упорядочить предметы по невозрастанию размеров и использовать правило: клади предмет в первый подходящий контейнер, то для задачи $P||C_{\max}$ получится расписание длины не более чем в $1.22+2^{-k}$ раз больше оптимального, где k — число итераций бинарного поиска. Через девять лет Хочбаум и Шмойс [106], используя двойственность задачи $P||C_{\max}$ и задачи об упаковке в контейнеры, построили для задачи $P||C_{\max}$ полиномиальную приближенную схему.

Интересные и важные результаты были получены в Институте Математики СО РАН профессором Севастьяновым. Для цеховых задач теории расписаний он разработал алгоритмы с абсолютной оценкой погрешности в терминах длины максимальной операции, основанные на построении приближенного решения для соответствующих задач о компактном суммировании векторов [16, 17, 18, 155]. В конце 90-х годов, Свириденко, Солис-Оба и Янсен, используя алгоритм Севастьянова, построили для цеховой задачи рабочего типа $Jm||C_{\max}$ полиномиальную приближенную схему.

Все упомянутые выше результаты внесли важный вклад в развитие методов построения приближенных алгоритмов и послужили основой для большого числа результатов, полученных в последующие годы. Тем не менее основная часть исследователей работала над доказательством NP-трудности многочисленных задач теории расписаний и над выделением полиномиально разрешимых подклассов NP-трудных задач.

Интерес к построению приближенных алгоритмов резко возрос в 90-е годы. Возможно, частично это связано с выходом в 1991 году статьи Пападимитриоу и Янакакиса [143], в которой они представили первый теоретический подход к изучению вычислительной сложности построения приближенных алгоритмов для задач оптимизации. В частности, они ввели определения классов сложности для задач оптимизации, дали определение сводимости задач оптимизации, сохраняющей аппроксимацию, и доказали, что некоторые известные и важные задачи оптимизации, включая максимизационную версию задачу о 3-Выполнимости, являются полными относительно этой сводимости. В последующие несколько лет было установлено, что многие другие оптимизационные задачи также полны относительно этой сводимости. Статья Пападимитриоу и Янакакиса подхлестнула интерес исследователей к построению приближенных алгоритмов, и в течение последних 20 лет было получено много важных и интересных результатов.

Выяснилось, что несмотря на эквивалентность большинства труднорешаемых задач с точки зрения нахождения точного решения, они сильно отличаются в сложности построения для них приближенных алгоритмов с гарантированной оценкой точ-

ности. Например, для задачи о вершинном покрытии известен простой 2-приближенный алгоритм, для задачи о рюкзаке для любого $\varepsilon > 0$ существует $(1+\varepsilon)$-приближенный алгоритм с временем работы ограниченным полиномом от числа предметов и величины $\frac{1}{\varepsilon}$ [109], а для задачи о покрытии или задачи раскраски графов в минимальное число цветов никаких полиномиальных алгоритмов с оценкой точности, ограниченной константой, неизвестно. Схожая картина наблюдается и для задач теории расписаний.

- $P2||C_{max}$: Задача построения минимального по длине расписания на двух машинах является специальным случаем задачи о рюкзаке и, следовательно, для нее для любого $\varepsilon > 0$ существует $(1 + \varepsilon)$-приближенный алгоритм с временем работы, ограниченным полиномом от числа работ и величины $\frac{1}{\varepsilon}$.

- $P||C_{max}$: Для задачи построения минимального по длине расписания на произвольном числе машин в [106] предложен $(1+\varepsilon)$-приближенный алгоритм с временем работы $O(n^{\frac{1}{\varepsilon}})$.

- $O||C_{max}$: Для задачи построения минимального по длине расписания в цеховой задаче открытого типа любой жадный алгоритм строит расписание не более чем в два раза длиннее оптимального [53].

- $J||C_{max}$: Для задачи построения минимального по длине расписания в цеховой задаче рабочего типа неизвестно полиномиального алгоритма с оценкой точности, ограниченной константой. Лучший известный детерминированный алгоритм [150] строит расписание длины не более чем в $\frac{\log^2(m\mu)}{\log\log(m\mu)}$ раз хуже оптимальной, где μ – максимальное число операций одной работы.

Как видно из последнего результата, с точки зрения аппроксимации цеховые задачи, по-видимому, относятся к числу наиболее сложных задач в теории расписаний. Действительно, в [164] показано, что существование приближенных алгоритмов с погрешностью $\rho = 5/4$ для задач $F||C_{max}$, $J||C_{max}$ и $O||C_{max}$ влечет совпадение классов P и NP. Отметим, что на данный момент для задач $F||C_{max}$ и $J||C_{max}$ неизвестно никаких приближенных алгоритмов, которые для любого примера находят решение не более чем в ρ раз хуже оптимального, где ρ — некоторая заданная константа. Однако, для случая когда число машин фиксировано, для многих постановок цеховых задач удается построить приближенные полиномиальные схемы [103, 114, 129, 157].

Отметим, что параллельно с построением приближенных алгоритмов с гарантированной оценкой точности развиваются и методы, позволяющие строить условные нижние оценки на аппроксимируемость рассматриваемых задач теории расписаний. Условность таких результатов связана с тем, что для их получения используются различные правдоподобные гипотезы, основной из которых является гипотеза о несовпадении классов P и NP.

Необходимо подчеркнуть, что развитие теории расписаний идет не только вглубь, но и вширь: появляются новые модели, которые описывают различные приложения,

возникающие в индустрии, телекоммуникациях, экономике и, особенно, в оптимизации компьютерных вычислений. Часто эти модели совмещают постановки различных классических задач дискретной оптимизации. Например, последняя глава диссертации посвящена цеховым задачам с маршрутизацией, которые являются обобщением классических цеховых задач открытого типа и метрической задачи коммивояжера.

Результаты, представленные в диссертации, охватывают весь спектр описанных в этом разделе исследований, однако основное внимание уделено построению приближенных алгоритмов. Это связано с тем, что результаты автора, вошедшие в диссертацию, получены в период с 1999 года по сегодняшней день, когда построение приближенных алгоритмов было основным направлением в развитии теории расписаний.

Глава 2

Задачи теории расписаний с воспроизводимым ресурсом

В этой главе рассматриваются задачи теории расписаний с воспроизводимым ресурсом. Множество работ $\mathcal{J} = \{J_1, \ldots, J_n\}$ должно быть выполнено на одной или нескольких машинах. Для работ задан общий ресурс объема Ω_0. Перед началом выполнения работа J_i потребляет α_i единиц ресурса и воспроизводит β_i единиц ресурса сразу после своего завершения. В дальнейшем величину α_i будем называть *расходом* ресурса на работу J_i, величину β_i — *доходом* ресурса от работы J_i и величину $\delta_i = \beta_i - \alpha_i$ — *прибылью* ресурса от работы J_i. Длительность работы J_i равна p_i. Прерывания в обслуживании работ запрещены. Требуется найти такое расписание σ, в котором выполнены все работы и в каждый момент времени t не нарушено следующее ограничение на ресурсы

$$\bar{\Omega}_t(\sigma) \doteq \Omega_0 + \sum_{\{J_i \in \mathcal{N} | C_i(\sigma) \le t\}} (\beta_i - \alpha_i) - \sum_{\{J_i \in \mathcal{N} | s_i(\sigma) \le t < C_i(\sigma)\}} \alpha_i \ge 0. \tag{2.1}$$

Величину $\bar{\Omega}_t(\sigma)$ будем называть остатком ресурса в момент t. Величину Ω_0 будем называть начальным объемом ресурса и величину

$$\Omega_t(\sigma) = \Omega_0 + \sum_{\{J_i \in \mathcal{N} | C_i(\sigma) \le t\}} (\beta_i - \alpha_i) - \sum_{\{J_i \in \mathcal{N} | s_i(\sigma) < t < C_i(\sigma)\}} \alpha_i$$

будем называть доступным объемом ресурса в момент t в расписании σ. Нетрудно проверить, что

$$\bar{\Omega}_t(\sigma) = \Omega_t(\sigma) - \sum_{\{J_i \in \mathcal{N} | t = s_i(\sigma);\, t < C_i(\sigma)\}} \alpha_i.$$

В дальнейшем будем опускать аргумент σ, если из контекста ясно, о каком расписании идет речь.

В англоязычной литературе задачи теории расписаний с воспроизводимым ресурсом называются задачами перепланировки (relocation problems). В первой статье [117], посвященной этому типу ресурсов, рассматривалась задача перепланировки жилого микрорайона в восточной части Бостона. По проекту требовалось снести

часть ветхого жилья и на освободившемся месте построить новые жилые дома и объекты торгового и социального значения. Перед сносом жилого дома всех его жителей необходимо расселить в некоторые временные помещения. Вместимость новых домов необязательно равна вместимости старых домов. Более того, если на месте новых домов планируется разбить парк, то новые квартиры для жилья могут в принципе отсутствовать. С другой стороны, вместимость новых жилых домов может значительно превосходить вместимость домов, снесенных на этом месте. Дополнительно предполагалось, что жильцы могут быть размещены в имеющийся фонд временного жилья произвольным образом, и вновь построенное жилье может быть использовано для временного размещения жильцов. При известном количестве мест во временном фонде жилья требовалось определить последовательность перепланировки объектов микрорайона так, чтобы у всех его жильцов всегда была крыша над головой.

Описанная проблема эквивалентна задаче нахождения допустимой последовательности работ при заданных Ω_0, α_i и β_i, $i = 1, \ldots, n$. В [118] Каплан и Амир показали, что эта задача сводится к классической задаче Джонсона с длительностями работы α_i на первой машине и β_i на второй машине. При этом минимальное количество начального ресурса, гарантирующее существование допустимой последовательности, равно суммарной длительности простоя второй машины в оптимальном расписании в задаче Джонсона.

В работах [118, 165] отмечено, что задачи с воспроизводимым ресурсом возникают в управлении базами данных и в планировании финансовых инвестиций. В частности, воспроизводимый ресурс рассматривается в задачах календарного планирования инвестиционных проектов с учетом возможности кредитования [6, 14, 19].

Как упоминалось в первой главе, понятие воспроизводимого ресурса обобщает понятия как возобновимого, так и невозобновимого ресурсов. Действительно, полагая в определении воспроизводимого ресурса $\alpha_i = \beta_i$ для всех работ, получаем возобновимый ресурс. А в случае, если для всех работ выполнено $\beta_i = 0$, имеем невозобновимый ресурс. В свою очередь, классическая задача об упаковке в контейнеры может быть сформулирована как задача с возобновимым ресурсом и единичными длительностями работ. Таким образом, задача с воспроизводимым ресурсом и единичными длительностями работ является естественным обобщением этой широко известной задачи комбинаторной оптимизации. Более подробно о связи этих задач изложено в [125] и параграфах 2.2.2, 2.2.4 этой главы.

Сформулированные во второй половине 80-х годов прошлого века как задачи перепланировки, задачи с воспроизводимым ресурсом не привлекли большого внимания. Возможно, непривычное название отпугнуло исследователей, работавших в области теории расписаний. В [118] Каплан и Амир рассмотрели задачи с воспроизводимым ресурсом на параллельных машинах. Они отметили, что задача $W1, Pm|p_i = 1|C_{max}$ с единичными длительностями работ является NP-трудной уже в случае трех машин. Сложность задачи на двух машинах $W1, P2|p_i = 1|C_{max}$ оставалась открытой больше 15 лет, пока Кононов и Лин [125] не показали, что задача остается NP-трудной в сильном смысле даже если $\alpha_i \leq \beta_i$ для всех работ $J_i \in \mathcal{J}$. В той же работе были получены нижние оценки на аппроксимируемость задачи $W1|p_i = 1|C_{max}$ и пред-

ставлены приближенные алгоритмы с гарантированной оценкой точности для задач $W1|p_i = 1|C_{max}$ и $W1, P|p_i = 1|C_{max}$. В [126] Кононов и Лин рассмотрели задачи с воспроизводимым ресурсом на одной машине, в которых требуется минимизировать взвешенную сумму моментов окончания работ, и показали, что задача является NP-трудной в сильном смысле даже если у всех работ единичные длительности или единичные веса. Там же для задач $W1, 1|\delta_i \leq 0|\sum C_i$ и $W1, 1|p_i = 1, \delta_i \geq 0|\sum w_i C_i$ предложен 2-приближенный алгоритм. Григорьев, Кононов и Свириденко [101] представили первый приближенный алгоритм для задачи $W1||C_{max}$ с воспроизводимым ресурсом и произвольными длительностями работ. Севастьянов, Лин и Хуанг [156] рассмотрели задачи с воспроизводимым ресурсом на одной машине, в которых для работ заданы различные моменты поступления и требуется минимизировать длину расписания. Они доказали NP-трудность различных версий задачи и построили для ее решения алгоритм динамического программирования, время работы которого ограничено полиномом от размера задачи в унарной кодировке при фиксированном числе машин и фиксированном числе различных моментов поступлений работ. Другие варианты задач теории расписаний, в которых рассматривался воспроизводимый ресурс, можно найти в [70, 137]. Данная глава диссертации посвящена результатам автора для задач теории расписаний с воспроизводимым ресурсом.

2.1 Одна машина. Минимизация взвешенной суммы моментов окончания работ

В этом разделе рассмотрим задачу на одной машине. Для каждой работы J_i заданы величины α_i и β_i, длительность p_i и вес w_i. Требуется найти расписание, допустимое относительно начального ресурса Ω_0, которое минимизирует величину $\sum_{i=1}^{n} w_i C_i$. Согласно классификации, введенной в параграфе 1.2.5, обозначим данную задачу через $W1, 1||\sum w_i C_i$.

В первом параграфе покажем, что рассматриваемая задача является NP-трудной даже при дополнительных ограничениях на величины α_i, β_i и w_i, $i = 1, \ldots, n$. Во втором параграфе представим 2-приближенный алгоритм для двух частных случаев задачи $W1, 1||\sum w_i C_i$. Заметим, что в оптимальном расписании машина работает без простоев, поэтому можно считать, что каждое "разумное" расписание задается перестановкой работ.

2.1.1 NP-трудность

В этом параграфе докажем NP-трудность в сильном смысле задачи $W1, 1||\sum w_i C_i$ и ее подзадач. Обозначим через $Z(\sigma)$ взвешенную сумму моментов окончания работ и через $\sigma(i)$ — работу в позиции i, $i = 1, \ldots, n$ в расписании σ. Напомним, что аналогичная задача $1||\sum w_i C_i$ без воспроизводимого ресурса является полиномиально разрешимой. Оптимальная перестановка в ней может быть найдена по правилу Сми-

та [161]: упорядочить работы по неубыванию отношения длительности работы к ее весу. Более того, нетрудно показать, что задача $1||\sum w_iC_i$ остается полиномиально разрешимой и в случае, когда дополнительный ресурс является возобновимым или невозобновимым.

Сначала докажем NP-трудность задачи $W1,1|p_j = 1,\delta_j \le 0|\sum w_iC_i$, в которой длительности всех работ равны единице и прибыль каждой работы не положительна. Сведем к задаче $W1,1|p_i = 1,\delta_i \le 0|\sum w_iC_i$ NP-полную в сильном смысле задачу УПОРЯДОЧЕННОЕ ТРЕХМЕРНОЕ СОЧЕТАНИЕ С ОГРАНИЧЕНИЯМИ ПО ВЕСУ (задача У3-СО в списке NP-полных задач в [4]).

УПОРЯДОЧЕННОЕ ТРЕХМЕРНОЕ СОЧЕТАНИЕ С ОГРАНИЧЕНИЯМИ ПО ВЕСУ

Условие: Заданы три m-элементных множества, $\mathcal{A}_1 = \{1,\dots,m\}, \mathcal{A}_2 = \{m{+}1,\dots,2m\}$, $\mathcal{A}_3 = \{2m+1,\dots,3m\}$, граница $B \in Z^+$ и такие размеры всех элементов $x_i \in Z^+$, $1 \le i \le 3m$, что $\sum_{i=1}^{3m} x_i = mB$ и

$$x_1 \ge x_2 \ge \dots \ge x_m \ge x_{m+1} \ge \dots \ge x_{2m} \ge x_{2m+1} \ge \dots \ge x_{3m}. \tag{2.2}$$

Вопрос: Можно ли множество $\mathcal{A}_1 \cup \mathcal{A}_2 \cup \mathcal{A}_3$ так разбить на m непересекающихся подмножеств $A_1, A_2, \dots, A_m$, что каждое множество $A_j, 1 \le j \le m$, содержит ровно по одному элементу из множеств $\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3$ и $\sum\limits_{i\in A_j} x_i = B$?

Заметим, что к задаче У3-СО легко сводится классическая NP-полная в сильном смысле задача ТРЕХМЕРНОЕ СОЧЕТАНИЕ С ОГРАНИЧЕНИЯМИ ПО ВЕСУ (задача МР-16 в списке NP-полных задач в [4]).

ТРЕХМЕРНОЕ СОЧЕТАНИЕ С ОГРАНИЧЕНИЯМИ ПО ВЕСУ

Условие: Заданы три m-элементных множества, $\mathcal{A}_1 = \{1,\dots,m\}, \mathcal{A}_2 = \{m{+}1,\dots,2m\}$, $\mathcal{A}_3 = \{2m+1,\dots,3m\}$, граница $B \in Z^+$ и такие размеры всех элементов $x_i \in Z^+$, $1 \le i \le 3m$, что $\sum\limits_{i=1}^{3m} x_i = mB$ и

$$\frac{B}{4} < x_i < \frac{B}{2}. \tag{2.3}$$

Вопрос: Можно ли множество $\mathcal{A}_1 \cup \mathcal{A}_2 \cup \mathcal{A}_3$ так разбить на m непересекающихся подмножеств $A_1, A_2, \dots, A_m$, что каждое множество $A_j, 1 \le j \le m$, содержит ровно по одному элементу из множеств $\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3$ и $\sum\limits_{i\in A_j} x_i = B$?

Задача МР-16 отличается от задачи У3-СО наличием условия (2.3) и отсутствием условия (2.2) на размеры элементов x_i, $1 \le i \le 3m$. Добавляя к весу каждого слагаемого из множества $\mathcal{A}_k$, $k = 1,2,3$ дополнительный вес kB и увеличивая границу до $7B$, получим сведение задачи МР-16 к задаче У3-СО.

Теорема 1 *Задача $W1,1|p_j = 1,\delta_j \le 0|\sum w_iC_i$ является NP-трудной в сильном смысле.*

Доказательство. Пусть множества элементов $\mathcal{A}_1 = \{1, \ldots, m\}, \mathcal{A}_2 = \{m+1, \ldots, 2m\}$, $\mathcal{A}_3 = \{2m+1, \ldots, 3m, \}$, их размеры x_i, $1 \le i \le 3m$, и B определяют произвольную индивидуальную задачу УЗ-СО. Положим $\eta = 6m^2B$. Соответствующий пример I задачи $W1, 1|p_j = 1, \delta_j \le 0| \sum w_iC_i$ состоит из $3m$ базовых работ и $3m-3$ связующих работ. Объем начального ресурса $\Omega_0 = (3mB+B)m$, а параметры работ заданы следующим образом:

- базовые работы J_i, $1 \le i \le 3m$: $\alpha_i = mB + x_i$, $\beta_i = 0, w_i = \eta + x_i$;
- связующие работы $J_{3(m+l-1)+k}$, $1 \le l \le m-1$, $1 \le k \le 3$: $\alpha_{3(m+l-1)+k} = \beta_{3(m+l-1)+k} = (3mB+B)(m-l), w_{3(m+l-1)+k} = 0$.

Обозначим через Z^* величину $3m\eta(3m-1) + 3mB(m-1) + \sum\limits_{i\in\mathcal{A}_1} x_i + 2\sum\limits_{i\in\mathcal{A}_2} x_i + 3\sum\limits_{i\in\mathcal{A}_3} x_i$. Заметим, что $Z^* < 3m\eta(3m-1) + 3mB(m-1) + 3(\sum\limits_{i\in\mathcal{A}_1} x_i + \sum\limits_{i\in\mathcal{A}_2} x_i + \sum\limits_{i\in\mathcal{A}_3} x_i) = 3m\eta(3m-1)+3m^2B$. Покажем, что в индивидуальной задаче I существует допустимое расписание с $\sum w_iC_i \le Z^*$ тогда и только тогда, когда для исходной индивидуальной задачи УЗ-СО существует искомое разбиение.

Далее будем рассматривать только расписания, в которых связующие работы расположены в порядке возрастания их индексов. Назовем такие расписания *правильными.* Действительно, пусть в некотором расписании связующая работа J_j предшествует другой связующей работе J_i и $i < j$. Поменяем эти работы местами. Поскольку прибыль обеих работ равна нулю и вес обеих работ равен нулю, то перестановка работ J_i и J_j не изменит объем свободного ресурса в каждый момент времени и не изменит значение целевой функции. Таким образом, осталось проверить, что объема ресурса достаточно для выполнения работ J_i и J_j в новых позициях. Так как все работы в I дают неположительную прибыль, то доступный объем ресурса в моменты начала выполнения работ не увеличивается в течение времени. Последнее вместе с неравенством $\alpha_i \ge \alpha_j$ влечет допустимость расписания, полученного после перестановки работ J_i и J_j.

Предположим сначала, что подмножества $A_1, A_2, \ldots, A_m$ содержат ровно по одному элементу из множеств $\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3$ и $\sum\limits_{i\in A_j} x_i = B$. Пусть π_l обозначает перестановку индексов элементов из множества A_l в порядке их возрастания. Определим последовательность индексов

$$\begin{aligned}\sigma_0 \;=\; & (\pi_1, 3m+1, 3m+2, 3m+3, \pi_2, \ldots, \pi_l, 3(m+l-1)+1,\\ & 3(m+l-1)+2, 3(m+l-1)+3, \pi_{l+1}, \ldots, \pi_m).\end{aligned}$$

Проверим, что расписание работ, в котором порядок индексов совпадает с перестановкой σ_0, является допустимым. Пусть $\Omega' = \Omega_{6l-3}$ — объем ресурса после выполнения базовых работ с индексами из π_l. Так как прибыль связующих работ равна нулю, то

$$\Omega' = \Omega_0 - \sum_{j=1}^{l}\sum_{i\in\pi_j}(mB + x_i) = (3mB+B)m - l(3mB+B) = B(3m+1)(m-l).$$

Так как $\alpha_{3(m+l-1)+k} = \beta_{3(m+l-1)+k} = (3mB+B)(m-l)$, $k = 1,2,3$, и $B(3m+1)(m-l) \geq B(3m+1)$ для $l \leq m-1$, то объема ресурса Ω' достаточно для последовательного выполнения связующих работ $J_{3(m+l-1)+k}$, $k = 1,2,3$, и базовых работ с индексами из множества π_{l+1}.

Вычислим взвешенную сумму моментов окончания работ $Z(\sigma_0)$. Учитывая, что работы с индексами из π_l выполняются в интервале $(6l-6, 6l-3)$ и веса связующих работ равны нулю, имеем

$$\begin{aligned} Z(\sigma_0) &= \sum_{l=1}^{m}\sum_{k=1}^{3}(6(l-1)+k)(\eta + x_{[6(l-1)+k]}) = \sum_{l=1}^{m}(18l-12)\eta \\ &+ \sum_{l=1}^{m}\sum_{k=1}^{3}6(l-1)x_{[6(l-1)+k]} + \sum_{l=1}^{m}\sum_{k=1}^{3}kx_{[6(l-1)+k]}. \end{aligned}$$

Здесь индексы в квадратных скобках обозначают позицию работы в перестановке σ_0. Замечая, что $\sum_{k=1}^{3} x_{[6(l-1)+k]} = B$ для всех $l = 1,\ldots,m$, получим

$$\begin{aligned} Z(\sigma_0) &= 9m(m+1)\eta - 12m\eta + 6\sum_{l=1}^{m}(l-1)B + \sum_{l=1}^{m}\sum_{k=1}^{3}kx_{[6(l-1)+k]} \\ &= 3m\eta(3m-1) + 3mB(m-1) + \sum_{l=1}^{m}\sum_{k=1}^{3}kx_{[6(l-1)+k]}. \end{aligned}$$

При построении σ_0 каждый элемент из $\mathcal{A}_k$, $k = 1,2,3$, соответствует одной из позиций $6(l-1)+k$, $l = 1,2,\ldots,m$ в σ_0. Следовательно, $\sum_{l=1}^{m} kx_{[6(l-1)+k]} = k\sum_{i\in\mathcal{A}_k} x_i$ для $k = 1,2,3$. Окончательно получим $Z(\sigma_0) = Z^*$.

Предположим, что существует правильное допустимое расписание σ индивидуальной задачи I такое, что $Z(\sigma) \leq Z^*$. Сначала покажем, что связующая работа $J_{3(m+l-1)+k}$ должна выполняться в σ до момента $6l+k-4, 1 \leq l \leq m-1, 1 \leq k \leq 3$. Пусть это не так, и работа $J_{3(m+l-1)+k}$ начинается в момент $\tau > 6l+k-4$. Так как σ — правильное расписание, то ровно $3(l-1)+k-1$ связующих работ предшествуют в σ работе $J_{3(m+l-1)+k}$. Следовательно, по крайней мере $3l+1$ базовая работа закончится до момента τ. Имеем,

$$\begin{aligned} \Omega_\tau &\leq (3mB+B)m - (3l+1)mB \\ &= (3mB+B)(m-l) - B(m-l) < (3mB+B)(m-l). \end{aligned}$$

Последнее строгое неравенство следует из неравенства $l \leq m-1$. Таким образом получим, $\alpha_{3(m+l-1)+k} = (3mB+B)(m-l) > \Omega_\tau$, что противоречит допустимости расписания σ. Следовательно, для всех l и k, $1 \leq l \leq m-1, 1 \leq k \leq 3$, работа $J_{3(m+l-1)+k}$ должна завершиться не позднее момента $6l+k-3$.

Отсюда следует, что для любого целого l, $1 \leq l \leq m-1$, в интервале $(0, 6l)$ выполняется ровно $3l$ базовых и $3l$ связующих работ. Докажем, что базовые работы должны выполняться в интервалах $(6l-6, 6l-3)$, $l = 1,\ldots,m$. Пусть это не так,

тогда

$$\begin{aligned} Z(\sigma) &> \sum_{l=1}^{m}\sum_{k=1}^{3}(6(l-1)+k)\eta+\eta \\ &= 3m\eta(3m-1)+\eta = 3m\eta(3m-1)+6m^2B > Z^*, \end{aligned}$$

и получаем противоречие с предположением $Z(\sigma) \le Z^*$.

Таким образом, в расписании σ базовые работы выполняются в интервалах $(6l-6, 6l-3)$, $l = 1, \ldots, m$, и, соответственно, связующие работы — в интервалах $[6l-3, 6l), l = 1, \ldots, m-1$. Пусть $B_l = \sum_{k=1}^{3} x_{[6(l-1)+k]}$. Докажем, что

$$\sum_{l=1}^{s} B_l = \sum_{l=1}^{s}\sum_{k=1}^{3} x_{[6(l-1)+k]} \le sB \tag{2.4}$$

для всех $s = 1, \ldots, m$. Это утверждение, очевидно, верно для $s = m$. Пусть утверждение не выполнено для некоторого индекса $s' < m$. Тогда

$$\begin{aligned} \Omega_{6s'-3}(\sigma) &= \Omega_0 - \sum_{l=1}^{s'}\sum_{k=1}^{3}\alpha_{[6(l-1)+k]} = (3mB+B)m \\ &- \left[3s'mB + \sum_{l=1}^{s'}\sum_{k=1}^{3} x_{[6(l-1)+k]}\right] < (3mB+B)(m-s'). \end{aligned}$$

Так как $\alpha_{[6s-2]} = \alpha_{3(m+s'-1)+1} = (3mB+B)(m-s')$, то получим противоречие с допустимостью расписания σ.

С учетом вышесказанного оценим значение целевой функции в перестановке σ.

$$Z(\sigma) = \sum_{l=1}^{m}\sum_{k=1}^{3}(6(l-1)+k)(\eta + x_{[6(l-1)+k]})$$

$$= 9m(m+1)\eta - 12m\eta + \sum_{l=1}^{m}\sum_{k=1}^{3} 6(l-1)x_{[6(l-1)+k]} + \sum_{l=1}^{m}\sum_{k=1}^{3} kx_{[6(l-1)+k]}.$$

Преобразуем второе слагаемое в последнем выражении.

$$\begin{aligned} \sum_{l=1}^{m}\sum_{k=1}^{3} 6(l-1)x_{[6(l-1)+k]} &= 6\sum_{l=1}^{m}(l-1)B_l \\ &= 6\left(\sum_{l=1}^{m} mB_l - \sum_{l=1}^{m}(m-l+1)B_l\right) \\ &= 6\left(m^2B - (mB_1 + (m-1)B_2 + \cdots + B_m)\right) \\ &= 6\left(m^2B - \left(\sum_{l=1}^{m} B_l + \sum_{l=1}^{m-1} B_l + \cdots + \sum_{l=1}^{1} B_l\right)\right) \\ &\ge 6m^2B - 6\left(mB + (m-1)B + \cdots + B\right) \\ &= 3mB(m-1). \end{aligned}$$

Предпоследнее неравенство следует из (2.4). Заменяя второе слагаемое в выражении для $Z(\sigma)$ на $3mB(m-1)$, получим

$$Z(\sigma) \geq 3m\eta(3m-1) + 3mB(m-1) + \sum_{l=1}^{m}\sum_{k=1}^{3} kx_{[6(l-1)+k]}. \qquad (2.5)$$

Последнее слагаемое в (2.5) можно оценить снизу через $\sum_{i\in\mathcal{A}_1} x_i + 2\sum_{i\in\mathcal{A}_2} x_i + 3\sum_{i\in\mathcal{A}_3} x_i$. Более того, равенство

$$\sum_{l=1}^{m}\sum_{k=1}^{3} kx_{[6(l-1)+k]} = \sum_{i\in\mathcal{A}_1} x_i + 2\sum_{i\in\mathcal{A}_2} x_i + 3\sum_{i\in\mathcal{A}_3} x_i$$

достигается только в случае, если для всех s индекс работы $J_{[6(s-1)+1]}$ соответствует некоторому элементу из $\mathcal{A}_1$, индекс работы $J_{[6(s-1)+2]}$ — некоторому элементу из $\mathcal{A}_2$, и индекс работы $J_{[6(s-1)+3]}$ — некоторому элементу из $\mathcal{A}_3$.

Замечая, что из неравенств (2.4) и (2.5) следует $Z = Z^*$ тогда и только тогда, когда $\sum_{l=1}^{s} B_l = sB$ для всех $s, 1 \leq s \leq m$, получим $B_l = B, 1 \leq l \leq m$, и для исходного примера задачи УЗ-СО существует искомое разбиение. □

Аналогично доказывается NP-трудность задачи $W1,1|p_j = 1, \delta_j \geq 0|\sum w_iC_i$, в которой длительности всех работ равны единице и прибыль каждой работы неотрицательна.

Теорема 2 *Задача $W1,1|p_j = 1, \delta_j \geq 0|\sum w_iC_i$ является NP-трудной в сильном смысле.*

Доказательство. Как и в доказательстве теоремы 1, по произвольной индивидуальной задаче УЗ-СО построим пример I задачи $W1,1|p_j = 1, \delta_j \geq 0|\sum w_iC_i$. Пример I состоит из $3m$ базовых работ и $3m-3$ связующих работ. Объем начального ресурса $\Omega_0 = 0$, а параметры работ заданы следующим образом:

- базовые работы $J_i, 1 \leq i \leq 3m$: $\alpha_i = 0, \beta_i = mB + x_i, w_i = B - x_i$;
- связующие работы $J_{3(m+i-1)+k}, 1 \leq i \leq m-1, 1 \leq k \leq 3$:
  $\alpha_{3(m+i-1)+k} = \beta_{3(m+i-1)+k} = (3mB + B)i, w_{3(m+i-1)+k} = \eta$.

Пусть $Z^* = (9m^2 - 12m)\eta + 6m^2B - 3\sum_{i\in\mathcal{A}_1} x_i - 2\sum_{i\in\mathcal{A}_2} x_i - \sum_{i\in\mathcal{A}_3} x_i$. Покажем, что в индивидуальной задаче I существует допустимое расписание с $\sum w_iC_i \leq Z^*$ тогда и только тогда, когда для исходной индивидуальной задачи УЗ-СО существует требуемое разбиение.

Как и в доказательстве предыдущей теоремы, будем рассматривать только правильные расписания, в которых связующие работы расположены в порядке возрастания их индексов. Используя перестановочный прием, легко показать, что любое

допустимое расписании можно перестроить в правильное допустимое расписание без изменения значения целевой функции.

Предположим сначала, что подмножества $A_1, A_2, \ldots, A_m$, содержат ровно по одному элементу из множеств $\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3$ и $\sum_{i \in A_j} x_i = B$. Пусть π_l обозначает перестановку индексов элементов из множества A_l в порядке их возрастания. Определим последовательность индексов

$$\begin{aligned}\sigma_0 \;=\; & (\pi_1, 3m+1, 3m+2, 3m+3, \pi_2, \ldots, \pi_l, 3(m+l-1)+1,\\ & 3(m+l-1)+2, 3(m+l-1)+3, \pi_{l+1}, \ldots, \pi_m).\end{aligned}$$

Проверим, что расписание работ, в котором порядок индексов совпадает с перестановкой σ_0, является допустимым. Пусть $\Omega' = \Omega_{6l-3}$ — объем ресурса после выполнения базовых работ с индексами из π_l. Так как прибыль связующих работ равна нулю, то

$$\Omega' = \sum_{j=1}^{l} \sum_{i \in \pi_j} (mB + x_i) = (3mB + B)l.$$

Поскольку $\alpha_{3(m+l-1)+k} = \beta_{3(m+l-1)+k} = (3mB + B)l$, $k = 1, 2, 3$, а расход ресурса всех базовых работ равен нулю, объема ресурса Ω' достаточно для последовательного выполнения связующих работ $J_{3(m+l-1)+k}$, $k = 1, 2, 3$, и базовых работ с индексами из множества π_{l+1}.

Вычислим взвешенную сумму моментов окончания работ $Z(\sigma_0)$.

$$\begin{aligned}Z(\sigma_0) \;=\; & \sum_{l=1}^{m} \sum_{k=1}^{3} (6(l-1)+k)(B - x_{[6(l-1)+k]}) + \sum_{l=1}^{m-1} \sum_{k=1}^{3} (6l-3+k)\eta\\ =\; & \sum_{l=1}^{m} (18l-12)B - \sum_{l=1}^{m} \sum_{k=1}^{3} 6(l-1)x_{[6(l-1)+k]} - \sum_{l=1}^{m} \sum_{k=1}^{3} k x_{[6(l-1)+k]}\\ +\; & \sum_{l=1}^{m} (18l-3)\eta = (9m^2-3m)B - \sum_{l=1}^{m} \sum_{k=1}^{3} 6(l-1)x_{[6(l-1)+k]}\\ -\; & \sum_{l=1}^{m} \sum_{k=1}^{3} k x_{[6(l-1)+k]} + (9m^2-12m)\eta.\end{aligned}$$

Замечая, что $\sum_{k=1}^{3} x_{[6(l-1)+k]} = B$ для всех $l = 1, \ldots, m$, получим

$$\begin{aligned}Z(\sigma_0) \;=\; & (9m^2-12m)\eta + (9m^2-3m)B - \sum_{l=1}^{m} 6(l-1)B - \sum_{l=1}^{m} \sum_{k=1}^{3} k x_{[6(l-1)+k]}\\ =\; & (9m^2-12m)\eta + (9m^2-3m)B + (3m-3m^2)B - \sum_{l=1}^{m} \sum_{k=1}^{3} k x_{[6(l-1)+k]}\\ =\; & (9m^2-12m)\eta + 6m^2 B - \sum_{l=1}^{m} \sum_{k=1}^{3} k x_{[6(l-1)+k]}.\end{aligned}$$

При построении σ_0 каждый элемент из $\mathcal{A}_k$, $k = 1, 2, 3$, соответствует одной из позиций $6(l-1)+k$, $l = 1, 2, \ldots, m$ в σ_0. Следовательно, $\sum_{l=1}^{m} k x_{[6(l-1)+k]} = k \sum_{i \in \mathcal{A}_k} x_i$ для $k = 1, 2, 3$. Окончательно получим $Z(\sigma_0) = Z^*$.

Предположим, что существует правильное допустимое расписание σ индивидуальной задачи I такое, что $Z(\sigma) \leq Z^*$. Сначала покажем, что связующая работа $J_{3(m+l-1)+k}$ может начаться в σ не ранее момента $6l + k - 4, 1 \leq l \leq m-1, 1 \leq k \leq 3$. Пусть это не так, и работа $J_{3(m+l-1)+k}$ начинается в момент $\tau < 6l+k-4$. Так как σ — правильное расписание, то ровно $3(l-1)+k-1$ связующих работ предшествуют в σ работе $J_{3(m+l-1)+k}$. Следовательно, к моменту τ завершилось не более $3l-1$ базовых работ. Учитывая, что $\sum_{i=1}^{3m} x_i = mB$, имеем $\Omega_\tau < mB(3l-1) + mB = 3lmB$. Таким образом, получим $\alpha_{3(m+l-1)+k} = (3mB + B)l > \Omega_\tau$, что противоречит допустимости расписания σ. Следовательно, для всех l и k, $1 \leq l \leq m-1, 1 \leq k \leq 3$, работа $J_{3(m+l-1)+k}$ должна начаться не ранее момента $6l + k - 4$.

Предположим, что работа $J_{3(m+l-1)+k}$ начинается позже момента $6l + k - 4$. Тогда имеем

$$Z(\sigma) \geq \sum_{l=1}^{m-1} \sum_{k=1}^{3} (6l - 3 + k)\eta + \eta = (9m^2 - 12m)\eta + 6m^2 B > Z^*$$

и получаем противоречие с предположением $Z(\sigma) \leq Z^*$.

Таким образом, в расписании σ связующие работы выполняются в интервалах $[6l-3, 6l), l = 1, \ldots, m-1$, и, соответственно, базовые работы — в интервалах $(6l-6, 6l-3)$, $l = 1, \ldots, m$. Пусть $B_l = \sum_{k=1}^{3} x_{[6(l-1)+k]}$. Докажем, что

$$\sum_{l=1}^{s} B_l = \sum_{l=1}^{s} \sum_{k=1}^{3} x_{[6(l-1)+k]} \geq sB \tag{2.6}$$

для всех $s = 1, \ldots, m$. Это утверждение, очевидно, верно для $s = m$. Пусть утверждение не выполнено для некоторого индекса $s' < m$. Тогда

$$\Omega_{6s'-3}(\sigma) = \sum_{l=1}^{s'} \sum_{k=1}^{3} \beta_{[6(l-1)+k]} = 3s'mB + \sum_{l=1}^{s'} \sum_{k=1}^{3} x_{[6(l-1)+k]} < (3mB + B)s'.$$

Так как $\alpha_{[6s-2]} = \alpha_{3(m+s'-1)+1} = (3mB + B)s'$, получим противоречие с допустимостью расписания σ.

С учетом вышесказанного оценим значение целевой функции в перестановке σ.

$$\begin{aligned} Z(\sigma) &= \sum_{l=1}^{m}\sum_{k=1}^{3}(6(l-1)+k)(B - x_{[6(l-1)+k]}) + \sum_{l=1}^{m-1}\sum_{k=1}^{3}(6l-3+k)\eta \\ &= (9m^2-3m)B - \sum_{l=1}^{m}\sum_{k=1}^{3}6(l-1)x_{[6(l-1)+k]} \\ &\quad - \sum_{l=1}^{m}\sum_{k=1}^{3}kx_{[6(l-1)+k]} + (9m^2-12m)\eta. \end{aligned}$$

Преобразуем второе слагаемое в последнем выражении аналогично тому, как это было сделано в доказательстве теоремы 1.

$$\sum_{l=1}^{m}\sum_{k=1}^{3}6(l-1)x_{[6(l-1)+k]} = 6\left(m^2B - \left(\sum_{l=1}^{m}B_l + \sum_{l=1}^{m-1}B_l + \cdots + \sum_{l=1}^{1}B_l\right)\right)$$

$$\leq 6m^2B - 6\left(mB + (m-1)B + \cdots + B\right) = B(3m^2-3m).$$

Отсюда следует, что

$$Z(\sigma) \geq (9m^2-12m)\eta + 6m^2B - \sum_{l=1}^{m}\sum_{k=1}^{3}kx_{[6(l-1)+k]}. \tag{2.7}$$

Заметим, что $\sum_{l=1}^{m}\sum_{k=1}^{3}kx_{[6(l-1)+k]} \geq 3\sum_{i\in\mathcal{A}_1}x_i - 2\sum_{i\in\mathcal{A}_2}x_i - \sum_{i\in\mathcal{A}_3}x_i$. Более того, равенство в последнем выражении достигается только в том случае, когда для всех s индекс работы $J_{[6(s-1)+1]}$ соответствует некоторому элементу из $\mathcal{A}_3$, индекс работы $J_{[6(s-1)+2]}$ соответствует некоторому элементу из $\mathcal{A}_2$, и индекс работы $J_{[6(s-1)+3]}$ соответствует некоторому элементу из $\mathcal{A}_1$. Таким образом $Z(\sigma) \geq Z^*$, и из неравенств (2.6) и (2.7) следует, что $Z(\sigma) = Z^*$ тогда и только тогда, когда $\sum_{l=1}^{s}B_l = sB$ для всех $s, 1 \leq s \leq m$. Следовательно, $B_l = B, 1 \leq l \leq m$, и для исходного примера задачи У3-СО существует искомое разбиение. □

Используя результаты теорем 1 и 2, докажем NP-трудность задач $W1,1|\delta_j \leq 0|\sum C_j$ и $W1,1|\delta_j \geq 0|\sum C_j$ с произвольными длительностями работ и единичными весами. Для этого покажем тесную взаимосвязь между задачами с единичными длительностями и задачами с единичными весами.

Рассмотрим две задачи минимизации $\mathcal{P}$ и $\mathcal{Q}$ с целевыми функциями $\Phi_{\mathcal{P}}$ и $\Phi_{\mathcal{Q}}$ соответственно. Будем говорить, что задача $\mathcal{P}$ вполне сводится к задаче $\mathcal{Q}$, если существуют функции f и g, вычислимые в линейное от размера входа время, такие что f преобразует пример I задачи $\mathcal{P}$ в пример $\bar{I}$ задачи $\mathcal{Q}$, g преобразует решение $\bar{\sigma}$ примера $\bar{I}$ в решение σ примера I и $\Phi_{\mathcal{P}}(I,\sigma) = \Phi_{\mathcal{Q}}(\bar{I},\bar{\sigma})$. Если задача $\mathcal{P}$ вполне сводится к задаче $\mathcal{Q}$ и задача $\mathcal{Q}$ вполне сводится к задаче $\mathcal{P}$, то задачи называются *вполне эквивалентными.*

Теорема 3 *Задачи* $W1,1||\sum C_i$ *и* $W1,1|p_j=1|\sum w_iC_i$ *вполне эквивалентны.*

Доказательство. Покажем, что задача $W1,1|p_j=1|\sum w_iC_i$ вполне сводится к задаче $W1,1||\sum C_i$. Рассмотрим пример I задачи $W1,1|p_j=1|\sum w_iC_i$ с n работами и заданными параметрами α_i, β_i, w_i для каждой работы J_i. Пусть начальный объем ресурса равен Ω_0. Построим пример $\tilde{I}$ задачи $W1,1||\sum C_i$ с n работами и параметрами $\tilde{\alpha}_i=\beta_i$, $\tilde{\beta}_i=\alpha_i$, $\tilde{p}_i=w_i$ для работы J_i. Объем начального ресурса в примере $\tilde{I}$ положим $\tilde{\Omega}_0=\Omega_0+\sum\limits_{i=1}^{n}(\beta_i-\alpha_i)$.

Пусть $\sigma=(\sigma(1),\sigma(2),\ldots,\sigma(n))$ — допустимая перестановка в примере I. Покажем, что перестановка $\tilde{\sigma}=(\sigma(n),\sigma(n-1),\ldots,\sigma(1))$ является допустимой перестановкой в примере $\tilde{I}$. Действительно, пусть $\tilde{\Omega}_k$ обозначает объем ресурса после выполнения работ $J_{\sigma(n)},\ldots,J_{\sigma(k+1)}$ в $\tilde{\sigma}$. Имеем, $\tilde{\Omega}_k-\tilde{\alpha}_{\sigma(k)}=\tilde{\Omega}_0+\sum\limits_{i=k+1}^{n}(\tilde{\beta}_{\sigma(i)}-\tilde{\alpha}_{\sigma(i)})-\tilde{\alpha}_{\sigma(k)}=\Omega_0+\sum\limits_{i=1}^{k}(\beta_{\sigma(i)}-\alpha_{\sigma(i)})-\beta_{\sigma(k)}=\Omega_0+\sum\limits_{i=1}^{k-1}(\beta_{\sigma(i)}-\alpha_{\sigma(i)})-\alpha_{\sigma(k)}\geq 0$. Последнее неравенство следует из допустимости расписания σ. В итоге получаем $\tilde{\Omega}_k\geq\tilde{\alpha}_{\sigma(k)}$, и, следовательно, расписание $\tilde{\sigma}$ допустимо.

Пусть $C_i(\sigma)$ и $C_i(\tilde{\sigma})$ обозначают время завершения работы J_i в расписаниях σ и $\tilde{\sigma}$ соответственно. Имеем $\sum\limits_{i=1}^{n}C_i(\tilde{\sigma})=\sum\limits_{k=1}^{n}k\tilde{p}_{\tilde{\sigma}(n-k+1)}=\sum\limits_{k=1}^{n}kw_{\sigma(k)}=\sum\limits_{i=1}^{n}w_iC_i(\sigma)$.

Аналогично проверяется, что задача $W1,1||\sum C_i$ вполне сводится к задаче $W1,1|p_j=1|\sum w_iC_i$.

Из доказательства теоремы 3 следует, что задача $W1,1|\delta_i\geq 0|\sum C_i$ вполне эквивалентна задаче $W1,1|p_j=1,\delta_i\leq 0|\sum w_iC_i$ и задача $W1,1|\delta_i\leq 0|\sum C_i$ вполне эквивалентна задаче $W1,1|p_j=1,\delta_i\geq 0|\sum w_iC_i$.

Следствие 1 *Задача* $W1,1|\delta_i\geq 0|\sum C_i$ *является NP-трудной в сильном смысле.*

Следствие 2 *Задача* $W1,1|\delta_i\leq 0|\sum C_i$ *является NP-трудной в сильном смысле.*

2.1.2 Приближенный алгоритм

В этом разделе представим 2-приближенный жадный алгоритм для задачи $W1,1|p_j=1,\delta_i\geq 0|\sum w_iC_i$. Зададим два приоритета на множестве работ. Интуитивно ясно, что целесообразно выполнить первыми либо работы, производящие максимальный дополнительный ресурс δ_i, либо работы, имеющие наибольший вес w_i. Принимая в расчет оба аргумента, построим последовательность π_1, в которой работы упорядочены по невозрастанию величин w_i, и последовательность π_2, в которой работы упорядочены по невозрастанию величин δ_i. Алгоритм 2.1 строит частичное расписание, поочередно выбирая из каждого множества работ одну и устанавливая ее в расписание. В ходе работы алгоритма работа J_i называется *доступной* в момент времени t, если она не установлена в текущее частичное расписание и $\alpha_i\leq\Omega_t$. Сначала

алгоритм помещает первую доступную работу из π_1 на первую позицию в искомом расписании σ и удаляет ее из обеих последовательностей. Затем алгоритм помещает первую доступную работу из π_2 на вторую позицию в σ и удаляет ее из обеих последовательностей. Процесс установки работ в расписании с попеременным использованием последовательностей π_1 и π_2 продолжается до тех пор, пока не будут расписаны все работы или не останется доступных работ. В последнем случае алгоритм устанавливает, что объема начального ресурса недостаточно для выполнения всех работ данного множества, и задача не имеет решения.

Алгоритм 2.1

1: **for** $i = 0, 1, \ldots, \lfloor \frac{n}{2} \rfloor$ **do**
2: **if** нет доступных работ **then**
3: **stop** и выдать "Допустимого расписания не существует".
4: Найти доступную работу J_k с наименьшим индексом в π_1.
5: Положить $\sigma(2i+1) = k$ и удалить J_k из π_1 и π_2.
6: Положить $\Omega_{2i+1} = \Omega_{2i} + \delta_k$.
7: **if** все работы уже расписаны **then**
8: **stop** и выдать σ.
9: **if** нет доступных работ **then**
10: **stop** и выдать "Допустимого расписания не существует".
11: Найти доступную работу J_k с наименьшим индексом в π_2.
12: Положить $\sigma(2i+2) = k$ и удалить J_k из π_1 и π_2.
13: Положить $\Omega_{2i+2} = \Omega_{2i+1} + \delta_k$.
14: Выдать σ.

Оценим время работы алгоритма 2.1. Алгоритм выполняет не более $\lfloor \frac{n}{2} \rfloor + 1$ итераций. Выбор первой доступной работы в перестановках π_1 и π_2 на каждой итерации требует $O(n)$ элементарных операций. Остальные шаги алгоритма можно выполнить за $O(1)$ элементарных операций. Следовательно, алгоритм находит решение за время $O(n^2)$.

Лемма 1 *Если для некоторого примера I алгоритм 2.1 не находит допустимого решения, то объема начального ресурса недостаточно для выполнения всех работ данного множества, и в примере I нет допустимых решений.*

Доказательство. Предположим, что алгоритм 2.1 не нашел допустимого расписания. Обозначим через $\mathcal{J}_s$ множество работ, установленных в расписание алгоритмом 2.1, а через $\mathcal{J}_u$ — множество остальных работ. Тогда для любой работы $J_i \in \mathcal{J}_u$ выполнено неравенство

$$a_i > \Omega_0 + \sum_{J_l \in \mathcal{J}_s} \delta_l. \tag{2.8}$$

Предположим, что пример I имеет допустимое решение σ'. Выберем из множества $\mathcal{J}_u$ первую работу в перестановке σ'. Пусть это будет работа J_i и пусть Ω' — объем

ресурса в момент начала ее выполнения в расписании σ'. Из допустимости расписания σ' и неотрицательности прибыли каждой работы имеем $a_i \le \Omega' \le \Omega_0 + \sum_{J_l \in \mathcal{J}_s} \delta_l$. Получаем противоречие с (2.8) и условием, что в момент остановки алгоритма нет доступных работ.

Перейдем к анализу точности алгоритма 2.1. Обозначим расписание, полученное алгоритмом 2.1, через σ. Пусть σ^* – произвольное оптимальное расписание.

Лемма 2 *$\Omega_t(\sigma^*) \le \Omega_{2t}(\sigma)$ для всех $t = 0, 1, \dots \lfloor \frac{n}{2} \rfloor$.*

Доказательство. Так как длительность каждой работы равна единице, и машина в расписаниях σ и σ^* работает без простоев, то $\Omega_t(\sigma^*) = \Omega_0 + \sum_{i=1}^{t} \delta_{\sigma^*(i)}$ и $\Omega_{2t}(\sigma) = \Omega_0 + \sum_{i=1}^{2t} \delta_{\sigma(i)}$. Индукцией по t покажем, что для любого $t = 1, \dots, \lfloor \frac{n}{2} \rfloor$ существует инъективное отображение $f_t : \{\delta_{\sigma^*(1)}, \dots, \delta_{\sigma^*(t)}\} \to \{\delta_{\sigma(1)}, \dots, \delta_{\sigma(2t)}\}$ такое, что

$$\delta_{\sigma^*(i)} \le f_t(\delta_{\sigma^*(i)}) \text{ для всех } 1 \le i \le t. \tag{2.9}$$

Для $t = 1$ определим $f_1(\delta_{\sigma^*(1)}) = \max\{\delta_{\sigma(1)}, \delta_{\sigma(2)}\} \ge \delta_{\sigma^*(1)}$. Пусть для некоторого $t < \lfloor \frac{n}{2} \rfloor$ отображение f_t найдено. Тогда имеем, $\Omega_t(\sigma^*) \le \Omega_{2t}(\sigma) \le \Omega_{2t+1}(\sigma)$. Построим отображение f_{t+1}. Определим $f_{t+1}(\delta_{\sigma^*(i)}) = f_t(\delta_{\sigma^*(i)})$ для всех $1 \le i \le t$. Если $\delta_{\sigma^*(t+1)} \le \delta_{\sigma(2t+2)}$, то доопределим $f_{t+1}(\delta_{\sigma^*(t+1)}) = \delta_{\sigma(2t+2)}$, и доказательство закончено. Пусть это не так, то есть $\delta_{\sigma^*(t+1)} > \delta_{\sigma(2t+2)}$, тогда согласно Алгоритму 2.1 в расписании σ работа $J_{\sigma^*(t+1)}$ должна быть выполнена раньше работы $J_{\sigma(2t+2)}$. Другими словами, $J_{\sigma^*(t+1)} \equiv J_{\sigma(j)}$ для некоторого j, $1 \le j \le 2t+1$. Если $\delta_{\sigma(j)}$ не принадлежит области значений f_{t+1}, то положим $f_{t+1}(\delta_{\sigma^*(t+1)}) = \delta_{\sigma(j)}$ и получим искомое инъективное отображение. Пусть существует такое k, что $f_{t+1}(\delta_{\sigma^*(k)}) = \delta_{\sigma(j)}$. Положим $f_{t+1}(\delta_{\sigma^*(k)}) = \delta_{\sigma(2t+2)}$ и $f_{t+1}(\delta_{\sigma^*(t+1)}) = \delta_{\sigma(j)}$. Если $\delta_{\sigma^*(k)} \le \delta_{\sigma(2t+2)}$, то опять найдено требуемое отображение f_{t+1}. В противном случае, учитывая $\Omega_{k-1}(\sigma^*) \le \Omega_t(\sigma^*)$, повторим описанную процедуру переопределения f_{t+1}. Так как $|\{\delta_{\sigma^*(1)}, \delta_{\sigma^*(2)}, \dots, \delta_{\sigma^*(t+1)}\}| < |\{\delta_{\sigma(1)}, \delta_{\sigma(2)}, \dots, \delta_{\sigma(2t+2)}\}|$, и ни один элемент из σ^* не рассматривается дважды, не более чем за $t+1$ итерацию получим инъективное отображение f_{t+1} со свойством (2.9). □

Теорема 4 *Алгоритм 2.1 является 2-приближенным алгоритмом для задачи $W1, 1|p_j = 1, \delta_i \ge 0| \sum w_i C_i$.*

Доказательство. Сначала рассмотрим вспомогательную задачу $1|p_i = 1, r_i| \sum w_i C_i$, в которой надо минимизировать взвешенную сумму моментов окончания единичных работ на одной машине с произвольными временами поступления работ. В этой задаче работа J_i доступна для выполнения с момента r_i. Используя перестановочный прием, легко показать, что следующий "жадный"алгоритм получает оптимальное

расписание. В каждый целый момент времени выбираем доступную работу с наибольшим весом. Обозначим через $\tilde{\sigma}$ полученное расписание.

По примеру I и расписанию σ построим пример $\tilde{I}$ задачи $1|p_i = 1, r_i|\sum w_i C_i$ с работами $\tilde{J}_1, \ldots, \tilde{J}_n$. Положим $r_i = \min\{t|\alpha_i \leq \Omega_{2t}(\sigma)\}$ и $\tilde{w}_i = w_i$, где r_i — момент поступления работы $\tilde{J}_i$, и $\tilde{w}_i$ — вес работы $\tilde{J}_i$. Заметим, что допустимость расписания σ влечет, что $r_i \leq \lfloor \frac{n}{2} \rfloor$ для всех работ $\tilde{J}_i$.

Из леммы 2 следует, что расписание σ^* является допустимым для примера $\tilde{I}$ задачи $1|p_i = 1, r_i|\sum w_i C_i$. Пусть $\tilde{Z}$ — значение целевой функции в оптимальном расписании работ в примере $\tilde{I}$. Тогда $\tilde{Z} \leq Z(\sigma^*)$.

По расписанию σ построим расписание в примере $\tilde{I}$. В момент времени ноль выполним работу J_{σ_1}. Далее последовательно для каждого целого $t \leq \lfloor \frac{n}{2} \rfloor$ выберем из множества $\{J_{\sigma_i}|i = 1, \ldots, 2t+1\}$ невыполненную работу с наибольшим весом. Выполним остальные работы в порядке невозрастания их весов. Покажем, что полученное расписание совпадает с оптимальным расписанием $\tilde{\sigma}$, полученным "жадным"алгоритмом.

Действительно, поскольку σ допустимо, для всех работ из множества $\{J_{\sigma_i}|i = 1, \ldots, 2t+1\}$ выполнено $\alpha_{\sigma_i} \leq \Omega_{2t}$ и, следовательно, $r_{\sigma_i} \leq t$. Таким образом, в построенном расписании все работы начинаются не ранее момента их поступления. Осталось показать, что в каждый момент времени t из невыполненных доступных работ выбирается работа J_j с максимальным w_j. Пусть это не так. Тогда существует работа с $l > 2t+1$, $w_{\sigma_l} > w_j$, и $r_{\sigma_l} \leq t$. Тогда $\alpha_{\sigma_l} \leq \Omega_{2t}$ и работа J_{σ_l} была допустимой в момент времени $2t$, что противоречит выбору алгоритма 2.1.

Так как в построении расписания $\tilde{\sigma}$ из расписания σ в момент времени t выполняется работа из множества $\{J_{\sigma_i}|i = 1, \ldots, 2t+1\}$, то для любой работы J_i выполнено $C_i(\sigma) \leq 2C_i(\tilde{\sigma}) - 1$. Учитывая, что $C_i(\tilde{\sigma}) \leq n$, получаем $Z(\sigma) \leq (2 - \frac{1}{n})\tilde{Z} \leq (2 - \frac{1}{n})Z(\sigma^*)$. □

Следующий пример показывает асимптотическую точность оценки, полученной в теореме 4. Пусть заданы $\Omega_0 = 0$ и множество работ $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$, где n четное число. Параметры работ определим как

$\alpha_i = 0, \beta_i = 0, w_i = 1$, для $i = 1, 2, \ldots, \frac{n}{2}$;
$\alpha_i = 0, \beta_i = 1, w_i = 0$, для $i = \frac{n}{2}+1, \frac{n}{2}+2, \ldots, n$.

Применяя к этому примеру алгоритм 2.1, получим расписание

$$\sigma = (1, \frac{n}{2}+1, 2, \frac{n}{2}+2, \ldots, \frac{n}{2}-1, n)$$

и $Z(\sigma) = 1 + 3 + \cdots + (n-1) = \frac{n^2-n}{4}$.

Оптимальное расписание задается перестановкой $\sigma^* = (1, 2, \ldots, \frac{n}{2}, \frac{n}{2}+1, \ldots, n)$ и $Z(\sigma^*) = 1 + 2 + \cdots + \frac{n}{2} = \frac{n^2+2n}{8}$. Получаем, что $\lim_{n\to\infty} \frac{Z(\sigma)}{Z(\sigma^*)} = 2$.

В заключение отметим, что по теореме 4 алгоритм 2.1 может быть использован для нахождения 2-приближенного решения задачи $W1, 1|\delta_i \leq 0|\sum C_i$.

2.2 Параллельные машины. Минимизация длины расписания. Сложность задач и свойства допустимых расписаний

В данном разделе изучаются задачи минимизации длины расписания множества работ на параллельных идентичных машинах. В задаче $W1, P||C_{max}$ число параллельных машин не фиксировано и может быть различным для каждой индивидуальной задачи. В задаче $W1, Pm||C_{max}$ число параллельных машин фиксировано и равно m. В этом разделе также изучается задача $W1||C_{max}$, в которой отсутствует ресурс типа "машина", то есть произвольное число работ может выполняться параллельно. Другая постановка этой задачи предполагает, что в системе имеется достаточное число параллельных машин, чтобы выполнить любое число работ одновременно, то есть $m \geq n$. Задача $W1||C_{max}$ рассматривается в этом разделе из-за ее близости к задачам на параллельных машинах. Легко понять, что любой алгоритм, решающий задачу $W1, P||C_{max}$, также решает и задачи $W1, Pm||C_{max}$ и $W1||C_{max}$. И обратно, NP-трудность задач $W1, Pm||C_{max}$ и $W1||C_{max}$ влечет NP-трудность задачи $W1, P||C_{max}$. В [31] показано, что задача $W1, P3|p_i = 1|C_{max}$ является NP-трудной в сильном смысле. В той же статье авторы поставили проблему определения комбинаторной сложности задачи $W1, P2|p_i = 1|C_{max}$, то есть задачи с воспроизводимым ресурсом, единичными работами и двумя параллельными идентичными машинами. В этом разделе доказывается, что задача $W1, P2|p_i = 1|C_{max}$ является NP-трудной в сильном смысле, даже если прибыль от каждой работы положительна. Более того, задача остается NP-трудной в сильном смысле, и когда назначение работ по машинам задано. Также в этом разделе приводятся приближенные алгоритмы с гарантированными оценками точности для задач с воспроизводимым ресурсом на параллельных идентичных машинах.

2.2.1 Зеркальный пример и зеркальное расписание

В этом параграфе введем понятие зеркального примера и получим простой но полезный результат о вполне эквивалентности задач с неотрицательной прибылью всех работ и задач с неположительной прибылью всех работ.

Рассмотрим произвольный пример I задачи $W1, P||C_{max}$ с параметрами Ω_0, p_i, α_i, β_i, $i = 1, \ldots, n$, и некоторое допустимое расписание $\sigma(I)$ длины $C_{max}(\sigma)$. Определим к примеру I *зеркальный пример* I_d следующим образом. Положим $p'_i = p_i, \alpha'_i = \beta_i, \beta'_i = \alpha_i, 1 \leq i \leq n$, и $\Omega'_0 = \Omega_0 + \sum_{i=1}^{n}(\beta_i - \alpha_i)$. По расписанию $\sigma(I)$ построим расписание $\sigma_d(I_d)$ для зеркального примера I_d. Определим моменты начала и завершения работы J_i в $\sigma_d(I_d)$ как

$$s_i(\sigma_d) = Z(\sigma) - C_i(\sigma) \text{ и } C_i(\sigma_d) = Z(\sigma) - s_i(\sigma). \tag{2.10}$$

Пусть в расписании $\sigma_d(I_d)$ все работы выполняются на тех же машинах, что и в расписании $\sigma(I)$, тогда расписание $\sigma_d(I_d)$ называется зеркальным расписанием к

$\sigma(I)$, и, наоборот, расписание $\sigma(I)$ называется зеркальным расписанием к $\sigma_d(I_d)$.

Лемма 3 *Расписание $\sigma_d(I_d)$ является допустимым, и его длина равна длине расписания $\sigma(I)$.*

Доказательство. Нетрудно убедиться, что в расписании $\sigma_d(I_d)$ каждая машина в каждый момент времени выполняет не более одной работы и $C_{max}(\sigma_d) = C_{max}(\sigma)$. Требуется проверить, что оно удовлетворяет ограничению на использование воспроизводимого ресурса (2.1). Покажем, что $\bar{\Omega}'_\tau(\sigma_d) \geq 0$ для всех $\tau \in [0, C_{max}(\sigma)]$.

Если $\tau = C_{max}(\sigma)$, то все работы уже завершились к моменту τ. Тогда

$$\bar{\Omega}'_\tau(\sigma_d) = \Omega'_0 + \sum_{\{J_i \in \mathcal{J}\}} (\beta'_i - \alpha'_i) = \Omega'_0 - \sum_{\{J_i \in \mathcal{J}\}} (\beta_i - \alpha_i) = \Omega_0 \geq 0.$$

Пусть $\tau < C_{max}(\sigma)$ и $t = C_{max}(\sigma) - \tau$. Тогда

$$\begin{aligned}\bar{\Omega}'_\tau(\sigma_d) &= \Omega'_0 + \sum_{\{J_i \in \mathcal{J} | C_i(\sigma_d) \leq \tau\}} (\beta'_i - \alpha'_i) - \sum_{\{J_i \in \mathcal{J} | s_i(\sigma_d) \leq \tau < C_i(\sigma_d)\}} \alpha'_i \\ &= \Omega_0 + \sum_{\{J_i \in \mathcal{J}\}} (\beta_i - \alpha_i) + \sum_{\{J_i \in \mathcal{J} | C_i(\sigma_d) \leq \tau\}} (\alpha_i - \beta_i) - \sum_{\{J_i \in \mathcal{J} | s_i(\sigma_d) \leq \tau < C_i(\sigma_d)\}} \beta_i.\end{aligned}$$

Используя равенства (2.10), получим

$$\begin{aligned}\bar{\Omega}'_\tau(\sigma_d) &= \Omega_0 + \sum_{\{J_i \in \mathcal{J} | C_i(\sigma) < t\}} \beta_i - \sum_{\{J_i \in \mathcal{J} | s_i(\sigma) < t\}} \alpha_i \\ &= \Omega_0 + \sum_{\{J_i \in \mathcal{J} | C_i(\sigma) < t\}} (\beta_i - \alpha_i) - \sum_{\{J_i \in \mathcal{J} | s_i(\sigma) < t \leq C_i(\sigma)\}} \alpha_i.\end{aligned}$$

Пусть $t_0 < t$ – последний момент времени, когда какая-то работа начинается или завершается в расписании σ. Если никакая работа не начинается до момента t, то положим $t_0 = 0$. Преобразуем последнее выражение в верхнем равенстве:

$$\begin{aligned}&\Omega_0 + \sum_{\{J_i \in \mathcal{N} | C_i(\sigma) < t\}} (\beta_i - \alpha_i) - \sum_{\{J_i \in \mathcal{N} | s_i(\sigma) < t \leq C_i(\sigma)\}} \alpha_i \\ = \;& \Omega_0 + \sum_{\{J_i \in \mathcal{N} | C_i(\sigma) \leq t_0\}} (\beta_i - \alpha_i) - \sum_{\{J_i \in \mathcal{N} | s_i(\sigma) \leq t_0 < C_i(\sigma)\}} \alpha_i \\ = \;& \bar{\Omega}_{t_0}(\sigma) \geq 0.\end{aligned}$$

Итак, получаем $\bar{\Omega}'_\tau(\sigma_d) \geq 0$ для всех $\tau \in [0, C_{max}(\sigma)]$. □

Непосредственно из леммы 3 вытекает следующая

Теорема 5 *Задачи $W1, P|\delta_i \geq 0|C_{max}$ ($W1, P|\delta_i > 0|C_{max}$) и $W1, P|\delta_i \leq 0|C_{max}$ ($W1, P|\delta_i < 0|C_{max}$) вполне эквивалентны.*

Заметим, что результаты, аналогичные результатам леммы 3 и теоремы 5, также верны и для подзадач задачи $W1, P||C_{max}$, в частности, для задач $W1, Pm||C_{max}$ и $W1||C_{max}$. Более того, из леммы 3 следует, что все результаты раздела для задач с положительной (неотрицательной) прибылью от всех работ также выполняются и для задач с отрицательной (неположительной) прибылью от всех работ.

2.2.2 Задачи с воспроизводимым ресурсом и задача об упаковке в контейнеры

Далее до конца раздела предположим, что длительности всех работ единичны, то есть $p_i = 1$ для всех $J_i \in \mathcal{J}$. В этом случае всегда существует оптимальное расписание, в котором все работы начинаются и завершаются в целые моменты времени, и следовательно, выполняются внутри единичных целочисленных интервалов. При этом уравнение (2.1) примет вид

$$\bar{\Omega}_t(\sigma) = \Omega_0 + \sum_{\{J_i \in \mathcal{N} | C_i(\sigma) \leq t\}} (\beta_i - \alpha_i) - \sum_{\{J_i \in \mathcal{N} | s_i(\sigma) = t\}} \alpha_i \geq 0. \tag{2.11}$$

Рассмотрим задачу $W1|p_i = 1, \delta_i = 0|C_{max}$ с неограниченным числом параллельных идентичных машин, в которой все работы дают нулевую прибыль, т.е. возвращают столько же ресурса сколько и потребляют. Задачу $W1|\delta_i = 0|C_{max}$ можно рассматривать как классическую задачу об упаковке предметов в контейнеры (задача УК).

УПАКОВКА В КОНТЕЙНЕРЫ
Условие: Заданы n предметов, их размеры $\alpha_1, \alpha_2, \ldots, \alpha_n$ и размер контейнеров Ω.
Задача: Найти минимальное целое число k и распределение предметов по контейнерам $f : \{1, \ldots, n\} \to \{1, \ldots, k\}$ такое, что $\sum_{i:f(i)=j} \alpha_j \leq \Omega$.

Поскольку в задаче $W1|p_i = 1, \delta_i = 0|C_{max}$ можно считать, что все работы выполняются внутри единичных интервалов и использованный ресурс в каждом интервале не превосходит Ω, каждый единичный интервал соответствует одному контейнеру размера Ω. При этом длина полученного расписания равна числу использованных контейнеров.

Задача УК NP-трудна в сильном смысле, так как соответствующая ей задача распознавания является NP-полной в сильном смысле (задача ХП-1 в списке NP-полных задач в [4]). Более того, она остается NP-трудной в сильном смысле, даже если число предметов, помещаемых в один контейнер, ограничено тремя. Если в каждый контейнер можно положить не больше двух предметов, задача УК сводится к задаче о нахождении максимального паросочетания и может быть решена за полиномиальное время. Соответственно, задачи $W1|p_i = 1, \delta_i = 0|C_{max}$, $W1, P|p_i = 1, \delta_i = 0|C_{max}$ и $W1, Pm|p_i = 1, \delta_i = 0|C_{max}$ для $m \geq 3$ являются NP-трудными в сильном смысле, а задача $W1, P2|p_i = 1, \delta_i = 0|C_{max}$ — полиномиально разрешимая.

Таким образом, задачи с воспроизводимым ресурсом и единичными длительностями работ можно рассматривать как естественное обобщение задачи УК, в которых размеры контейнеров не являются постоянной величиной и зависят от упаковки предыдущих предметов. В дальнейшем убедимся, что задачи с воспроизводимым ресурсом сложнее соответствующих задач об упаковке в контейнеры. В частности, в следующем параграфе будет доказана NP-трудность в сильном смысле задачи $W1, P2|p_i = 1, \delta_i \geq 0|C_{max}$ построения оптимального по длине расписания единичных работ на двух параллельных машинах с неотрицательной прибылью работ.

Отметим еще один важный результат для задач с воспроизводимым ресурсом, который является следствием известного результата для задачи УК. Для задач $W1|p_i = 1, \delta_i = 0|C_{max}$ и $W1, P|p_i = 1, \delta_i = 0|C_{max}$ существование ρ-приближенного алгоритма с $\rho < \frac{3}{2}$ влечет совпадение классов P и NP. Отсюда, в частности, следует несуществование полиномиальных приближенных схем для этих задач при условии, что P $\neq$ NP. Такой отрицательный результат во многом связан со спецификой задачи УК и достигается на примерах, в которых число используемых контейнеров невелико. При этом для задачи УК известны эффективные алгоритмы, которые имеют асимптотическую погрешность сколь угодно близкую к единице. Применительно к задачам с воспроизводимым ресурсом из знаменитого результата Фернандеса де ла Веги и Луекера [86] следует, что для любого фиксированного $\varepsilon > 0$ существует алгоритм A_ε линейной трудоемкости от входа примера I задачи $W1|p_i = 1, \delta_i = 0|C_{max}$, который строит расписание длины $C_{max}(A_\varepsilon, I) \leq (1 + \varepsilon)C^*_{max}(I) + \frac{1}{\varepsilon^2}$. В параграфе 2.2.4 покажем, что при произвольных δ_i в задаче $W1|p_i = 1, \delta_i \geq 0|C_{max}$ не существует приближенного алгоритма с погрешностью $\rho < \frac{4}{3}$, если P $\neq$ NP.

2.2.3 Две машины. Одинаковые длительности

В этом параграфе рассмотрим задачи, в которых работы имеют неотрицательную прибыль и они должны быть выполнены на двух параллельных машинах. Сначала предположим, что машины являются специализированными, то есть для каждой работы определено, на какой машине она выполняется. Обозначим эту задачу через $W1, D2|p_i = 1, \delta_i \geq 0|C_{max}$. В отсутствие ограничений на воспроизводимый ресурс данная задача даже с произвольными длительностями работ является тривиальной. Покажем, что при добавлении ограничения (2.11) задача построения минимального по длине расписания единичных работ на двух параллельных специализированных машинах становится NP-трудной в сильном смысле.

Сведем к задаче $W1, D2|p_i = 1, \delta_i \geq 0|C_{max}$ NP-полную в сильном смысле задачу ПАРОСОЧЕТАНИЕ С ОГРАНИЧЕНИЯМИ ПО ВЕСУ (задача МР-17 в списке NP-полных задач в [93]).

ПАРОСОЧЕТАНИЕ С ОГРАНИЧЕНИЯМИ ПО ВЕСУ

Условие: Заданы два n-элементных множества $\mathcal{A}_1 = \{1, \ldots, n\}$, $\mathcal{A}_2 = \{n + 1, \ldots, 2n\}$, размеры всех элементов $x_i \in Z^+$, $1 \leq i \leq 2n$, и вектор ограничений $\langle B_1, B_2, \ldots, B_n\rangle$ с положительными целыми координатами.

Вопрос: Можно ли множество $\mathcal{A}_1 \cup \mathcal{A}_2$ так разбить на n непересекающихся подмножеств $A_1, A_2, \ldots, A_n$, что каждое множество A_j содержит ровно по одному элементу из множеств $\mathcal{A}_1$, $\mathcal{A}_2$, и $\sum_{j \in A_i} x_j = B_i$?

Далее без ограничения общности будем считать, что в каждом нетривиальном примере задачи MP-17

$$\sum_{i=1}^{2n} x_i = \sum_{i=1}^{n} B_i. \tag{2.12}$$

Как замечено в [93], NP-полнота в сильном смысле задачи ПАРОСОЧЕТАНИЕ С ОГРАНИЧЕНИЯМИ ПО ВЕСУ может быть получена сведением к ней другой известной задачи ТРЕХМЕРНОЕ СОЧЕТАНИЕ С ОГРАНИЧЕНИЯМИ ПО ВЕСУ (задача MP-16). Напомним, что в последней заданы три n-элементных множества $\mathcal{A}_1 = \{1, \ldots, n\}, \mathcal{A}_2 = \{n+1, \ldots, 2n\}$, $\mathcal{A}_3 = \{2n+1, \ldots, 3n\}$, граница $E \in Z^+$ и такие размеры всех элементов $x_i \in Z^+$, $1 \leq i \leq 3n$, что $\sum_{i=1}^{3n} x_i = nE$ и $\frac{E}{4} < x_i < \frac{E}{2}$. Требуется проверить, можно ли множество $\mathcal{A}_1 \cup \mathcal{A}_2 \cup \mathcal{A}_3$ разбить на n непересекающихся подмножеств $A_1, A_2, \ldots, A_n$ так, что каждое множество A_j содержит ровно по одному элементу из множеств $\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3$, и $\sum_{i \in A_j} x_i = E$.

Без ограничения общности предположим, что $E/4$ — целое число. Если это не так, умножим все целые числа в примере задачи MP-16 на 4 и получим эквивалентный пример с требуемым свойством. По условию задачи MP-16 каждое из множеств $A_i, 1 \leq i \leq n$, содержит ровно один элемент из $\mathcal{A}_3$. Распределим элементы из $\mathcal{A}_3$ по множествам $A_1, A_2, \ldots, A_n$ произвольным образом. Пусть множество A_i содержит элемент $2n+i$, $i = 1, \ldots, n$. Для решения задачи MP-16 остается разместить в каждом множестве A_i ровно по одному элементу из множеств $\mathcal{A}_1$ и $\mathcal{A}_2$, так чтобы $\sum_{j \in A_i \setminus \{2n+i\}} x_j = E - x_{2n+i}$ для всех $i = 1, \ldots, n$. Таким образом, получим пример задачи MP-17. Заметим, что ограничения на размеры элементов в задаче MP-16 позволяют ввести соответствующие ограничения на размеры элементов и значения координат вектора ограничений в задаче MP-17. Далее будем предполагать, что существует число E такое, что $E/4 < x_i < E/2$, $1 \leq i \leq 2n$, $E/2 < B_i < 3E/4$, $1 \leq i \leq n$, и все числа в приведенных неравенствах целые.

Теорема 6 *Задача* $W1, D2|p_j = 1, \delta_j \geq 0|C_{max}$ *является NP-трудной в сильном смысле.*

Доказательство. Представим псевдо-полиномиальное сведение задачи MP-17 к задаче $W1, D2|p_j = 1, \delta_j \geq 0|C_{max}$. Пусть в примере I' задачи MP-17 заданы размеры элементов $\{x_1, x_2, \ldots, x_{2n}\}$ и вектор ограничений $\langle B_1, B_2, \cdots, B_n \rangle$. Предположим, что $B_1 \leq B_2 \leq \cdots \leq B_n$ и существует k различных значений координат вектора ограничений. Тогда цепочку предыдущих неравенств можно переписать в виде $B_1 = \cdots = B_{m_1} < B_{m_1+1} = \cdots = B_{m_1+m_2} < \cdots < B_{m_1+m_2+\cdots+m_{k-1}+1} = \cdots = B_{m_1+m_2+\cdots+m_k} = B_n$, где m_i – мощность i-го множества координат с одинаковыми значениями. Пусть

$M = 1 + \max_{i=1,\ldots,k} m_i$. Положим $\bar{B}_i = B_{m_1+m_2+\cdots+m_i}$ для всех $1 \le i \le k$. Имеем $\bar{B}_i < \bar{B}_{i+1}$ для всех $1 \le i \le k-1$.

Определим пример I задачи $W1, D2|p_j = 1, \delta_j \ge 0|C_{max}$ с $2n$ базовыми работами J_{1i}, $i = 1, 2, \ldots, n$, и J_{2i}, $i = 1, 2, \ldots, n$, и $k-1$ связующей работой $\bar{J}_i$, $i = 1, \ldots, k-1$. Пусть $\Omega_0 = 2MB_1$. Все связующие работы и работы J_{1i}, $i = 1, 2, \ldots, n$, должны быть выполнены на машине 1, все остальные работы должны быть выполнены на машине 2. Положим $\alpha_{1i} = 2Mx_i$ и $\beta_{1i} = 2Mx_i + 1$ для каждой базовой работы J_{1i}, $\alpha_{2i} = 2Mx_{n+i}$ и $\beta_{2i} = 2Mx_{n+i} + 1$ для каждой базовой работы J_{2i}. Заметим, что

$$\beta_{1i} > \alpha_{1i} > ME/2 \quad \text{для всех } i = 1, \ldots, n. \tag{2.13}$$

Обозначим через $\bar{\alpha}_i$, $\bar{\beta}_i$, $\bar{\delta}_i$ расход, доход и прибыль ресурса от связующей работы $\bar{J}_i$ и положим $\bar{\alpha}_i = 2M\bar{B}_i + 2m_i$ и $\bar{\beta}_i = 2M\bar{B}_{i+1}$. Заметим, что $\bar{\delta}_i = 2M\bar{B}_{i+1} - (2M\bar{B}_i + 2m_i) = 2M(\bar{B}_{i+1} - \bar{B}_i) - 2m_i \ge 2M - 2m_i > 0$. Таким образом построенный пример действительно является примером задачи $W1, D2|p_j = 1, \delta_j \ge 0|C_{max}$. Неравенство $\bar{B}_i < \bar{B}_{i+1}$ влечет $\bar{\alpha}_i < \bar{\alpha}_{i+1}$ для всех $i = 1, \ldots, k-2$. Дополнительно имеем

$$\bar{\alpha}_i = 2M\bar{B}_i + 2m_i > 2M\bar{B}_1 = 2MB_1 > ME. \tag{2.14}$$

Для доказательства сильной NP-трудности задачи $W1, D2|p_j = 1, \delta_j \ge 0|C_{max}$ достаточно показать, что искомое разбиение в примере I' существует тогда и только тогда, когда для примера I существует расписание длины не больше чем $n + k - 1$.

Сперва заметим, что максимальный объем доступного ресурса в любой момент времени не превосходит величины $3ME/2$. Действительно,

$$\begin{aligned}
& \Omega_0 + \textstyle\sum_{i=1}^{n} \delta_{1i} + \sum_{i=1}^{n} \delta_{2i} + \sum_{i=1}^{k-1} \bar{\delta}_i \\
= \;& 2MB_1 + 2n + \textstyle\sum_{i=1}^{k-1}(2M\bar{B}_{i+1} - 2M\bar{B}_i - 2m_i) \\
= \;& 2MB_1 + \textstyle\sum_{i=1}^{k} 2m_i + \sum_{i=1}^{k-1}(2M\bar{B}_{i+1} - 2M\bar{B}_i - 2m_i) \qquad (2.15) \\
\le \;& 2M\bar{B}_k + 2m_k \le 2MB_n + 2M - 2 \\
= \;& 2M(B_n + 1 - \tfrac{2}{M}) < \tfrac{3ME}{2}.
\end{aligned}$$

Последнее строгое неравенство следует из предположения о целочисленности величин B_n и $E/4$. Из неравенства (2.15) совместно с неравенствами (2.13) и (2.14) следует, что никакая связующая работа не может выполняться параллельно с любой другой работой. Следовательно, длина любого допустимого расписания в примере I не меньше чем $n + k - 1$. Кроме того, если длина допустимого расписания равна $n + k - 1$, то все базовые работы выполняются в нем по две параллельно в соответствующем единичном интервале.

Лемма 4 *В любом допустимом расписании $\sigma(I)$ связующая работа $\bar{J}_i$, $1 \le i \le k-1$, не может начаться до момента времени $\tau_i = \sum_{j=1}^{i} m_j + i - 1$.*

Доказательство. Докажем утверждение индукцией по i. Так как $\bar{\alpha}_1 = 2M\bar{B}_1 + 2m_1 > 2M\bar{B}_1$, то никакая связующая работа не может начаться в момент времени ноль. Напомним, что прибыль от каждой базовой работы равна 1 и не более двух работ можно выполнить за единицу времени. Следовательно, в любом допустимом расписании работа J_1 может начаться не ранее момента m_1.

Пусть утверждение верно для некоторого $i \geq 1$. Тогда количество свободного ресурса в момент времени $\sum_{j=1}^{i} m_j + i$ не превышает $2M\bar{B}_1 + \sum_{j=1}^{i} 2m_j + \sum_{j=1}^{i}(2M\bar{B}_{j+1} - 2M\bar{B}_j - 2m_j) = 2M\bar{B}_{i+1}$. Расход ресурса α_j у любой из связующих работ $\bar{J}_j$ с $j \geq i+1$, не меньше $2M\bar{B}_{i+1} + 2m_{i+1}$. Следовательно, работа J_{i+1} не может стартовать до момента $\sum_{j=1}^{i} m_j + i + m_{i+1} = \sum_{j=1}^{i+1} m_j + (i+1) - 1$, и по индукции утверждение леммы доказано. □

Продолжим доказательство теоремы 6. Предположим, что в примере I' существует требуемое разбиение множества элементов на n непересекающихся множеств $A_1, A_2, \ldots, A_n$ так, что $\sum_{e_j \in A_i} x_j = B_i, 1 \leq i \leq n$. Сгруппируем базовые работы в пары согласно их индексам в решении примера I'. Пусть множество A_i содержит элементы i_1 и $n + i_2$. Тогда имеем $\alpha_{1i_1} + \alpha_{2i_2} = 2MB_i, 1 \leq i \leq n$. Пусть $m_0 = 0$. Построим допустимое расписание $\sigma(I)$ следующим образом. Назначим начало связующей работы $\bar{J}_i$ в момент $\sum_{j=1}^{i} m_j + i - 1$ для всех $1 \leq i \leq k-1$ и упорядочим пары базовых работ с суммарным расходом $2M\bar{B}_i$, $1 \leq i \leq k$, в целочисленные интервалы времени $[\sum_{j=0}^{i-1} m_j + i - 1, \sum_{j=1}^{i} m_j + i - 1)$ в произвольном порядке. Легко проверить, что построенное расписание является допустимым и его длина равна $n + k - 1$.

Предположим, что существует расписание $\sigma(I)$ длины $n+k-1$. Как было замечено ранее, все базовые работы в таком расписании выполняются параллельно по две работы в соответствующем единичном интервале. Занумеруем пары базовых работ от 1 до n в том порядке, в котором они появляются в расписании $\sigma(I)$. Пусть B'_i – суммарный расход ресурса для выполнения двух работ из пары i. Если $B'_i = B_i$ для всех i, то получаем требуемое разбиение в примере I'. В противном случае, из равенства (2.12) следует, что существует пара работ i, для которой $B'_i > B_i$. Пусть $B_i = \bar{B}_l$. Тогда из определения $\bar{B}_l$ имеем $i \leq \sum_{j=1}^{l} m_j$. Из леммы 4 следует, что не более чем $l-1$ связующая работа заканчивается к моменту $\sum_{j=1}^{l} m_j + l - 1$. Отсюда вытекает, что базовые работы из пары i начнут выполняться не ранее момента $t \leq \sum_{j=1}^{l} m_j + l - 2$.

$$\begin{aligned}\Omega_t(\sigma) \leq \;& \Omega_0 + 2\sum_{j=1}^{i}\delta_j + \sum_{j=1}^{l-1}\bar{\delta}_i \\ \leq \;& 2M\bar{B}_1 + 2\sum_{j=1}^{l} m_j + \sum_{j=1}^{l-1}(2M\bar{B}_{j+1} - 2M\bar{B}_j - 2m_j) \\ = \;& 2M\bar{B}_l + 2m_l < 2M\bar{B}_l + 2M \leq 2M\bar{B}'_i.\end{aligned}$$

Последнее неравенство следует из неравенства $B'_i > B_i$ и целочисленности величин B'_i и B_i. Таким образом, объема ресурса в момент t недостаточно для выполнения соответствующей пары базовых работ, следовательно, получаем противоречие с предположением, что существует пара работ i, для которой $B'_i > B_i$. □

Итак, теорема 6 устанавливает NP-трудность в сильном смысле задачи $W1, D2|p_j = 1, \delta_j \geq 0|C_{max}$. Более того, из доказательства теоремы 6 вытекает, что следующая задача распознавания также является NP-полной в сильном смысле.

НОРМАЛЬНОЕ РАСПИСАНИЕ В $W1, D2|p_j = 1, \delta_j \geq 0|C_{max}$ (задача НР).
Условие: Заданы две машины и $n+l$ единичных работ, начальный объем Ω_0 воспроизводимого ресурса. Работы $J_1, \ldots, J_n$ должны быть выполнены на машине 1, и работы $J_{n+1}, \ldots, J_{n+l}$ — на машине 2. Каждая работа J_j потребляет α_j единиц ресурса перед началом выполнения и воспроизводит β_j единиц ресурса в момент своего завершения, $0 < \alpha_j \leq \beta_j$, $j = 1, \ldots, n+l$.
Вопрос: Существует ли нормальное расписание, то есть расписание длины $\max\{n, l\}$?

Сведем задачу НР к задаче $W1, P2|p_j = 1, \delta_j \geq 0|C_{max}$. Пусть $\Psi = \Omega_0 + \sum_{J_i \in \mathcal{J}} \delta_i$. Без ограничения общности предположим, что $n \geq l$. Пример $\bar{I}$ задачи $W1, P2|p_j = 1, \delta_j \geq 0|C_{max}$ содержит $n + l$ работ, и $\bar{\Omega}_0 = \Psi + \Omega_0$. Первые n работ назовем *тяжелыми* и положим $\bar{\alpha}_i = \Psi + \alpha_i$, $\bar{\beta}_i = \Psi + \beta_i$ для $i = 1, \ldots, n$. Последние l работ назовем *легкими* и положим $\bar{\alpha}_i = \alpha_i$, $\bar{\beta}_i = \beta_i$ для $i = n+1, \ldots, n+l$.

Покажем, что в примере I задачи НР существует нормальное расписание (расписание длины n) тогда и только тогда, когда в примере $\bar{I}$ существует расписание $\sigma(\bar{I})$ длины не больше n.

Заметим, что любое нормальное допустимое расписание $\sigma(I)$ в примере I также является допустимым расписанием в примере $\bar{I}$. Действительно, объем ресурса, доступного в каждый момент времени, зависит от начального объема ресурса и прибыли от завершившихся работ. Начальный объем ресурса в примере $\bar{I}$ увеличился на Ψ по сравнению с начальным объемом ресурса в примере I. Прибыль каждой работы в обоих примерах совпадает. Следовательно, в каждый момент времени t объем доступного ресурса в расписании $\sigma(\bar{I})$ на Ψ единиц больше, чем в расписании $\sigma(I)$. Так как все тяжелые работы в этом расписании выполняются на машине 1, то

в каждом единичном интервале выполняется не более одной тяжелой работы. Следовательно, потребление ресурса в каждом единичном интервале возрастет не более чем на Ψ, и расписание $\sigma(\bar{I})$ является допустимым расписанием в примере $\bar{I}$, причем $C_{max}(\sigma(\bar{I})) = n$.

Так как 2Ψ является верхней оценкой на максимальный объем доступного ресурса, то в любой момент времени для любого расписания примера $\bar{I}$ никакие две тяжелые работы не могут выполняться параллельно. Следовательно, длина любого допустимого расписания не меньше n. Пусть расписание $\bar{\sigma}(\bar{I})$ имеет длину n. Следовательно, в каждом единичном интервале от 0 до n выполняется одна из тяжелых работ. Поскольку все работы выполняются на двух параллельных машинах и длина расписания равна n, не более одной легкой работы дополнительно выполняются в l некоторых единичных интервалах от 0 до n в $\bar{\sigma}(\bar{I})$. Таким образом, можно переназначить все тяжелые работы на машину 1 и все легкие на машину 2, получая допустимое расписание в примере I. Из допустимости расписания $\bar{\sigma}(\bar{I})$ следует, что для всех $t = 0, 1, \ldots, n$, выполнено

$$\begin{aligned} 0 \leq \bar{\Omega}_t(\bar{\sigma}(\bar{I})) &= \Psi + \Omega_0 + \sum_{\{J_i \in \mathcal{N} | C_i(\bar{\sigma}) \leq t\}} (\bar{\beta}_i - \bar{\alpha}_i) - \sum_{\{J_i \in \mathcal{N} | s_i(\bar{\sigma}) = t\}} \bar{\alpha}_i \\ &= \Psi + \Omega_0 + \sum_{\{J_i \in \mathcal{N} | C_i(\bar{\sigma}) \leq t\}} (\beta_i - \alpha_i) \\ &- \sum_{\{J_i \in \mathcal{N} | s_i(\bar{\sigma}) = t\}} \alpha_i - \Psi = \bar{\Omega}_t(\bar{\sigma}(I)). \end{aligned}$$

Следовательно, расписание $\bar{\sigma}(I)$ удовлетворяет ограничениям на ресурсы, имеет длину n и является допустимым расписанием в примере I. Таким образом, доказана следующая

Теорема 7 *Задача* $W1, P2|p_j = 1, \delta_j \geq 0|C_{max}$ *является NP-трудной в сильном смысле.*

2.2.4 Неограниченное число машин. Единичные длительности и положительная прибыль всех работ. Неаппроксимируемость

В этом параграфе установим нижнюю границу на аппроксимируемость задачи $W1|p_i = 1, \delta_i \geq 0|C_{max}$. Для этого покажем, что асимптотический ρ-приближенный алгоритм с $\rho < \frac{4}{3}$ для задачи $W1|\delta_j \geq 0|C_{max}$ можно использовать для решения классической NP-полной задачи РАЗБИЕНИЕ (задача МР-12 в списке NP-полных задач в [93]).

РАЗБИЕНИЕ

Условие: Задано конечное множество $X = \{1, \ldots, n\}$ и размеры всех элементов $e_i \in Z^+$, $i = 1, \ldots, n$ такие, что $\sum_{i=1}^{n} e_i = 2E$.

Вопрос: Существует ли такое подмножество $X_1 \subseteq X$, что $\sum\limits_{e_i \in X_1} e_i = E$?

Теорема 8 *Существование асимптотического ρ-приближенного алгоритма с $\rho < \frac{4}{3}$ для задачи $W1|\delta_j \geq 0|C_{max}$ влечет совпадение классов P и NP.*

Доказательство. По примеру I задачи РАЗБИЕНИЕ определим пример I' задачи $W1|\delta_j \geq 0|C_{max}$. Пусть $K > 0$ – фиксированное целое число. Пример I' содержит $n(K+1)$ базовых работ J_{ki} с $\alpha_{ki} = (E+1)^k e_i$ и $\beta_{ki} = (E+1)^k e_i + \frac{1}{n}$ для всех $k = 0, 1, \ldots, K$ и $i = 1, 2, \ldots, n$. Заметим, что для удобства вычислений параметры β_{ki} не удовлетворяют ограничению целочисленности. Эта небольшая вольность в изложении может быть легко устранена умножением всех параметров на число n.

Сгруппируем базовые работы в $K+1$ множество из n работ согласно их первому индексу, то есть $N_k = \{J_{k1}, J_{k2}, \ldots, J_{kn}\}$. Кроме того, определим K связующих работ $\bar{J}_k, 1 \leq k \leq K$, с $\bar{\alpha}_k = E(E+1)^{k-1} + 1$ и $\bar{\beta}_k = E(E+1)^k$. Объем начального ресурса зададим равным $\Omega_0 = E$.

Легко проверить, что в любом допустимом расписании $\sigma(I')$ для примера I' работы выполняются в порядке $N_0, \{\bar{J}_1\}, N_1, \{\bar{J}_2\}, N_2, \{\bar{J}_3\}, N_3, \cdots, \{\bar{J}_K\}, N_K$. Действительно, так как $\Omega_0 = E$, то только работы из множества N_0 могут начаться в момент ноль. Более того, так как прибыль от каждой базовой работы равна $\frac{1}{n}$, то только связующая работа $\bar{J}_1$ станет доступной после завершения всех работ из множества N_0. После завершения работы J_1 объем свободного ресурса составит $E(E+1)$ и появится возможность выполнить работы множества N_1. Повторяя подобную цепочку рассуждений для остальных работ, получим, что расписание имеет вид $\sigma = N_0 \circ \{\bar{J}_1\} \circ N_1 \circ \{\bar{J}_2\} \circ N_2 \circ \{\bar{J}_3\} \circ N_3 \circ \cdots \circ \{\bar{J}_K\} \circ N_K$. Кроме того, из рассуждений видно, что в начале своего выполнения каждая связующая работа $\bar{J}_1$ поглощает весь доступный ресурс и, следовательно, не может выполняться параллельно ни с какой другой работой.

Теперь рассмотрим, как можно расписать работы множества N_k для произвольного $0 \leq k \leq K$. Суммарный расход ресурса работ из множества N_k равен $\sum\limits_{i=1}^{n} \alpha_{ki} = \sum\limits_{i=1}^{n} (E+1)^k e_i = 2E(E+1)^k$Объем свободного ресурса после выполнения работ $N_0 \circ \{\bar{J}_1\} \circ N_1 \circ \{\bar{J}_2\} \circ N_2 \circ \{\bar{J}_3\} \circ N_3 \circ \cdots \circ \{\bar{J}_k\}$ равен $E(E+1)^k$. Так как $\delta_{ki} = \frac{1}{n}$, и все e_i – целые, то работы множества N_k можно выполнить в интервале длины 2 тогда и только тогда, когда существует разбиение работ в два множества таких, что $\sum\limits_{e_i \in X_1} e_i = \sum\limits_{e_i \in X_1 \setminus X} e_i = E$. Пусть $OPT(I')$ – длина оптимального расписания в примере I'. Если существует подмножество $X_1 \subseteq X$ такое, что $\sum_{e_i \in X_1} e_i = E$ в примере I задачи РАЗБИЕНИЕ, то $OPT(I') = 2(K+1) + K = 3K + 2$. В случае, если задача РАЗБИЕНИЕ не имеет решения, $OPT(I') = 3(K+1) + K = 4K + 3$.

Предположим, что существует асимптотический ρ-приближенный алгоритм *App* с $\rho < \frac{4}{3}$ для задачи $W1|\delta_j \geq 0|C_{max}$. Пусть $\rho = \frac{4}{3} - \varepsilon$ для некоторого фиксированного $\varepsilon \in (0, \frac{1}{3}]$ и пусть $c \geq 0$ фиксированная константа. Положим $K = \lceil \frac{c}{3\varepsilon} \rceil$. Если пример I задачи РАЗБИЕНИЕ имеет решение, то алгоритм *App* найдет расписание длины

$$\begin{aligned} A(I') &\leq \rho \times OPT(I') + c = (\tfrac{4}{3} - \varepsilon)(3K + 2) + c \\ &= 4K + \tfrac{8}{3} - 2\varepsilon - 3K\varepsilon + c < 4K + \tfrac{8}{3}. \end{aligned}$$

Эта величина меньше чем $4K+3$ и, следовательно, задача РАЗБИЕНИЕ будет решена алгоритмом *App* за полиномиальное время, что влечет совпадение классов P и NP. □

2.3 Параллельные машины. Минимизация длины расписания. Приближенные алгоритмы

В этом разделе рассмотрим жадные приближенные алгоритмы для задач с воспроизводимым ресурсом на параллельных машинах. В основе этих алгоритмов лежит идея частичного выполнения работ в случае, когда свободного ресурса не хватает для выполнения выбранной работы. При этом получается недопустимое расписание, которое с одной стороны дает нижнюю оценку на длину оптимального расписания, а с другой стороны может быть перестроено в допустимое так, что его длина увеличится не более чем в два раза, если длительность всех работ одинакова. В случае произвольных длительностей та же идея, примененная к округленному примеру, позволяет получить $O(\log n)$-приближенный алгоритм.

2.3.1 Приближенные алгоритмы для задач с единичными длительностями

Сначала рассмотрим задачу $W1|p_i = 1, \delta_i \geq 0|C_{max}$ с единичными длительностями, в которых произвольное число работ может выполняться параллельно.

Напомним, что в расписании σ работа J_i является доступной в момент t, если $\alpha_i \leq \Omega_t(\sigma)$. Предположим, что каждую работу можно разделить на фрагменты, которые могут выполняться в одном или нескольких, не обязательно последовательных единичных интервалах. Пусть $0 \leq x_{it} \leq 1$ – фрагмент работы J_i, который выполняется в интервале $(t, t+1)$. Тогда работа J_i потребляет $\alpha_i x_{it}$ единиц ресурса в момент t и производит $\beta_i x_{it}$ единиц ресурса в момент $t+1$. Если $x_{it} > 0$, то длина работы J_i не зависит от величины x_{it}, и она выполняется в течение всего интервала $(t, t+1)$. Будем называть работу J_i *незавершенной* в момент $\tau \geq 1$, если $\sum_{t=0}^{\tau-1} x_{it} < 1$. Все работы являются незавершенными в момент 0. Назначение σ работ по единичным интервалам назовем *слабо допустимым расписанием*, если выполнены следующие условия:

(У1) $x_{it} = 0$ для всех работ J_i с $\alpha_i > \Omega_t(\sigma)$, $t = 0, \ldots, C_{\max}(\sigma) - 1$,

(У2) $\sum_{i=1}^{n} \alpha_i x_{it} \leq \Omega_t(\sigma)$ для всех $t = 0, \ldots, C_{\max}(\sigma) - 1$,

(У3) $\sum_{t=0}^{C_{\max}(\sigma)-1} x_{it} = 1$, для всех $i = 1, \ldots, n$.

Заметим, что назначение работ в единичные интервалы задает допустимое расписание, если оно удовлетворяет трем сформулированным выше условиям и $x_{it} \in \{0, 1\}$. Следовательно, длина минимального слабо допустимого расписания является нижней оценкой на длину оптимального расписания в задаче $W1|p_i = 1, \delta_i \geq 0|C_{\max}$.

Рассмотрим произвольный пример I задачи $W1|p_i = 1, \delta_i \geq 0|C_{\max}$. Вектор $\overrightarrow{v_\sigma} = \langle \Omega_0(\sigma), \Omega_1(\sigma), \ldots, \Omega_{C_{\max}(\sigma)}(\sigma) \rangle$ назовем *вектором ресурса* слабо допустимого расписания $\sigma(I)$. Пусть σ и σ' два слабо допустимых расписания и $T \leq \min\{C_{\max}(\sigma), C_{\max}(\sigma')\}$. Будем говорить, что вектор $\overrightarrow{v_\sigma}$ не меньше вектора $\overrightarrow{v_{\sigma'}}$ на отрезке $[0, T]$, и обозначать $\overrightarrow{v_\sigma} \prec_T \overrightarrow{v_{\sigma'}}$, если $\Omega_t(\sigma) \geq \Omega_t(\sigma')$ для всех $0 \leq t \leq T$. Вектор ресурса называется *максимальным* на отрезке $[0, T]$, если $\overrightarrow{v_\sigma} \prec_T \overrightarrow{v_{\sigma'}}$ для любого слабо допустимого расписания σ'.

Приведем жадный алгоритм построения слабо допустимого расписания и установим некоторые его свойства. Начиная с интервала $[0, 1)$, жадный алгоритм назначает доступные работы так, чтобы максимизировать суммарный доход назначенных работ. Таким образом, этот алгоритм на каждом шаге решает хорошо известную задачу ДРОБНЫЙ РЮКЗАК [130] на множестве доступных работ.

ДРОБНЫЙ РЮКЗАК

Условие: Задано n предметов, их удельные веса $\alpha_1, \ldots, \alpha_n \in \mathbf{Q}^+$, стоимости $\beta_1, \ldots, \beta_n \in \mathbf{Q}^+$, объемы $\bar{y}_1, \bar{y}_2, \ldots, \bar{y}_n \in \mathbf{Q}^+$ и размер рюкзака V.

Требование: Найти рациональные числа $y_1, y_2, \ldots, y_n$ такие, что $0 \leq y_i \leq \bar{y}_i$ для всех $i = 1, \ldots, n$, $\sum_{j=1}^{n} \alpha_j y_j \leq V$, и при этом $\sum_{j=1}^{n} \beta_j y_j$ является максимальной.

Следующий результат позволяет построить простой алгоритм, который решает задачу ДРОБНЫЙ РЮКЗАК(A) на множестве предметов A, сортируя их соответствующим образом.

Утверждение 1 [81] *Пусть $\alpha_1, \ldots, \alpha_n$, $\beta_1, \ldots, \beta_n$ и V неотрицательные целые числа такие, что*

$$\frac{\beta_1}{\alpha_1} \geq \frac{\beta_2}{\alpha_2} \geq \cdots \geq \frac{\beta_n}{\alpha_n},$$

и пусть $k = \min\{j \in \{1, \ldots, n\} : \sum_{i=1}^{j} \alpha_i \bar{y}_i > V\}$. Тогда оптимальное решение $\langle y \rangle = (y_1, y_2, \ldots, y_n)$ для данного примера задачи ДРОБНЫЙ РЮКЗАК определя-

ется формулами

$$\begin{aligned} y_j &:= \bar{y}_j, && \text{для} \quad j = 1, \ldots, k-1; \\ y_k &:= \frac{V - \sum_{j=1}^{k-1} \alpha_j \bar{y}_j}{\alpha_k}; \\ y_j &:= 0, && \text{для} \quad j = k+1, \ldots, n. \end{aligned}$$

Используя решение задачи ДРОБНЫЙ РЮКЗАК, приведем алгоритм, строящий слабо допустимое расписание в задаче $W1|p_i = 1, \delta_i \geq 0|C_{max}$ минимальной длины. Для каждого единичного интервала, начиная с интервала $[0, 1)$, алгоритм 2.2 последовательно вычисляет множество работ, которые будут в нем выполняться.

Алгоритм 2.2

1: Положить $A := \emptyset$, $t := 0$, $v := \Omega_0$, и $\bar{x}_j := 1$ для $j = 1, \ldots, n$.
2: **for** $j := 1$ **to** n **do**
3: **if** $\alpha_j \leq V$ **then**
4: положить $\bar{y}_j := \bar{x}_j$ и $A := A \cup \{j\}$.
5: **if** $A = \emptyset$ **then**
6: **stop** (нет доступных работ).
7: **else**
8: Найти $\langle y \rangle$ решение задачи ДРОБНЫЙ РЮКЗАК(A).
9: Положить $x_{jt} := y_j$, $\bar{x}_j := \bar{x}_j - x_{jt}$ для $j = 1, \ldots, n$.
10: Положить $v := v + \sum_{j=1}^{n} x_{jt}\delta_j$ и $t := t + 1$.
11: **if** $\bar{x}_j = 0$ для всех $j = 1, \ldots, n$ **then**
12: Выдать слабо допустимое расписание x_{it}.
13: **else**
14: **Go to** 2.

Лемма 5 *Пусть I — некоторый пример задачи $W1|\delta_j \geq 0|C_{max}$.*

1. *Если по окончании работы алгоритма 2.2 остались незавершенные работы, то не существует допустимого расписания в примере I.*

2. *Если в примере I существует допустимое расписание, то алгоритм 2.2 находит минимальное по длине слабо допустимое расписание σ^*.*

3. *Вектор ресурса $\overrightarrow{v_{\sigma^*}}$ является максимальным на отрезке $[0, {}_{max}(\sigma^*)]$.*

Доказательство. Предположим, что после остановки алгоритма 2.2 существует непустое множество $\mathcal{J}'$ незавершенных работ . Тогда для каждой работы $J_i \in \mathcal{J}'$ выполняется неравенство $\alpha_i > \Omega_0 + \sum_{J_j \in \mathcal{J} \setminus \mathcal{J}'} \delta_j$. Пусть существует допустимое расписание σ. Рассмотрим работу $J_i \in \mathcal{J}'$, которая начинается в σ не позднее любой другой

работы из $\mathcal{J}'$. Обозначим через $\mathcal{J}''$ множество работ в σ, завершившихся до начала работы J_i. Заметим, что $\mathcal{J}'' \subseteq \mathcal{J} \setminus \mathcal{J}'$. Тогда $\alpha_i \leq \Omega_0 + \sum\limits_{j \in \mathcal{J}''} \delta_j \leq \Omega_0 + \sum\limits_{j \in \mathcal{J} \setminus \mathcal{J}'} \delta_j < \alpha_i$. Второе неравенство следует из условия, что прибыль всех работ неотрицательна. Получаем противоречие, которое влечет справедливость первого утверждения леммы.

Пусть y_{it} — произвольное слабо допустимое расписание. Покажем, что решение y_{it} может быть трансформировано в решение x_{it} полученное алгоритмом 2.2 без увеличения длины расписания. Рассмотрим первый единичный интервал $[\tau, \tau+1)$, в котором $x_{i\tau} > y_{i\tau}$ для некоторого i. Так как оба расписания совпадают до момента τ, то объем доступного ресурса Ω_τ в этот момент одинаков в обоих расписаниях. Следовательно, множество доступных работ в обоих расписаниях в момент τ также совпадает. Перенумеруем доступные работы в том же порядке, как они занумерованы в алгоритме 2.2, и пусть J_j — работа с наименьшим индексом таким, что $x_{j\tau} > y_{j\tau}$. Так как $\sum\limits_{t=0}^{\tau-1} x_{jt} = \sum\limits_{t=0}^{\tau-1} y_{jt}$, то существует момент $\tau' > \tau$ такой, что $y_{j\tau'} > 0$. Если $\sum\limits_{J_i \in \mathcal{J}} y_{i\tau}\alpha_i < \Omega_\tau$, то увеличим $y_{j\tau}$ на

$$\frac{\min\{\alpha_j y_{j\tau'}, \Omega_\tau - \sum_{i \in N} \alpha_i y_{i\tau}\}}{\alpha_j}.$$

В противном случае существует некоторая работа J_k с $k \geq j$ и $y_{k\tau} > 0$. Пусть $\alpha = \min\{\alpha_j y_{j\tau'}, \alpha_k y_{k\tau}\}$. Так как прибыль каждой работы неотрицательна, то в любом расписании $\Omega_{\tau'} \geq \Omega_\tau$, и работа J_k доступна в момент τ'. Поменяем местами $\frac{\alpha}{\alpha_j}$-фрагмент работы J_j и $\frac{\alpha}{\alpha_k}$-фрагмент работы J_k. Обозначим через Ω'_t объем ресурса в момент t в новом расписании. Тогда $\Omega'_t = \Omega_t$ для всех $t \leq \tau$ и $t > \tau'$. Для $\tau < t \leq \tau'$, получаем $\Omega'_t = \Omega_t + \beta_j \frac{\alpha}{\alpha_j} - \beta_k \frac{\alpha}{\alpha_k} = \Omega_t + \alpha(\frac{\beta_j}{\alpha_j} - \frac{\beta_k}{\alpha_k}) \geq \Omega_t$. Следовательно, после перестановки фрагментов работ J_j и J_k все ограничения на ресурсы не нарушены и полученное расписание является слабо допустимым. Повторяя приведенное рассуждение, из расписания y_{it} получим расписание x_{it}. Заметив, что при каждом преобразовании длина полученного расписания не увеличивается и объем доступного ресурса в каждой целочисленной точке не уменьшается, получим справедливость второго и третьего утверждения леммы. □

Отметим, что алгоритм 2.2 применим к случаю, когда прибыль от каждой работы неотрицательна. Рассмотрим пример I задачи $W1|p_i = 1, \delta_i \leq 0|C_{max}$, в котором прибыль каждой работы неположительна. Положим $t_d = C_{max}(\sigma^*) - t - 1$. Тогда назначение $y_{it_d} = x_{it}$ соответствует зеркальному расписанию σ_d к σ. Следовательно, применив алгоритм 2.2 к зеркальному примеру I_d, можно использовать полученное решение для построения минимального по длине слабо допустимого расписания в примере I.

Рассмотрим задачу $W1|p_i = 1|C_{max}$, в который величины δ_i — произвольные целые числа. Обозначим через $\mathcal{J}^+$ множество работ с $\alpha_i \leq \beta_i$ и через $\mathcal{J}^-$ множество работ с $\alpha_i > \beta_i$. Следующее утверждение является прямым следствием процеду-

ры проверки существования допустимого решения в задаче $W1,1||C_{max}$, описанной в [118].

Утверждение 2 *Если существует допустимое расписание в примере I задачи $W1|p_i = 1|C_{max}$, то существует допустимое расписание, в котором все работы множества $\mathcal{J}^-$ начинаются после завершения всех работ из множества $\mathcal{J}^+$.*

Итак, пусть I — произвольный пример задачи $W1|p_i = 1|C_{max}$ с множеством работ $\mathcal{J}$ и начальным объемом ресурса Ω_0. Определим два новых примера I^+ и I^-. Пусть пример I^+ состоит из множества работ $\mathcal{J}^+$ и начального объема ресурса Ω_0, а пример I^- состоит из множества работ $\mathcal{J}^-$ и начального объема ресурса $\Omega_0 + \sum_{J_i \in \mathcal{J}^+} \delta_i$. Алгоритм 2.3 строит слабо допустимое решение задачи $W1||C_{max}$.

Алгоритм 2.3

1: Применить Алгоритм 2.2 к I^+. Пусть $\sigma(I^+)$ — полученное расписание.
2: Применить Алгоритм 2.2 к зеркальному примеру I_d^-. примера I^-.
Пусть $\sigma(I_d^-)$ — полученное расписание и $\sigma(I^-)$ — зеркальное к нему.
3: Выдать $\sigma(I) = \sigma(I^+)\sigma(I^-)$.

Следующий результат является прямым следствием утверждения 2 и первого утверждения леммы 5.

Лемма 6 *Пусть I — некоторый пример задачи $W1||C_{max}$. Если в решении $\sigma(I)$ существуют незавершенные работы, то не существует допустимого расписания в примере I.*

Пусть $\sigma^*(I) = \sigma^*(I^+)\sigma^*(I^-)$ — расписание, полученное алгоритмом 2.3. Будем говорить, что

- интервал $(t, t+1)$ — *заполненный*, если $\sum_{J_i \in N} \alpha_i x_{it} = \Omega_t(\sigma^*)$;
- интервал $(t, t+1)$ — *свободный*, если $\sum_{J_i \in N} \alpha_i x_{it} < \Omega_t(\sigma^*)$ и существует работа $J_i \in \mathcal{J}$ такая, что $x_{it} = 1$;
- интервал $(t, t+1)$ — *фиктивный*, если $\sum_{J_i \in N} \alpha_i x_{it} < \Omega_t(\sigma^*)$, и $x_{it} < 1$ для всех работ $J_i \in \mathcal{J}$.

Установим несколько простых свойств расписания $\sigma^*(I)$.

Лемма 7 *Пусть* $\sigma_d^*(I)$ *— зеркальное расписание к* $\sigma^*(I)$, *и* $t_d = C_{max}(\sigma^*) - t - 1$. *Если* $(t, t+1)$ *— заполненный интервал в* $\sigma^*(I)$, *то* (t_d, t_d+1) *— заполненный интервал в* $\sigma_d^*(I)$.

Доказательство. По определению зеркального расписания и заполненного интервала имеем

$$\Omega_{t_d}(\sigma_d^*) = \Omega_{t+1}(\sigma^*) = \Omega_t(\sigma^*) - \sum_{J_i \in \mathcal{J}} \alpha_i x_{it} + \sum_{J_i \in \mathcal{J}} \beta_i x_{it} = \sum_{J_i \in \mathcal{J}} \beta_i x_{it} = \sum_{J_i \in \mathcal{J}} \beta_i y_{it_d}.$$

□

Для упрощения записи обозначим через ζ_t интервал $(t, t+1)$. Пусть Φ^+ обозначает множество заполненных интервалов в $\sigma^*(I^+)$, Φ^- – множество заполненных интервалов в $\sigma^*(I^-)$, Ψ^+ – множество свободных интервалов в $\sigma^*(I^+)$ и Ψ^- – множество свободных интервалов в $\sigma^*(I^-)$. Пусть $A = \sum_{J_i \in N} \alpha_i$.

Лемма 8 *Если* $|\Psi^+| = 1$, *то*

$$A > \sum_{\zeta_t \in \Phi^+} \Omega_t(\sigma^*) + \sum_{\zeta_t \in \Phi^-} \Omega_t(\sigma^*). \tag{2.16}$$

Если $|\Psi^+| > 1$ *и* $\tau_1, \ldots, \tau_{|\Psi^+|}$ *– свободные интервалы в* $\sigma^*(I^+)$, *то*

$$A > \sum_{\zeta_t \in \Phi^+} \Omega_t(\sigma^*) + \sum_{\zeta_t \in \Phi^-} \Omega_t(\sigma^*) + \sum_{i=1}^{|\Psi^+|-1} \Omega_{\tau_i}(\sigma^*). \tag{2.17}$$

Доказательство. По определению заполненного интервала суммарный расход работ, помещенных в эти интервалы, не меньше суммарного объема, подсчитанного в первых двух слагаемых правой части неравенств (2.16) и (2.17).

Если $|\Psi^+| = 1$, то строгое неравенство в (2.16) следует из того, что в свободном интервале выполняется по крайней мере одна целая работа.

Пусть $|\Psi^+| > 1$. Рассмотрим два свободных интервала ζ_{τ_i} и $\zeta_{\tau_{i+1}}$ для $i = 1, \ldots, |\Psi^+| - 1$. По определению свободного интервала существует работа $J_j \in \mathcal{J}$ с $x_{j\tau_{i+1}} = 1$. Так как эта работа не попала в интервал ζ_{τ_i}, то $\alpha_j > \Omega_{\tau_i}(\sigma^*)$. Кроме того, из равенства $x_{j\tau_{i+1}} = 1$ следует, что эта работа не входит в заполненные интервалы, и ее расход не учтен в первых двух слагаемых правой части неравенства (2.17). Отсюда получим

$$\sum_{i=1}^{|\Psi^+|} \sum_{j=1}^{n} \alpha_j x_{j\tau_i} > \sum_{i=1}^{|\Psi^+|-1} \Omega_{\tau_i}(\sigma^*).$$

□

Лемма 9 *В любом допустимом расписании* $\sigma(I)$

$$C_{max}(\sigma) \geq OPT \geq |\Phi^+| + |\Psi^+| + |\Phi^-|.$$

Доказательство. Пусть $\sigma^*(I) = \sigma^*(I^+)\sigma^*(I^-)$ – расписание, полученное алгоритмом 2.3 для примера I. Пусть $T = C_{max}(\sigma^*(I^+))$. По лемме 5 вектор ресурса $\overrightarrow{v_{\sigma^*}} = [\Omega_0(\sigma^*), \Omega_1(\sigma^*), \ldots, \Omega_T(\sigma^*)]$ является максимальным на отрезке $[0, T]$. Более того, так как пример I^+ содержит только работы с неотрицательной прибылью, то

$$\Omega_0(\sigma^*) \leq \Omega_1(\sigma^*) \leq \ldots \leq \Omega_T(\sigma^*). \tag{2.18}$$

Пусть $\nu_1 < \nu_2 < \ldots < \nu_{|\Phi^+|+|\Psi^+|-1}$ – левые крайние точки заполненных и свободных интервалов в $\sigma^*(I^+)$. Рассмотрим специальный вектор ресурса

$$\overrightarrow{v^*} = [\Omega_{\nu_1}(\sigma^*), \Omega_{\nu_2}(\sigma^*), \ldots, \Omega_{\nu_{|\Phi^+|+|\Psi^+|-1}}(\sigma^*)].$$

Неравенство (2.18) влечет $\Omega_{\nu_t}(\sigma^*) \geq \Omega_t(\sigma^*)$ для любых $t, 0 \leq t \leq \nu_{|\Phi^+|+|\Psi^+|-1}+1$. Таким образом, в любом расписании суммарный по времени объем свободного ресурса в первых $|\Phi^+| + |\Psi^+| - 1$ интервалах не превышает $\sum\limits_{t=1}^{|\Phi^+|+|\Psi^+|-1} \Omega_{\nu_t}(\sigma^*) = \sum\limits_{I_t \in \Phi^+} \Omega_t + \sum\limits_{i=1}^{|\Psi^+|-1} \Omega_{\tau_i}$.

Рассмотрим зеркальный пример I_d. Пусть $\mu_1 < \mu_2 < \ldots < \mu_{|\Phi^-|}$ – левые крайние точки заполненных интервалов в $\sigma_d^*(I^-)$. Пусть $T_d = C_{max}(\sigma_d^*(I^-))$. По лемме 5 вектор ресурса $\overrightarrow{v_{\sigma_d^*}} = [\Omega_0(\sigma_d^*), \Omega_1(\sigma_d^*), \ldots, \Omega_{T_d}(\sigma_d^*)]$ является максимальным на отрезке $[0, T_d]$. Определим специальный зеркальный вектор ресурса

$$\overrightarrow{v_d^*} = [\Omega_{\mu_1}(\sigma_d^*), \Omega_{\mu_2}(\sigma_d^*), \ldots, \Omega_{\mu_{|\Phi^-|}}(\sigma_d^*)].$$

Тогда в любом расписании суммарный по времени объем свободного ресурса в последних $|\Phi^-|$ интервалах не превышает $\sum\limits_{t=0}^{|\Phi^-|} \Omega_{\mu_t}(\sigma_d^*) = \sum\limits_{I_t \in \Phi^-} \Omega_t$.

Предположим, что существует допустимое расписание σ длины меньше чем $|\Phi^+| + |\Phi^-| + |\Psi^+|$. Следовательно, суммарный по времени объем свободного ресурса в σ не превосходит $\sum\limits_{I_t \in \Phi^+} \Omega_t + \sum\limits_{I_t \in \Phi^-} \Omega_t + \sum\limits_{i=1}^{|\Psi^+|-1} \Omega_{\tau_i}$. Однако, по лемме 8 имеем, что для выполнения всех работ требуется по крайней мере $A > \sum\limits_{I_t \in \Phi^+} \Omega_t + \sum\limits_{I_t \in \Phi^-} \Omega_t + \sum\limits_{i=1}^{|\Psi^+|-1} \Omega_{\tau_i}$ единиц ресурса. Получаем противоречие с допустимостью σ. □

Используя аналогичные рассуждения для зеркального примера, можно получить следующий результат.

Лемма 10 *В любом допустимом расписании* $\sigma(I)$,

$$C_{max}(\sigma) \geq OPT \geq |\Phi^-| + |\Psi^-| + |\Phi^+|.$$

Далее покажем, что слабо допустимое расписание может быть преобразовано в допустимое, и его длина увеличится не более чем в два раза. Назовем работу J_i в слабо допустимом расписании σ дробной, если она выполняется более чем в одном единичном интервале. Округлим все дробные значения назначения x_{it} до 0 или 1.

Алгоритм 2.4

1: Применить Алгоритм 2.3 к примеру I.
2: **for** каждой дробной работы J_i **do**
3: **if** $\delta_i \geq 0$ **then**
4: положить $\tau = \min\{t | x_{it} > 0\}$.
5: **if** $\delta_i < 0$ **then**
6: положить $\tau = \max\{t | x_{it} > 0\}$.
7: Вставить пустой интервал $(\tau, \tau + 1)$.
8: Назначить работу J_i в интервал $(\tau, \tau+1)$ и удалить все ее дробные компоненты, то есть положить $x_{i\tau} = 1$ и $x_{it} = 0$ для всех $t \neq \tau$.
9: Удалить все пустые интервалы.
10: Выдать полученное расписание σ.

Теорема 9 *Алгоритм 2.4 является 2-приближенным алгоритмом для задачи* $W1|p_i = 1|C_{max}$.

Доказательство. Сначала покажем, что расписание σ допустимо. По лемме 5, Алгоритм 2.3 строит слабо допустимое расписание $\bar{\sigma}$. Добавление пустого интервала в $\bar{\sigma}$ не нарушает условий (У1)-(У3) в определении слабо допустимого расписания. Пусть J_i — работа с дробным назначением в $\bar{\sigma}$ такая, что $\delta_i \geq 0$ и $\tau = \min\{t | x_{it} > 0\}$. Перенос любого фрагмента работы J_i в любой из интервалов, расположенных слева, не уменьшает доступный объем ресурса ни в одном из единичных интервалов. Таким образом, осталось проверить, что в новом интервале достаточно ресурса для выполнения работы J_i. Из $x_{i\tau} > 0$ и условия (У1) в определении слабо допустимого расписания имеем $\alpha_i \leq \Omega_\tau$. Следовательно, работа J_i является доступной в момент τ. Применяя аналогичные рассуждения ко всем дробным работам с $\delta_i \geq 0$ и аналогичные рассуждения к зеркальному расписанию для работ с $\delta_i \leq 0$, получим, что расписание σ, построенное алгоритмом 2.4, является допустимым.

Оценим длину расписания σ. Заметим, что число дробных работ не превышает числа заполненных интервалов. Кроме того, все фиктивные интервалы после удаления из них фрагментов дробных работ становятся пустыми и на шаге 3 алгоритма 2.4 удаляются из расписания. Таким образом, длина полученного расписания не превосходит $C_{max}(\sigma) \leq 2|\Phi^+| + 2|\Phi^-| + |\Psi^+| + |\Psi^-|$. Отсюда совместно с результатами лемм 9 и 10 получаем, что $C_{max}(\sigma) \leq 2OPT$. □

Расписание, построенное алгоритмом 2.4, также можно использовать для построения расписания в задачах $W1, P|p_i = 1|C_{max}$ и $W1, Pm|p_i = 1|C_{max}$. Действительно,

пусть в примере I задачи $W1, P|p_i = 1|C_{max}$ требуется выполнить множество работ на m машинах. Если в полученном расписании в каждом интервале выполняется не более m работ, то полученное алгоритмом 2.4 расписание является в нем допустимым и, следовательно, 2-приближенным. В противном случае применим к нему известную процедуру "расщепления" расписания.

Алгоритм 2.5

1: Применить Алгоритм 2.4 к примеру I.
Обозначить полученное расписание через $\bar{\sigma}$. Пусть n_t — число работ, назначенных в интервал $(t, t+1)$ для $t = 0, \ldots, C_{max}(\bar{\sigma}) - 1$.
2: **for** t таких, что $n_t > m$ **do**
3: Положить $\tau := \lceil \frac{n_t}{m} \rceil$
4: Назначить n_t работ в интервалы $[t, t+1), [t+1, t+2), \ldots, [t+\tau-1, t+\tau)$ так, что первые $\tau - 1$ интервалов содержат по m работ.
5: Для каждой работы J_i с $C_i(\sigma) \geq t+1$ положить $C_i(\sigma) = C_i(\sigma) + \tau$.
6: Выдать расписание σ.

Теорема 10 *Алгоритм 2.5 является* $(3 - \frac{2}{m})$*-приближенным алгоритмом для задачи* $W1, P|p_i = 1|C_{max}$.

Доказательство. Пусть σ — расписание, полученное алгоритмом 2.5. Назовем интервал $(t, t+1)$ *полным*, если в нем выполняется ровно m работ, и неполным, если в нем выполняется меньше чем m работ. Пусть h и l — число полных и неполных интервалов в σ соответственно. Тогда $C_{max}(\sigma) = l + h$.

Заметим, что после процедуры "расщепления" единичного интервала образуется не более одного нового неполного интервала. Следовательно, l не превышает длины расписания, полученного алгоритмом 2.4, и $l \leq 2 \times OPT$. С другой стороны, в рассматриваемом примере требуется выполнить по крайней мере $\frac{hm+l}{m}$ единичных работ на m машинах и, следовательно, $\frac{hm+l}{m} \leq OPT$. Комбинируя эти неравенства, получаем

$$\frac{C_{max}(\sigma)}{OPT} \leq \frac{l+h}{\max\{\frac{l}{2}, \frac{hm+l}{m}\}}.$$

Покажем, что эта величина не больше $3 - \frac{2}{m}$.

Действительно, если $l/2 \geq (hm+l)/m$, то

$$\frac{C_{max}(\sigma)}{OPT} \leq \frac{2(l+h)}{l} = 2 + \frac{2h}{l} \leq 2 + \frac{hm}{hm+l} = 3 - \frac{l}{hm+l} \leq 3 - \frac{l}{0.5lm} = 3 - \frac{2}{m}.$$

В противном случае получаем

$$\frac{C_{max}(\sigma)}{OPT} \leq \frac{m(l+h)}{hm+l} = 1 + \frac{lm-l}{hm+l} \leq 1 + \frac{lm-l}{0.5lm} = 3 - \frac{2}{m}.$$

□

Покажем, что оценки, полученные в теоремах 9 и 10, являются точными. Рассмотрим пример, который содержит m^2k работ. Для удобства разобьем эти работы на $k(m-2)+1$ множество так, что первые $k(m-2)$ множеств содержат по $m+2$ работ, и последнее множество содержит $4k$ работ.

1. Каждое из множеств $\mathcal{J}_i$, $i = 1, ..., k(m-2)$, содержит
   (a) две "легкие" работы с $\alpha_{ij} = 1/m^2$ для $i = 1, ..., k(m-2)$, и $j = 1, 2$;
   (b) $m-1$ "нормальную" работу с $\alpha_{ij} = 1/(2(m-1))$ для $i = 1, ..., k(m-2)$ и $j = 3, ..., m+1$;
   (c) одну "тяжелую" работу с $\alpha_{im+2} = 1/2$ для $i = 1, ..., k(m-2)$.
2. Все работы последнего множества $N_{k(m-2)+1}$ одинаковые и $\alpha_{ij} = 1/2 - (m-2)/(2m^2)$ для $i = k(m-2)+1$, и $j = 1, ..., 4k$. Назовем их "специальными" работами.

Пусть $\Omega_0 = 1$ и $\alpha_{ij} = \beta_{ij}$ для всех работ. Поскольку $\delta_{ij} = 0$ для всех работ, то все работы имеют равный приоритет и алгоритм 2.2 может их обслуживать в произвольном порядке. Предположим, что алгоритм 2.2 назначил работы в лексикографическом порядке. Тогда он получил слабо допустимое расписание длины km, в котором все "тяжелые" работы и $2k-1$ "специальная" работа являются дробными. Округление дробных переменных в этом решении алгоритмом 2.4 приведет к допустимому расписанию задачи $W1|p_i = 1|C_{max}$, длина которого равна $2km - 1$. Причем $k(m-2)$ единичных интервала будут содержать по $m+1$ работ, в одном единичном интервале будет две работы, и в остальных интервалах — по одной работе. Применив к полученному решению алгоритм 2.5, получим допустимое расписание задачи $W1, P|p_i = 1|C_{max}$ длины $k(3m-2)-1$.

Осталось заметить, что для данного примера существует оптимальное расписание длины km. Сгруппируем $m-1$ "нормальных" работ с одной "тяжелой" работой. Тогда выполнение всех таких работ потребует $k(m-2)$ единиц времени. Сгруппируем $m-2$ легких работ с двумя "специальными" работами. Выполнение таких работ потребует $2k$ единиц времени. В итоге длина построенного расписания равна km. Выбирая k достаточно большим, получим, что оценки на алгоритмы 2.4 и 2.5, доказанные в теоремах 9 и 10, являются точными.

2.3.2 Приближенные алгоритмы для задач с произвольными длительностями

В заключение рассмотрим задачу $W1||C_{max}$, в которой длительности работ произвольны и неограниченное число работ может выполняться одновременно. Как и в предыдущем параграфе, сначала предположим, что все работы имеют неотрицательную прибыль, т.е. $\delta_i \geq 0$. Отметим, что если в задаче $W1|\delta_i \geq 0|C_{max}$ величины Ω,

α_i и β_i являются целыми числами и $\alpha_i = \beta_i$ для всех i, то получается известная в теории расписаний задача минимизации длины на параллельных идентичных машинах, в которой работы должны быть обслужены более чем одной машиной одновременно.

В этом разделе для задач $W1|\delta_i \geq 0|C_{max}$ и $W1||C_{max}$ представим способ построения $O(\log n)$-приближенного решения.

Рассмотрим произвольный пример I задачи $W1|\delta_i \geq 0|C_{max}$. Пусть пример I' получается из I округлением в большую сторону длительностей работ к ближайшей степени числа 2. Легко показать, что $C^*_{max}(I') \leq 2C^*_{max}(I)$. Действительно, рассмотрим некоторое оптимальное расписание $\sigma^*(I)$ для примера I и построим допустимое расписание $\sigma(I')$ такое, что $C_{max}(\sigma(I')) \leq 2C_{max}(\sigma^*(I))$. Увеличим моменты начала и окончания каждой работы в расписании $\sigma^*(I)$ в два раза. При этом длительность каждой работы в новом расписании также увеличится в два раза. Так как количество доступного ресурса зависит только от моментов начала и окончания работ, полученное расписание останется допустимым. Уменьшим длительность каждой работы так, чтобы она равнялась длительности этой работы в примере I'. Из ограничения $\delta_j \geq 0$ получим, что объем ресурса в каждый момент времени не уменьшится, и, следовательно, полученное расписание останется допустимым. Отсюда имеем $C^*_{max}(I') \leq C_{max}(\sigma(I')) \leq 2C^*_{max}(I)$. Отметим, что в примере I' число различных длительностей работ не превосходит $\lambda = \lceil \log p_{\max} \rceil$.

Далее покажем, что для примера I' существует *гнездовое* расписание σ', в котором каждая работа стартует в целый момент, пропорциональный ее длительности, и $C_{max}(\sigma') \leq 2C^*_{max}(I')$. Рассмотрим некоторое оптимальное расписание $\sigma^*(I')$ и увеличим моменты начала и окончания каждой работы в нем в два раза. Как и в предыдущем случае получим допустимое расписание. Пусть работа J_i начинается в момент τ и заканчивается в момент $\tau + 2p_j$. Пусть l — минимальное целое такое, что $lp_j \geq \tau$. Так как $lp_j \leq \tau + p_j$, то, поместив работу в интервал $[lp_j, (l+1)p_j)$, получим расписание, допустимое относительно ресурсных ограничений и первоначальной длительности работы. Повторив рассуждение для всех остальных работ, получим требуемый результат.

Рассмотрим произвольное гнездовое расписание σ и пусть $\mathcal{J}_i^k$ — множество (возможно пустое) работ, длина которых равна 2^i и они выполняются в интервале $\upsilon_i^k = [2^i(k-1), 2^i k)$ при условии, что $2^i k \leq C_{\max}(\sigma)$. Растянем расписание σ следующим образом. Каждому множеству работ $\mathcal{J}_i^k$ поставим в соответствие интервал длины 2^i, в котором они будут выполняться. Пусть $C(\mathcal{J}_i^k)$ момент завершения работ множества $\mathcal{J}_i^k$ в расписании σ Определим последовательность выполнения работ по следующему правилу. Если $C(\mathcal{J}_{i'}^{k'}) < C(\mathcal{J}_{i''}^{k''})$, сначала выполняются работы из $\mathcal{J}_{i'}^{k'}$, а затем работы из $\mathcal{J}_{i''}^{k''}$. В случае, когда моменты завершения работ двух множеств совпадают, первыми выполняются работы с меньшей длительностью. Обозначим полученное расписание через σ'. Нетрудно проверить, что оно также является допустимым расписанием для примера I'. Пример преобразования расписания σ в расписание σ' изображен на рисунке 2.1.

Оценим длину расписания σ'. Рассмотрим множество работ, выполняющихся в расписании σ внутри интервала $[(l-1)2^\lambda, l2^\lambda)$ для некоторого целого положительно-

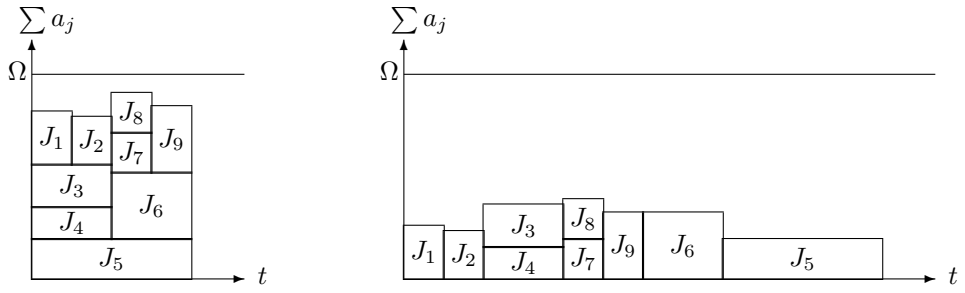

Рис. 2.1: Растяжение гнездового расписания

го l. В расписании σ' эти работы выполняются в последовательных интервалах, общая длина которых не превышает $\lambda 2^\lambda$. Пусть k – наибольшее целое такое, что $k2^\lambda \le C_{\max}(\sigma)$. Тогда

$$C_{\max}(\sigma') \le (k+1)\lambda 2^\lambda \le 2\lambda C_{\max}(\sigma).$$

Обозначим через $\mathcal{J}_i$ множество работ в примере I', длина которых равна 2^i. Будем строить расписание, в котором допускается параллельное выполнение работ, только если они принадлежат одному множеству. Определим последовательность индексов π по следующим рекуррентным соотношениям. Положим $\pi(1) = 0$. Пусть r_i — количество появлений числа i в последовательности π. Тогда

$$\pi(k+1) = \begin{cases} 0, & \text{если } r_{\pi(k)} \text{ — нечетное или } \pi(k) = \lambda, \\ \pi(k)+1, & \text{если } r_{\pi(k)} \text{ — четное и } \pi(k) < \lambda. \end{cases} \tag{2.19}$$

Пусть $\Upsilon_i = [\tau_i, \bar{\tau}_i)$ — интервал длины $2^{\pi(i)}$, точные границы которого определяются алгоритмом, строящим слабое допустимое расписание.

Пусть $0 \le x_{ji} \le 1$ — фрагмент работы $J_j \in \mathcal{J}_{\pi(i)}$, который выполняется в интервале Υ_i. Тогда работа J_j потребляет $\alpha_j x_{ji}$ единиц ресурса в момент τ_i и производит $\beta_j x_{ji}$ единиц ресурса в момент $\bar{\tau}_i$. Если $x_{ji} > 0$, то длина работы J_j не зависит от величины x_{ji} и она выполняется в течение всего интервала Υ_i.

Аналогично доказательству леммы 5 легко по индукции доказать, что величина ресурса Ω при назначении работ в интервал $\Upsilon_i = [\tau_i, \bar{\tau}_i)$ на i-ой итерации цикла **while** алгоритма 2.6 больше чем величина ресурса, доступного в момент τ_i в расписании σ'. Отсюда следует, что длина слабо допустимого расписания, построенного алгоритмом 2.6, не превосходит длины расписания σ'. Округление полученного слабо допустимого решения путем назначения дробной работы в отдельный интервал увеличит длину расписания не больше чем в два раза. Принимая во внимание рассуждения, изложенные в этом параграфе, получим, что для задачи $W1|\delta_i \ge 0|C_{max}$ существует $O(\log p_{\max})$-приближенный алгоритм.

Покажем, что можно преобразовать входные данные примера I задачи $W1|\delta_i \ge 0|C_{max}$, так что $p_{\max} \le n$. Положим $p'_j = \frac{np_j}{p_{max}}$. Получим, что длина максимальной работы в новом примере равна n. Округлим длительности работ вверх до ближайшего целого. Длина любого допустимого расписания увеличится при этом не более, чем на

Алгоритм 2.6

Положить $A := \emptyset$, $i := 0$, $v := \Omega_0$, $\tau_1 := 0$, $\bar{\tau}_1 := 1$ и $\bar{x}_j := 1$ для $j = 1, \ldots, n$.
while есть доступные незавершенные работы **do**
  if $J_j \in \mathcal{J}_i$ & $\alpha_j \leq V$ **then**
    положить $\bar{y}_j := \bar{x}_j$ и $A := A \cup \{j\}$.
  if $A = \emptyset$ **then**
    положить $i := i + 1$, $\tau_i := \bar{\tau}_{i-1}$, и $\bar{\tau}_i := \tau_i + 2^{\pi(i)}$.
  else
    Найти $\langle y \rangle$ решение задачи ДРОБНЫЙ РЮКЗАК(A).
    Положить $x_{ji} := y_j$, $\bar{x}_j := \bar{x}_j - x_{jt}$ для $j = 1, \ldots, n$.
    Положить $v := v + \sum_{j=1}^{n} x_{jt}\delta_j$, $i := i + 1$, $\tau_i := \bar{\tau}_{i-1}$, и $\bar{\tau}_i := \tau_i + 2^{\pi(i)}$.
if $\bar{x}_j = 0$ для всех $j = 1, \ldots, n$ **then**
  Выдать слабо допустимое расписание x_{ji}.
else
  Выдать "нет допустимого расписания".

n. Учитывая, что длина оптимального расписания не меньше n, получим, что длина оптимального расписания в полученном примере не более чем в два раза больше длины оптимального расписания в исходном примере. Применяя описанную выше процедуру для преобразованного примера, получим, что для задачи $W1|\delta_i \geq 0|C_{max}$ существует $O(\log n)$-приближенный алгоритм.

Для того чтобы получить приближенное решение задачи $W1||C_{max}$ с произвольным доходом от работ, разобьем множество работ на два с отрицательным и неотрицательным доходом и решим каждую из задач отдельно. Пусть σ — расписание, полученное методом описанным выше для задачи $W1|\delta_i \geq 0|C_{max}$. Для множества работ с $\delta_j < 0$ перейдем к зеркальной задаче, в которой доход всех работ положителен, а затем возьмем зеркальное расписание $\bar{\sigma}$ к полученному решению. Получим решение задачи $W1||C_{max}$ как объединение полученных расписаний $\sigma \circ \bar{\sigma}$ такое, в котором работы с $\delta_j \geq 0$ предшествуют работам с $\delta_j < 0$. Так как $C_{\max}(\sigma \circ \bar{\sigma}) \leq 2\max\{C_{max}(\sigma), C_{\max}(\bar{\sigma})\}$, то в итоге получим основной результат этого параграфа.

Теорема 11 *Для задачи $W1||C_{max}$ существует $O(\log n)$-приближенный алгоритм.*

Глава 3

Энергетически эффективные расписания

В этой главе рассматриваются задачи построения допустимых расписаний, в которых скорость выполнения работ или их операций зависит от количества энергии, потраченной на их выполнение. Множество работ $\mathcal{J} = \{J_1, \ldots, J_n\}$ должно быть выполнено на одной или нескольких машинах. В однородной модели для каждой работы заданы время ее поступления r_j, директивный срок d_j и ее объем W_j.

Длительность работы зависит от скорости, с которой она выполняется. Выбор скорости работы каждой машины $S(t)$ определяется при составлении расписания и может меняться с течением времени. Обозначим через Int_j объединение интервалов, в которых выполняется работа J_j. Без ограничения общности можно считать, что число интервалов, входящих в Int_j, конечно и среди них нет интервалов нулевой длины. Пусть $\delta(j,t) = 1$, если $t \in Int_j$, и $\delta(j,t) = 0$, в противном случае. Тогда для каждой работы J_j должно быть выполнено условие

$$\int_{r_j}^{d_j} S(t)\delta(j,t)dt = W_j.$$

Пусть машина работает в интервале времени $[t_0, t_1)$. Количество энергии, которое она израсходует, равно

$$E = \int_{t_0}^{t_1} (S(t))^\alpha dt, \tag{3.1}$$

где $\alpha > 1$ – некоторая заданная константа. Для каждой работы требуется определить интервалы, в которых она будет выполняться, и скорость ее выполнения так, чтобы расписание было допустимым относительно времен поступления и директивных сроков и общий расход энергии был минимальным.

Задачи, в которых требуется минимизировать расход энергии, привлекают огромное внимание исследователей на протяжении двух последних десятилетий. Это связано как с бурным ростом промышленности, а, следовательно, и с потреблением энергии в развитых и развивающихся странах, так и с развитием компьютерных технологий, которые позволяют ускорять решение задач за счет дополнительной

энергии. Действительно, многие современные многопроцессорные компьютеры, например, Intel SpeedStep или AMD PowerNow, могут варьировать свою скорость. Высокие скорости приводят к более быстрому и качественному обслуживанию, но при этом и к большим затратам энергии. Низкие скорости позволяют сэкономить энергию, но ухудшают качество обслуживания. Широко известное в инженерных кругах правило кубического корня для устройств, использующих микросхемы CMOS, гласит, что скорость вычислений пропорциональна кубическому корню от потребляемой мощности, то есть значение параметра $\alpha = 3$. В статьях по теории расписаний, как правило, рассматривается произвольное значение $\alpha > 1$. Расход энергии определяется интегрированием функции мощности на интервале времени работы машины.

Исследованиям по оптимизации расхода энергии путем выбора оптимальных скоростей выполнения работ положила начало работа Яо, Демерс и Шенкер 1995 года, представленная на ежегодном симпозиуме по основам компьютерных наук (FOCS 1995) [166]. В этой статье рассматривается задача $1|pmtn, r_j, d_j|E$ минимизации энергии на одной машине в предположении, что каждая работа может быть прервана и продолжена позднее.

Для задачи $1|pmtn, r_j, d_j|E$ в [166] предложен точный полиномиальный алгоритм YDS, основанный на понятии плотности временного интервала. Рассмотрим последовательность моментов $t_0 < t_1 < \cdots < t_k$, соответствующих временам поступления и директивным срокам всех работ. Если несколько времен поступления или директивных сроков соответствуют одному моменту, то он рассматривается один раз. Определим интервал $I_{p,q} = [t_p, t_q]$ для всех $0 \leq p < q \leq k$ и величину $|I_{p,q}| = t_q - t_p$ как длину интервала $I_{p,q}$. Говорят, что работа J_j *задействована* в интервале $I_{p,q}$, если $[r_j, d_j] \subseteq I_{p,q}$. Множество задействованных работ в интервале $I_{p,q}$ обозначим через $A(I_{p,q})$. *Плотностью* $\Delta(I_{p,q})$ интервала $I_{p,q}$ называется минимальная средняя скорость, необходимая для выполнения всех работ из $A(I_{p,q})$ внутри этого интервала, $\Delta(I_{p,q}) = \frac{\sum_{J_j \in A(I_{p,q})} W_j}{|I_{p,q}|}$.

На каждой итерации алгоритма YDS определяется интервал $I_{p,q}$ максимальной плотности. В этом интервале машина со скоростью $\Delta(I_{p,q})$ выполняет все работы из множества $A(\Delta(I_{p,q}))$ в порядке неубывания директивных сроков. После этого работы множества $A(\Delta(I_{p,q}))$ и интервал $\Delta(I_{p,q})$ удаляются из рассмотрения. Для любой не назначенной в расписание работы J_i с $d_i \in (t_p, t_q]$ назначается новый директивный срок на начало этого интервала, $d_i = t_p$. Для всех работ с $r_i \in I_{p,q}$ время их поступления полагается равным $r_i = t_q$. Интервал $I_{p,q}$ полагается занятым, и его длина вычитается из длин оставшихся интервалов, которые его включают. Плотности незанятых интервалов пересчитываются, и описанная выше процедура повторяется до тех пор, пока не останется неназначенных в расписание работ. Ниже приведем формальное описание алгоритма YDS (алгоритм 3.1).

Легко увидеть, что алгоритм YDS имеет полиномиальную трудоемкость. В [135] было показано, что его можно реализовать за время $O(n^2 \log n)$. В работе [166] Яо, Демерс и Шенкер не доказали, что алгоритм YDS строит оптимальное решение. Позднее это было установлено в [49]. Впоследствии изучение задач на минимизацию

Алгоритм 3.1 Алгоритм YDS

1: Положить $\mathcal{J} = \{J_1, \ldots, J_n\}$
2: **while** $\mathcal{J} \neq \emptyset$ **do**
3: Определить интервал $I_{p,q}$ максимальной плотности.
4: Выполнить в интервале $I_{p,q}$ работы из множества $A(I_{p,q})$ со скоростью $\Delta(I_{p,q})$ в порядке неубывания директивных сроков.
5: Положить $\mathcal{J} = \mathcal{J} \setminus A(I_{p,q})$.
6: Удалить интервал $I_{p,q}$ из рассмотрения и пересчитать времена поступления и директивные сроки работ не назначенных в расписание.
7: Выдать полученное расписание.

расхода энергии разделилось на несколько направлений. В частности, большое количество интересных результатов было получено для он-лайн задач. В он-лайн задачах предполагается, что информация об объеме работы и ее директивном сроке становится известна только в момент поступления работы на обслуживание. Обзор различных постановок он-лайн задач и алгоритмов их решения можно найти в [28, 111]. Поскольку он-лайн постановки выходят за рамки диссертации, далее приведем обзор результатов для традиционных постановок, когда вся информация о задаче известна на момент принятия решения.

Как упоминалось ранее, задача $1|pmtn, r_j, d_j|E$ является полиномиально разрешимой. Две различные постановки задач с прерываниями рассматриваются для случая, когда в системе есть несколько параллельных машин. В задаче $P|pmtn*, r_j, d_j|E$ требуется, чтобы работа целиком выполнялась на одной из параллельных машин. В задаче $P|pmtn, r_j, d_j|E$ предполагается, что после прерывания работу можно продолжить на любой другой машине. Для второй задачи в [29, 33, 48, 56] были представлены точные полиномиальные алгоритмы, основанные на различных ее сведениях к задаче о максимальном потоке минимальной стоимости.

В [30] Альберс, Мюллер и Шмельцер доказали, что задача $P|pmtn*, r_j, d_j|E$ является NP-трудной в сильном смысле, даже если все работы имеют равные объемы. Задача с равными объемами $P|pmtn*, W_j = 1, r_j, d_j|E$ становится полиномиально разрешимой, если работы имеют согласованные времена поступления и директивные сроки, т.е. для любых двух работ J_i и J_j из неравенства $r_i \leq r_j$ следует неравенство $d_i \leq d_j$. Если объемы выполнения работ произвольны, то задача $P|pmtn*, r_j, d_j|E$ является NP-трудной, даже когда все работы должны быть выполнены внутри одного интервала времени, т.е. имеют общие время поступления и директивный срок. В той же работе [30] авторы предложили приближенные алгоритмы для различных вариантов задачи $P|pmtn*, r_j, d_j|E$. В частности, они построили $(\alpha^\alpha 2^{4\alpha})$-приближенный алгоритм для случая, когда все работы имеют единичный объем, $2(2-\frac{1}{m})^\alpha$-приближенный алгоритм для случая, когда все работы имеют одинаковые времена поступления и директивные сроки и $(\alpha^\alpha 2^{4\alpha})$-приближенный алгоритм для случая, когда времена поступления работ и их директивные сроки согласованы. Грейнер, Ноннер и Соуза [100] разработали общий метод, позволяющий по опти-

мальному расписанию задачи $P|pmtn, r_j, d_j|E$ получить $B_{\lceil\alpha\rceil}$-приближенное решение задачи $P|pmtn*, r_j, d_j|E$ в случае, если $\alpha \leq m$, где $B_{\lceil\alpha\rceil}$ — $\lceil\alpha\rceil$-е число Белла. *Числом Белла* B_n называется число всех неупорядоченных разбиений n-элементного множества. Для чисел Белла справедлива формула Добинского

$$B_n = \sum_{k=0}^{\infty} \frac{k^n e^{-1}}{k!}.$$

Отсюда видно, что n-е число Белла равно n-у моменту случайной величины с распределением Пуассона с математическим ожиданием 1. Последнее замечание позволяет ввести более общее определение. Для произвольного действительного параметра $\alpha \geq 0$ назовем *обобщенным числом Белла* величину

$$\tilde{B}_\alpha = \sum_{k=0}^{\infty} \frac{k^\alpha e^{-1}}{k!},$$

соответствующую α-му (дробному) моменту случайной величины с распределением Пуассона с математическим ожиданием 1.

Заметим, что задача $P|pmtn*, r_j, d_j|E$ с согласованными временами поступления и директивными сроками эквивалентна задаче $P|r_j, d_j|E$ с согласованными временами поступления и директивными сроками, в которой прерывания запрещены. Действительно, любое расписание с прерываниями для первой задачи можно преобразовать в расписание без прерываний без увеличения расхода энергии, расположив работы в порядке их поступления на обслуживание. Следовательно, результаты работ [30] и [100] для согласованных директивных сроков верны и для задач без прерываний.

В [36] Антониадис и Хванг доказали, что задача $1|r_j, d_j|E$ на одной машине без прерываний NP-трудна, даже если интервалы выполнения всех работ вложены друг в друга, т.е. если для любых двух работ J_i и J_j из неравенства $r_i \leq r_j$ следует неравенство $d_i \geq d_j$. Они также построили $2^{4\alpha-3}$-приближенный алгоритм для этого случая и $2^{5\alpha-4}$-приближенный алгоритм для общего случая задачи $1|r_j, d_j|E$.

В этой главе изучаются два подхода к построению приближенных алгоритмов с гарантированной оценкой точности для различных задач на минимизацию расхода энергии. Первый подход основан на преобразовании оптимальных решений, в которых есть прерывания работ, в допустимые решения без прерываний. Анализ применимости данного метода и построение алгоритмов с гарантированной оценкой точности рассматриваются в разделе 3.1.

Второй подход основан на представлении рассматриваемых задач как задач линейного программирования с большим не полиномиальным числом ограничений. Используя метод эллипсоидов, при выполнении определенных условий для таких задач, можно найти точное дробное решение за полиномиальное время. Округление полученного решения позволяет находить хорошие приближенные решения для неоднородных задач, в которых параметры работ зависят от машин, на которых они выполняются. Приближенные алгоритмы, основанные на втором подходе, рассмотрены в разделе 3.2.

3.1 Минимизация расхода энергии: от расписаний с прерываниями к расписаниям без прерываний

Как упоминалось выше, задачи $1|pmtn, r_j, d_j|E$ и $P|pmtn, r_j, d_j|E$ разрешимы за полиномиальное время. Длина оптимального расписания с прерываниями является нижней оценкой на длину оптимального расписания без прерываний. Одна из стандартных техник построения приближенных алгоритмов для задач теории расписаний, в которых прерывания запрещены, основана на преобразовании оптимального расписания с прерываниями в расписание без прерываний. При этом желательно, чтобы значение целевой функции полученного расписания без прерываний не сильно отличалась от значения целевой функции исходного оптимального расписания с прерываниями. К сожалению, как показывает следующий пример, отношение величины оптимального решения в задаче без прерываний к величине оптимального решения задачи с прерываниями может быть сколь угодно большим даже для случая одной машины.

Пусть на одной машине требуется выполнить $n-1$ маленькую работу единичного объема $J_1, J_2, \ldots, J_{n-1}$ и работу J_n с $W_n = n$. Время поступления и директивный срок работы J_j, $1 \leq j \leq n-1$, равны $r_j = 2j-1$ и $d_j = 2j$ соответственно. Время поступления и директивный срок работы J_n равны $r_n = 0$ и $d_n = 2n-1$ (см. рис. 3.1).

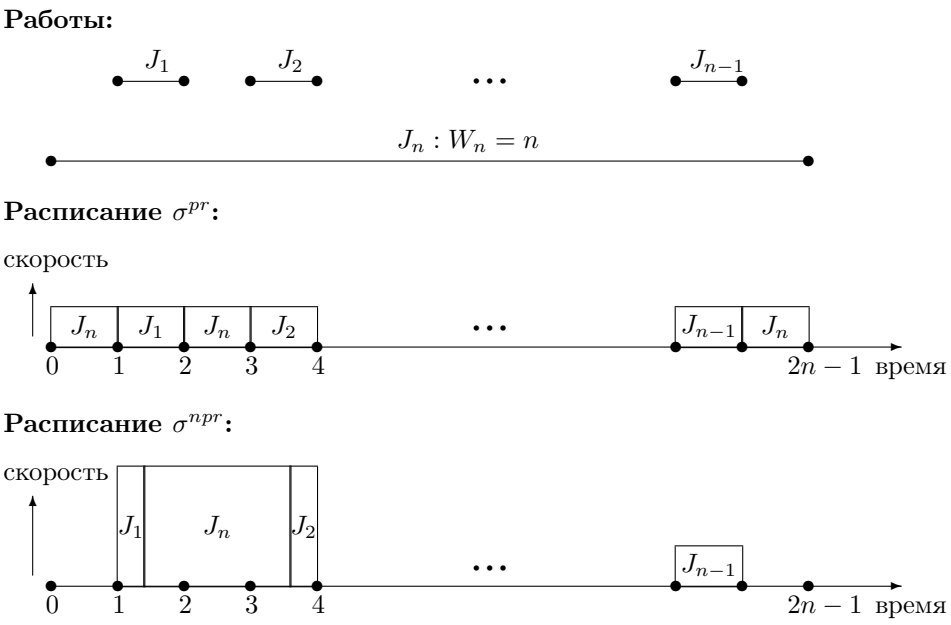

Рис. 3.1: Пример задачи $1|r_j, d_j|E$, для которого отношение оптимума расписания без прерываний к оптимуму расписания с прерываниями равно $\Omega(n^{\alpha-1})$.

Для удобства изложения символом σ^{pr} будем обозначать расписания, в которых разрешено прерывание работ, и символом σ^{npr} — расписания без прерываний.

В оптимальном расписании σ^{pr} для этого примера все работы выполняются со

скоростью 1. При этом каждая работа J_j, $1 \leq j \leq n-1$, выполняется в течение всего времени, отведенного ей на обслуживание. Работа J_n будет выполняться в n промежутках времени единичный длины, когда не обслуживаются другие работы. Общий расход энергии составит $E(\sigma^{pr}) = (n-1) \cdot 1^\alpha + n \cdot 1^\alpha = 2n-1$.

В оптимальном расписании σ^{npr} работа J_n должна выполняться целиком между двумя маленькими работами, например, между работами J_1 и J_2. В этом случае работы J_n, J_1 и J_2 будут выполнены в интервале $[1,4]$ со скоростью $\frac{n+2}{3}$. Все остальные работы будут выполнены в тех же интервалах и с той же скоростью, как и в расписании σ^{pr}. Суммарный расход энергии составит $E(\sigma^{npr}) = 3 \cdot (\frac{n+2}{3})^\alpha + (n-3) \cdot 1^\alpha$. В итоге получим

$$\frac{E(\sigma^{npr})}{E(\sigma^{pr})} = \frac{3 \cdot (\frac{n+2}{3})^\alpha + (n-3) \cdot 1^\alpha}{2n-1} \geq \frac{n^{\alpha-1}}{2 \cdot 3^\alpha} = \Omega(n^{\alpha-1}).$$

Таким образом, в общем случае оптимум расписания с прерываниями не является хорошей нижней оценкой на оптимум расписания без прерываний. Тем не менее при дополнительных ограничениях на задачи $1|r_j, d_j|E$ и $P|r_j, d_j|E$ удается преобразовать оптимальное расписание с прерываниями в допустимое расписание без прерываний и оценить в них погрешность расхода энергии в худшем случае.

Заметим, что из выпуклости функции, стоящей под интегралом в равенстве (3.1), следует, что в любом оптимальном расписании каждая работа J_j должна выполняться с постоянной скоростью S_j. Далее будем рассматривать расписания, в которых каждая работа выполняется с одной скоростью. Обозначим через $E(\sigma, J_j)$ энергию, потраченную на выполнение работы J_j в расписании σ. Тогда $E(\sigma, J_j) = W_j S_j^{\alpha-1}$ и общее количество потраченной энергии на выполнение всех работ в σ равно $E(\sigma) = \sum_{j=1}^{n} E(\sigma, J_j)$.

Следующие два простых, но полезных утверждения были доказаны в [36] и [30] соответственно.

Утверждение 3 [36] *Предположим, что в расписаниях σ и σ' работа J_j выполняется со скоростями S и S' соответственно. Пусть $S \leq \gamma S'$ для некоторого $\gamma \geq 1$. Тогда $E(\sigma, J_j) \leq \gamma^{\alpha-1} E(\sigma', J_j)$.*

Утверждение 4 [30] *Для заданного множества работ $\mathcal{J}$ пусть σ — оптимальное расписание работ на одной машине и σ' — оптимальное расписание работ на m машинах, тогда $E(\sigma) \leq m^{\alpha-1} \cdot E(\sigma')$.*

3.1.1 Одна машина. Свойства оптимальных расписаний на одной машине с прерываниями

Рассмотрим задачу минимизации расхода энергии при выполнении множества работ на одной машине. В следующем параграфе будет рассмотрен приближенный алгоритм, строящий расписание работ без прерываний в процессе обслуживания. Этот алгоритм перестраивает оптимальное расписание, в котором разрешены прерывания

работ, в расписание без прерываний. Для построения оптимального расписания работ на одной машине с прерываниями используется алгоритм YDS, описанный в начале этой главы. В этом параграфе установим свойства расписаний, получаемых этим алгоритмом.

Лемма 11 *Рассмотрим оптимальное расписание σ^{pr}, полученное алгоритмом YDS. Для любых двух работ J_j и $J_{j'}$ в расписании σ^{pr} выполнено*

(i) если $s_j < s_{j'}$, то либо $C_j > C_{j'}$, либо $C_j \leq s_{j'}$, и

(ii) если $s_j < s_{j'}$ и $C_j > C_{j'}$, то $S_j \leq S_{j'}$.

Доказательство. (i) Предположим противное, т.е. что в расписании σ^{pr} существуют две работы J_j и $J_{j'}$ с $s_j < s_{j'}$, $C_j < C_{j'}$ и $C_j > s_{j'}$. Напомним, что на каждой итерации алгоритма YDS выбирается интервал максимальной плотности и все работы, задействованные в этом интервале, назначаются на выполнение.

Пусть работа J_j была назначена в расписание на k-ой итерации и работа $J_{j'}$ — на l-ой итерации. Предположим, что $k < l$. Пусть I_k был выбран как интервал с максимальной плотностью на k-й итерации. Так как $s_j < s_{j'} < C_j$, то существует подинтервал $I' \subseteq [s_{j'}, C_j] \subset [s_j, C_j] \subseteq I_k$, в котором выполняется часть работы $J_{j'}$.

Значит, работа $J_{j'}$ была назначена в расписании либо на k-ой итерации, либо на одной из предшествующих итераций и, следовательно, $k \geq l$. Получили противоречие с тем, что $k < l$. Аналогично можно показать, что l-ая итерация не может предшествовать k-ой итерации.

Предположим, что $k = l$, и, следовательно, обе работы были назначены на выполнение в течение одной итерации. На каждой итерации алгоритм назначает все задействованные работы последовательно в порядке неубывания директивных сроков. Следовательно, интервалы выполнения работ J_j и $J_{j'}$ не пересекаются, что противоречит исходному предположению. Первое утверждение леммы доказано.

(ii) Предположим противное, т.е. пусть в расписании σ^{pr} существуют две работы J_j и $J_{j'}$ с $s_j < s_{j'}$, $C_j > C_{j'}$ и $S_j > S_{j'}$. Так как $S_j > S_{j'}$, работа J_j должна быть назначена на более ранней итерации, чем работа $J_{j'}$. Пусть в момент назначения работы J_j интервал $I_{p,q}$ имел максимальную плотность. По предположению имеем $[s_{j'}, C_{j'}] \subset [s_j, C_j] \subseteq I_{p,q}$, и, следовательно, работа $J_{j'}$ должна быть назначена в расписание на той же итерации, что и J_j; противоречие. □

Из леммы 11 следует, что граф интервалов выполнения работ, порождаемый расписанием σ^{pr}, имеет гнездовую структуру. Используя это свойство, построим древесное представление T расписания σ^{pr}. Каждой работе J_j поставим в однозначное соответствие вершину в ориентированном дереве $T = (V, E)$. Для каждой пары работ J_j и $J_{j'}$ с $[s_{j'}, C_{j'}] \subset [s_j, C_j]$ создадим дугу $(J_j, J_{j'})$ тогда и только тогда, когда не существует третьей работы $J_{j''}$ с $[s_{j'}, C_{j'}] \subset [s_{j''}, C_{j''}] \subset [s_j, C_j]$. Заметим, что получившийся ориентированный граф однозначно определяется расписанием σ^{pr} и является

ориентированным лесом. Более того, из второго утверждения леммы 11 следует, что для каждой дуги $(J_j, J_{j'}) \in E$ в расписании σ^{pr} выполнено $S_j \leq S_{j'}$.

Обозначим через $T(J_j)$ максимальное ориентированное поддерево графа T с корнем в вершине $J_j \in V$. Работу, которой соответствует отличная от корня вершина в графе $T(J_j)$, назовем *потомком* работы J_j в T. Работу J_i такую, что в графе T есть дуга (J_j, J_i), назовем *прямым потомком* работы J_j в T, и через n_j обозначим число прямых потомков J_j в T.

Лемма 12 *Рассмотрим оптимальное расписание σ^{pr}, полученное алгоритмом YDS и соответствующее ему древесное представление $T = (V, E)$. Число прерываний работы J_j, $j = 1, \ldots, n$, в расписании σ^{pr} не превосходит n_j.*

Доказательство. Занумеруем индексами от 1 до n_j прямых потомков работы J_j в порядке их появления в расписании σ^{pr}. Пусть a_i и b_i обозначают момент начала и момент завершения i-го потомка работы J_j в расписании σ^{pr}. Согласно первому утверждению леммы 11 интервалы выполнения этих работ не пересекаются. Для удобства моменты начала и завершения работы J_j в расписании σ^{pr} обозначим через b_0 и a_{n_j+1} соответственно. Тогда $b_0 < a_1 < b_1 \leq a_2 < b_2 \leq \cdots \leq a_{n_j} < b_{n_j} \leq a_{n_j+1}$. Докажем два вспомогательных утверждения.

(a) Работа J_j целиком выполняется в интервалах $[b_{i-1}, a_i)$ ненулевой длины, $i = 1, \ldots, n_j$.

(b) Никакая другая работа не выполняется в интервалах $[b_{i-1}, a_i)$ ненулевой длины, $i = 1, \ldots, n_j$.

Согласно построению ориентированного графа T работа J_i является потомком работы J_j в графе T тогда и только тогда, когда $[s_i, C_i] \subset [s_j, C_j]$. Тогда из первого утверждения леммы 11 следует, что только потомки работы J_j и она сама могут выполняться в интервале $[b_0, a_{n_j+1})$. Заметим, что работа J_j не может быть потомком своего потомка. Следовательно, если работа J_i является потомком работы J_j, то работа J_j не может выполняться внутри интервала $[s_i, C_i]$ в расписании σ^{pr}. Последнее означает, что работа J_j не может выполняться внутри интервала $[a_i, b_i)$, $i = 1, \ldots, n_j$.

Осталось показать, что все потомки работы J_j выполняются внутри одного из интервалов $[a_i, b_i)$, $i = 1, \ldots, n_j$. Утверждение очевидно для прямых потомков работы J_j. Пусть работа J_k является не прямым потомком работы J_j. Тогда она является потомком одного из прямых потомков работы J_j. Пусть это i-й прямой потомок работы J_j. Тогда по построению ориентированного графа T работа J_k целиком выполняется внутри интервала $[a_i, b_i)$.

Утверждения (a) и (b) влекут утверждение леммы. □

3.1.2 Одна машина. Приближенный алгоритм

В этом параграфе приведем приближенный алгоритм 3.2, оценка точности которого зависит от отношения максимального объема работы $W_{\max}$ к минимальному $W_{\min}$. Если объемы работ одинаковы, алгоритм 3.2 находит 2^{α}-приближенное решение в худшем случае. Основная идея нового алгоритма состоит в том, чтобы преобразовать оптимальное расписание σ^{pr}, полученное алгоритмом YDS, в допустимое расписание без прерываний. При перестроении расписания каждой работы будет учитываться количество ее прямых потомков в ориентированном графе T. Обозначим через V_0 множество работ, не имеющих потомков, через V_1 — множество вершин с одним прямым потомком и через V_2 — множество вершин с более чем одним прямым потомком в графе T. Множества V_0, V_1 и V_2 будут изменяться в ходе работы алгоритма.

Алгоритм 3.2

1: Найти оптимальное расписание σ^{pr}, используя алгоритм YDS.
2: По расписанию σ^{pr} построить ориентированный граф $T = (V, E)$ и определить множества V_0, V_1 и V_2.
3: Перестроить расписание σ^{pr} в расписание σ^{npr} следующим образом:
4: **for** каждой работы $J_j \in V_1$ **do**
5: Назначить работу целиком в интервал наибольшей длины, в котором она выполнялась в расписании σ^{pr}. Скорость работы машины в этом интервале положить равной отношению объема работы на длину интервала.
6: **while** $V_2 \neq \emptyset$ **do**
7: Выбрать работу $J_j \in V_2$, у которой нет прямых потомков в V_2.
8: Выбрать произвольную работу $J_{j'} \in T(J_j) \bigcap V_0$.
9: Назначить работы J_j и $J_{j'}$ в интервал, в котором выполнялась работа $J_{j'}$ в расписании σ^{pr}. Скорость работы машины в этом интервале положить равной отношению суммы объемов работ к длине интервала.
10: Удалить работы J_j и $J_{j'}$ из множеств V_2 и V_0 соответственно.
11: **for** каждой работы $J_j \in V_0$ **do**
12: Выполнить работу J_j в том же интервале и с той же скоростью, в котором она выполнялись в расписании σ^{pr}.
13: Выдать расписание σ^{npr}.

Теорема 12 *Алгоритм 3.2 является* $(1 + \frac{W_{\max}}{W_{\min}})^{\alpha}$*-приближенным алгоритмом для задачи* $1|r_j, d_j|E$.

Доказательство. Сначала рассмотрим работы, у которых был ровно один потомок в T. По лемме 12 каждая такая работа J_j выполнялась не более чем в двух интервалах в σ^{pr}. В расписании σ^{npr} работа J_j целиком выполняется в наибольшем из этих двух интервалов. Таким образом, скорость ее выполнения возрастет не более чем вдвое. Следовательно, для любой работы J_j с $n_j = 1$ согласно утверждению 3 получим $E(\sigma^{npr}, J_j) \leq 2^{\alpha-1} \cdot E(\sigma^{pr}, J_j)$.

Пусть работа J_j имеет больше одного потомка в дереве T. В любом поддереве число внутренних вершин меньше, чем число листьев. Следовательно, в графе $T(J_j)$ всегда существует работа $J_{j'} \in T(J_j) \bigcap V_0$, которая не была установлена в расписание σ^{npr} на предыдущей итерации.

Пусть в расписании σ^{pr} работа $J_{j'}$ выполняется в интервале I. Тогда скорость ее обслуживания в σ^{pr} равна $S_{j'} = \frac{W_{j'}}{|I|}$, и затраты энергии на ее обработку составляют $E(\sigma^{pr}, J_{j'}) = w_{j'} S_{j'}^{\alpha-1}$. В расписании σ^{npr} обе работы J_j и $J_{j'}$ выполняются в интервале I со скоростью $S = \frac{W_j + W_{j'}}{|I|}$. Затраты энергии на выполнение работ J_j и $J_{j'}$ в σ^{npr} составляют

$$E(\sigma^{npr}, J_j) + E(\sigma^{npr}, J_{j'}) = (W_j + W_{j'}) S^{\alpha-1} = (W_j + W_{j'}) \left(\frac{W_j + W_{j'}}{|I|} \right)^{\alpha-1}$$

$$= (W_j + W_{j'})^{\alpha} \left(\frac{S_{j'}}{w_{j'}} \right)^{\alpha-1} = \left(\frac{W_j + W_{j'}}{W_{j'}} \right)^{\alpha} \cdot w_{j'} S_{j'}^{\alpha-1} = \left(\frac{W_j + W_{j'}}{W_{j'}} \right)^{\alpha} \cdot E(\sigma^{pr}, J_{j'})$$

$$< \left(\frac{W_{\max} + W_{\min}}{W_{\min}} \right)^{\alpha} \cdot (E(\sigma^{pr}, J_j) + E(\sigma^{pr}, J_{j'})).$$

Осталось заметить, что $I \subseteq [r_j, d_j]$ и, следовательно, σ_{npr} является допустимым расписанием.

Для каждой работы $J_j \in V_0$, которая выполняется одинаково в обоих расписаниях, очевидно, выполнено $E(\sigma^{npr}, J_j) = E(\sigma^{pr}, J_j)$. Суммируя затраты энергии по всем работам для расписаний σ^{pr} и σ^{npr}, получаем утверждение теоремы. □

Для задачи $1|r_j, d_j, W_j = 1|E$, в которой все работы имеют одинаковые объемы, получаем

Следствие 3 *Алгоритм 3.2 является 2^{α}-приближенным алгоритмом для задачи $1|r_j, d_j, W_j = 1|E$.*

В следующих двух параграфах рассмотрим задачи построения энергетически эффективных расписаний работ на параллельных машинах. Сначала опишем приближенный алгоритм с гарантированной константной оценкой точности для задач, в которых согласованы времена поступления работ и директивные сроки. Затем рассмотрим общий случай и построим для него алгоритм с оценкой точности, зависящей от числа работ и числа машин.

3.1.3 Параллельные машины. Согласованные времена поступления и директивные сроки

Известно, что задача $P|r_j, d_j|E$ — NP-трудна, даже если работы имеют согласованные времена поступления и директивные сроки [30]. В [100] для задачи $P|pmtn^*, r_j, d_j|E$ предложен $B_{\lceil\alpha\rceil}$-приближенный алгоритм. Когда времена поступления и директивные сроки согласованы, условие, что работа должна выполняться на одной машине,

эквивалентно условию, что работа выполняется без прерываний. Таким образом, упомянутый выше алгоритм также является и $B_{\lceil\alpha\rceil}$-приближенным алгоритмом для задачи $P|r_j, d_j, \Uparrow|E$.

В этом параграфе приведем новый приближенный алгоритм для задачи $P|r_j, d_j, \Uparrow|E$, который имеет лучшую оценку точности и меньшую трудоемкость по сравнению с алгоритмом из [100]. Сначала, используя оптимальное расписание с прерываниями, определим скорость выполнения каждой работы и ее длительность. Затем построим раннее расписание без прерываний, упорядочив работы согласно их временам поступления и директивным срокам. Заметим, что оптимальное расписание с прерываниями является допустимым относительно директивных сроков, т.е. $r_j \le s_j(\sigma_{pr}) < C_j(\sigma_{pr}) \le d_j$ для любой работы $J_j \in \mathcal{J}$.

Алгоритм 3.3

1: Положить $\tau_i := 0$ для всех $i = 1, \ldots, m$.
2: Упорядочить работы по невозрастанию времен поступления.
3: Построить оптимальное расписание с прерываниями σ_{pr}. Пусть e_j — суммарное время выполнения работы $J_j \in \mathcal{J}$ в σ_{pr}.
4: Положить $p_j = e_j/(2 - \frac{1}{m})$ для всех $j = 1, \ldots, n$.
5: **for** $j := 1, \ldots, n$ **do**
6: Выбрать k такое, что $\tau_k = \min_{i=1,\ldots,m} \tau_i$.
7: Положить $s_j(\sigma_{npr}) = \min\{\tau_k, r_j\}$.
8: Положить $\tau_k = s_j(\sigma_{npr}) + p_j$.
9: Выдать расписание σ_{npr};

Теорема 13 *Алгоритм 3.3 является* $(2 - \frac{1}{m})^{\alpha-1}$*-приближенным алгоритмом для задачи* $P|r_j, d_j, \Uparrow|E$.

Доказательство. Пусть, как и в алгоритме 3.3, работы упорядочены по невозрастанию времен поступления и в силу согласованности — по невозрастанию директивных сроков. Обозначим через $C_j(\sigma_{npr}) = s_j(\sigma^{npr}) + p_j$ момент завершения работы J_j в расписании σ^{npr} и покажем, что алгоритм строит расписание, допустимое относительно директивных сроков, т.е. $C_j(\sigma^{npr}) \le d_j$.

Пусть $r_{\min} = \min_{J_j \in \mathcal{J}} r_j$ и $d_{\max} = \max_{J_j \in \mathcal{J}} d_j$. Тогда все работы выполняются в интервале $[r_{\min}, d_{\max}.]$ Разобьем этот интервал на полные и неполные максимальные интервалы. Интервал времени, в котором в расписании σ^{npr} простаивает по крайней мере одна машина, называется *неполным*. Интервал времени, в течение которого в расписании σ^{npr} на каждой машине выполняется некоторая работа, называется *полным*.

Пусть число неполных интервалов равно ℓ, и пусть $[\tau_i, t_i]$, $1 \le i \le \ell$, — i-й неполный интервал. Тогда $[t_{i-1}, \tau_i]$, $1 \le i \le \ell + 1$ — полный интервал. Для удобства положим $t_0 = 0$ и $\tau_{\ell+1} = \max_{J_j \in \mathcal{J}} C_j(\sigma_{npr})$. Заметим, что расписание может начинаться с неполного интервала, т.е. $t_0 = \tau_1$, или заканчиваться неполным интервалом, т.е. $t_\ell = \tau_{\ell+1}$.

Предположим, что работа $J_j \in \mathcal{J}$ поступила на выполнение в течение неполного интервала $[\tau_i, t_i]$. Тогда, согласно шагу 7 алгоритма 3.3, работа J_j начинает выполняться в момент поступления, т.е. $s_j(\sigma^{npr}) = r_j$. Так как $p_j \leq e_j$ и в расписании σ^{pr} работа J_j завершилась до ее директивного срока, то $C_j(\sigma^{npr}) \leq d_j$.

Теперь рассмотрим работу $J_j \in \mathcal{J}$, поступившую на выполнение в течение полного интервала $[t_i, \tau_{i+1}]$. Обозначим через $\mathcal{J}_i = \{J_j \in \mathcal{J} : r_j < t_i\}$ множество работ, поступивших на выполнение до момента t_i. Пусть $P_{npr,i}(t)$ — суммарное количество времени, в течение которого работы из $\mathcal{J}_i$ выполняются после момента $t \geq t_i$ в расписании σ^{npr}, и $E_{pr,i}(t)$ — суммарное количество времени, в течение которого работы из $\mathcal{J}_i$ выполняются после момента $t \geq t_i$ в расписании σ^{pr}. В случае, когда $t = t_i$, будем использовать сокращенные обозначения $P_{npr,i}$ и $E_{pr,i}$ соответственно. Докажем следующее утверждение.

Лемма 13 $P_{npr,i}(t) \leq \frac{E_{pr,i}(t)}{(2-\frac{1}{m})}$ *для всех* $0 \leq i \leq \ell$.

Доказательство леммы. Докажем утверждение индукцией по i.

Для $i = 0$ имеем $P_{npr,0} = E_{pr,0} = 0$. Предположим, что утверждение выполнено для некоторого i. Покажем, что $P_{npr,i+1}(t) \leq \frac{E_{pr,i+1}(t)}{(2-\frac{1}{m})}$.

Напомним, что $\mathcal{J}_{i+1}$ множество работ с $r_j < t_{i+1}$ в σ^{npr}. Разобьем работы из множества $\mathcal{J}_{i+1}$, которые заканчиваются после момента t, на три множества: множество A с $s_j(\sigma^{npr}) < t_i$, множество B с $t_i \leq s_j(\sigma^{npr}) < \tau_{i+1}$ и множество C с $\tau_{i+1} \leq s_j(\sigma^{npr}) < t_{i+1}$. Рассмотрим два случая.

Случай 1: $\tau_{i+1} \geq \frac{(1-\frac{1}{m})t+t_i}{(2-\frac{1}{m})}$.

Пусть $P_A \leq P_{npr,i}$ — суммарное время обслуживания работ из A после момента t_i. Так как в расписании σ^{npr} работы выполняются без прерываний, то каждая работа $J_j \in A$ обрабатывается $t - t_i$ единиц времени в интервале $[t_i, t]$. С учетом этого получаем

$$P_{npr,i+1}(t) = (P_A - |A|(t - t_i)) + \sum_{J_j \in B \bigcup C} (s_j(\sigma^{npr}) + p_j - t). \tag{3.2}$$

Для расписания σ^{pr} имеем

$$E_{pr,i+1}(t) \geq E_{pr,i} + \sum_{J_j \in \mathcal{J}_{i+1} \setminus \mathcal{J}_i} e_j - m \cdot (t - t_i)$$

$$= (2 - \frac{1}{m})(P_{npr,i} + \sum_{J_j \in \mathcal{J}_{i+1} \setminus \mathcal{J}_i} p_j) - m \cdot (t - t_i). \tag{3.3}$$

Заметим, что количество времени в выражении $P_{npr,i} + \sum_{J_j \in \mathcal{J}_{i+1} \setminus \mathcal{J}_i} p_j$ равно суммарному количеству времени обработки работ из $\mathcal{J}_{i+1}$ после момента t_i. По определению суммарное время обслуживания работ из A после момента t_i равно P_A. Напомним, что эти работы начинают выполнение до момента t_i, а заканчивают после момента t и, следовательно, занимают $|A|$ машин в течение всего интервала $[t_i, t]$. Рассмотрим множество работ, которые поступили на выполнение до момента τ_{i+1}, завершились после момента t_i и не лежат в множестве A. Эти работы вносят в выражение

$P_{npr,i}+\sum_{J_j\in\mathcal{J}_{i+1}\setminus\mathcal{J}_i} p_j$ по крайней мере $(m-|A|-|B|)(\tau_{i+1}-t_i)+\sum_{J_j\in B}(s_j(\sigma^{npr})+p_j-t_i)$ единиц времени, так как машины не простаивают в интервале $[t_i,\tau_{i+1}]$. Наконец, вклад работ из C в выражение $P_{npr,i}+\sum_{J_j\in\mathcal{J}_{i+1}\setminus\mathcal{J}_i} p_j$ равен $\sum_{J_j\in C} p_j$. Подставляя суммарный вклад всех работ из $\mathcal{J}_{i+1}$ в (3.3), получим

$$E_{pr,i+1}(t) \geq (2-\frac{1}{m})\Bigg(P_A+(m-|A|-|B|)(\tau_{i+1}-t_i)$$

$$+\sum_{J_j\in B}(s_j(\sigma^{npr})+p_j-t_i)+\sum_{J_j\in C}p_j\Bigg)-m\cdot(t-t_i).$$

Поделив последнее выражение на $(2-\frac{1}{m})$ и вычтя из него (3.2), получим

$$\frac{E_{pr,i+1}(t)}{(2-\frac{1}{m})}-P_{npr,i+1}(t)\geq(m-|A|-|B|)(\tau_{i+1}-t_i)-\sum_{J_j\in B}t_i-$$

$$\frac{m}{2-\frac{1}{m}}(t-t_i)+|A|(t-t_i)+\sum_{J_j\in B}t-\sum_{J_j\in C}(s_j(\sigma^{npr})-t)$$

$$=\tau_{i+1}(m-|A|-|B|)-m\Bigg(t_i+\frac{t-t_i}{2-\frac{1}{m}}\Bigg)+(|A|+|B|)t+\sum_{J_j\in C}(t-s_j(\sigma^{npr}))$$

$$\geq\Bigg(\frac{(1-\frac{1}{m})t+t_i}{2-\frac{1}{m}}\Bigg)(m-|A|-|B|)-m\Bigg(t_i+\frac{t-t_i}{2-\frac{1}{m}}\Bigg)+(|A|+|B|)t.$$

Последнее неравенство следует из предположения, что $\tau_{i+1}\geq\frac{(1-\frac{1}{m})t+t_i}{(2-\frac{1}{m})}$, и условия, что $t\geq s_j(\sigma^{npr})$ для каждой работы из множества C. Также предположим, что $|A|+|B|\geq 1$. В противном случае $P_{npr,i+1}=0$ и утверждение автоматически выполнено. Следовательно,

$$\frac{E_{pr,i+1}(t)}{(2-\frac{1}{m})}-P_{npr,i+1}(t)\geq m\Bigg(\frac{(1-\frac{1}{m})t+t_i}{2-\frac{1}{m}}-t_i-\frac{t-t_i}{2-\frac{1}{m}}\Bigg)$$

$$+(|A|+|B|)\Bigg(t-\frac{(1-\frac{1}{m})t+t_i}{2-\frac{1}{m}}\Bigg)\geq\frac{t_i-t}{2-\frac{1}{m}}+\frac{t-t_i}{2-\frac{1}{m}}\geq 0.$$

Случай 2: $\tau_{i+1}<\frac{(1-\frac{1}{m})t+t_i}{(2-\frac{1}{m})}$.

Рассмотрим работу $J_j\in A\cup B$. Для нее выполнено $s_j(\sigma^{npr})\leq\tau_{i+1}$ и $C_j(\sigma^{npr})>t$. Обозначим через $\delta_{npr,j}(t)=s_j(\sigma^{npr})+p_j-t$ время обработки работы J_j после момента t в расписании σ^{npr}. Заметим, что $\delta_{npr,j}(t)>0$ для любой работы $J_j\in A\cup B$.

Сначала покажем, что $e_j>t-t_i$. Предположим противное, пусть $e_j\leq t-t_i$. Тогда

$$\begin{aligned}\delta_{npr,j}(t) &= s_j(\sigma^{npr})+p_j-t=s_j(\sigma^{npr})+\frac{e_j}{(2-\frac{1}{m})}-t\\ &\leq \tau_{i+1}+\frac{e_j}{(2-\frac{1}{m})}-t<\frac{(1-\frac{1}{m})t+t_i}{(2-\frac{1}{m})}+\frac{t-t_i}{(2-\frac{1}{m})}-t=0.\end{aligned}$$

Получили противоречие с условием $\delta_{npr,j}(t) > 0$. Следовательно, $e_j > t - t_i$ для любой работы $J_j \in A \cup B$.

Все работы из A начинают выполняться до момента t_i и заканчиваются после момента t. Следовательно, по индукции имеем

$$\sum_{J_j \in A} \delta_{npr,j}(t) = P_{npr,i}(t) \leq \frac{E_{pr,i}(t)}{(2 - \frac{1}{m})}.$$

Рассмотрим работу $J_j \in B$. Обозначим через $\delta_{pr,j}(t)$ суммарное время выполнения работы J_j после момента t в расписании σ^{pr}. Имеем $\delta_{pr,j}(t) \geq e_j + t_i - t$. С учетом неравенств $s_j(\sigma^{npr}) \leq \tau_{i+1}$ и $\tau_{i+1} < \frac{(1-\frac{1}{m})t + t_i}{(2-\frac{1}{m})}$ получаем

$$\begin{aligned}
\frac{\delta_{pr,j}(t)}{(2 - \frac{1}{m})} - \delta_{npr,j}(t) &\geq \frac{e_j + t_i - t}{(2 - \frac{1}{m})} - (s_j(\sigma^{npr}) + p_j - t) \\
&= p_j + \frac{t_i - t}{(2 - \frac{1}{m})} - s_j(\sigma^{npr}) - p_j + t \\
&\geq \frac{t_i - t}{(2 - \frac{1}{m})} - \tau_{i+1} + t \\
&= \frac{(1 - \frac{1}{m})t + t_i}{(2 - \frac{1}{m})} - \tau_{i+1} > 0.
\end{aligned}$$

Рассмотрим работу $J_j \in C$. По определению множества C работа J_j начинает выполняться в неполном интервале $[\tau_{i+1}, t_{i+1}]$. Следовательно, $s_j(\sigma^{npr}) = r_j$. Отсюда получим $\delta_{npr,j}(t) = \max\{0, p_j - (t - r_j)\} = \max\{0, \frac{e_j}{2-1/m} - (t - r_j)\} \leq \frac{\max\{0, e_j - (t - r_j)\}}{2-1/m} \leq \frac{\delta_{pr,j}(t)}{2-1/m}$.

Суммируя по всем работам в $A \cup B \cup C$, получаем $P_{npr,i+1}(t) \leq \frac{E_{pr,i+1}(t)}{(2-\frac{1}{m})}$. □

Вернемся к доказательству теоремы. Пусть работа $J_j \in \mathcal{J}$ поступила на выполнение в течение интервала $[t_i, \tau_{i+1}]$. Так как интервал $[t_i, \tau_{i+1}]$ — полный, то существует работа, которая поступила на выполнение в момент t_i. Если таких работ несколько, выберем среди них работу с наименьшим индексом. Пусть это работа J_q, т.е. $r_q = t_i$. Для работы J_j получим

$$C_j(\sigma^{npr}) \leq t_i + \frac{P_{npr,i} + \sum_{k=q}^{j-1} p_k}{m} + p_j \leq t_i + \frac{\frac{E_{pr,i} + \sum_{k=q}^{j-1} e_k}{m} + e_j}{(2 - \frac{1}{m})}.$$

Так как σ^{pr} является допустимым расписанием, а времена поступления и директивные сроки работ согласованы, то работы $J_q, \ldots, J_j$ выполняются внутри интервала $[t_i, d_j]$ в σ^{pr}. Кроме того, из леммы 13 следует, что по крайней мере $E_{pr,i}$ единиц времени работ из $\mathcal{J}_i$ также выполняются в этом интервале. Следовательно, получаем $E_{pr,i} + \sum_{k=q}^{j} e_k \leq m(d_j - t_i)$ и $e_j \leq d_j - t_i$. Таким образом, $C_j(\sigma^{npr}) \leq t_i + (2 - \frac{1}{m})\frac{d_j - t_i}{(2-\frac{1}{m})} = d_j$ и расписание, полученное алгоритмом 3.3, является допустимым.

Осталось оценить точность алгоритма. При уменьшении времени обслуживания в $(2 - \frac{1}{m})$ раз скорость обработки увеличивается в то же число раз. Так как расход энергии в оптимальном расписании задачи $P|pmtn, r_j, d_j, \Uparrow | E$ является нижней

оценкой на величину оптимального решения в задаче $P|r_j, d_j, \uparrow\uparrow\ |E$, то, используя Утверждение 3, получим

$$E(\sigma_{npr}) \leq (2 - \frac{1}{m})^{\alpha-1} E(\sigma_{pr}) \leq (2 - \frac{1}{m})^{\alpha-1} OPT.$$

□

3.1.4 Параллельные машины. Произвольные времена поступления и директивные сроки

В этом параграфе приведем приближенный алгоритм для задачи $P|r_j, d_j|E$. Используя алгоритм YDS, построим расписание σ^{pr}, которое является оптимальным в задаче $1|pmtn, r_j, d_j|E$. Выберем работы, которые имеют в σ^{pr} не больше чем $n^{\frac{1}{m}}$ прерываний, и построим для них расписание без прерываний на одной машине, назначая каждую работу в максимальный по продолжительности интервал, в котором она выполнялась. Для оставшихся работ снова построим оптимальное расписание в задаче $1|pmtn, r_j, d_j|E$ и повторим описанную процедуру. Не более чем через m итераций все работы будут установлены в расписание.

Алгоритм 3.4

1: Положить $i = 1$; $\mathcal{J}_i = \mathcal{J}$;
2: **repeat**
3: На множестве работ $\mathcal{J}_i$ найти расписание σ_i^{pr}, используя алгоритм YDS.
4: По расписанию σ_i^{pr} построить его древесное представление $T = (V, E)$.
5: Определить $\mathcal{J}_{i+1}$ как множество работ, у которых по крайней мере $n^{\frac{1}{m}}$ прямых потомков в T.
6: Для каждой работы $J_j \in \mathcal{J}_i \setminus \mathcal{J}_{i+1}$ назначить ее в расписание σ^{npr} на машину i в наибольший интервал, в котором она выполнялась в σ_i^{pr}.
7: Положить $i = i + 1$;
8: **until** $\mathcal{J}_i \neq \emptyset$
9: Выдать расписание σ^{npr}.

Теорема 14 *Алгоритм 3.4 является* $m^{\alpha}(\sqrt[m]{n})^{\alpha-1}$*-приближенным алгоритмом для задачи* $P|r_j, d_j|E$.

Доказательство. Обозначим через n_i число работ в множестве $\mathcal{J}_i$ и покажем, что на итерации i в дереве T не более $n_i^{1-\frac{1}{m}}$ вершин имеют по крайней мере $n^{\frac{1}{m}}$ прямых потомков. Предположим, что это не так. Тогда в графе T на итерации i должно быть по крайней мере $n_i^{1-\frac{1}{m}} \cdot n^{\frac{1}{m}} + 1$ вершин, каждая из которых соответствует отдельной работе в множестве $\mathcal{J}_i$. Получаем противоречие с исходным числом работ.

Предположим, что алгоритм 3.4 сделал k итераций. По предыдущему наблюдению имеем, что $k \leq m$.

Рассмотрим i-ю итерацию. Каждая работа в $J_j \in \mathcal{J}_i \setminus \mathcal{J}_{i+1}$ имеет меньше чем $n^{\frac{1}{m}}$ прямых потомков в T, и, следовательно, по лемме 12 она выполняется не более чем в $n^{\frac{1}{m}}$ интервалах в σ_i^{pr}. В расписании σ^{npr} работа J_j выполняется в наибольшем из этих интервалов. Таким образом, скорость выполнения работы J_j в σ^{npr} не более чем в $n^{\frac{1}{m}}$ раз выше скорости ее выполнения в σ_i^{pr}. Следовательно, из утверждения 3 для любой работы J_j получим $E(\sigma^{npr}, J_j) \leq (\sqrt[m]{n})^{\alpha-1} \cdot E(\sigma_i^{pr}, J_j)$. Отсюда

$$E(\sigma^{npr}) = \sum_{i=1}^{k} \sum_{J_j \in \mathcal{J}_i \setminus \mathcal{J}_{i+1}} E(\sigma^{npr}, J_j) \leq (\sqrt[m]{n})^{\alpha-1} \cdot \sum_{i=1}^{k} \sum_{J_j \in \mathcal{J}_i \setminus \mathcal{J}_{i+1}} E(\sigma_i^{pr}, J_j). \tag{3.4}$$

Заметим, что расписание $\mathcal{S}_{pr,i}$ является оптимальным решением задачи $1|pmtn, r_j, d_j|E$ для множества работ $\mathcal{J}_i$, а расписание $\mathcal{S}_{pr,1}$ является оптимальным решением задачи $1|pmtn, r_j, d_j|E$ для множества работ $\mathcal{J}_1$. Из соотношения $\mathcal{J}_i \subset \mathcal{J}_1 = \mathcal{J}$ получаем

$$\sum_{J_j \in \mathcal{J}_i \setminus \mathcal{J}_{i+1}} E(\sigma_i^{pr}, J_j) \leq \sum_{J_j \in \mathcal{J}_i} E(\sigma_i^{pr}, J_j) \leq \sum_{J_j \in \mathcal{J}} E(\sigma_1^{pr}, J_j).$$

Перепишем неравенство 3.4 как

$$E(\sigma^{npr}) \leq (\sqrt[m]{n})^{\alpha-1} \cdot \sum_{i=1}^{k} \sum_{J_j \in \mathcal{J}} E(\sigma_1^{pr}, J_j) \leq m \cdot (\sqrt[m]{n})^{\alpha-1} \cdot \sum_{J_j \in \mathcal{J}} E(\sigma_1^{pr}, J_j).$$

Величина $\sum_{J_j \in \mathcal{J}} E(\sigma_1^{pr}, J_j)$ равна оптимальному расходу энергии в задаче с прерываниями работ на одной машине. Учитывая, что величина оптимального решения задачи $P|pmtn, r_j, d_j|E$ является нижней оценкой на величину оптимального решения задачи $P|r_j, d_j|E$, утверждение 4 влечет

$$E(\sigma^{npr}) \leq m^{\alpha} \cdot (\sqrt[m]{n})^{\alpha-1} \cdot OPT.$$

□

3.2 Минимизация расхода энергии: линейное программирование и вероятностное округление

В этом разделе рассмотрен общий подход к построению приближенных алгоритмов, основанный на решении задач линейного программирования с числом переменных, неограниченным полиномом от размера входа, и последующим вероятностным округлением дробных величин, входящих в полученное решение задачи линейного программирования. Показано, что предложенные методы применимы к различным задачам построения энергетически эффективных расписаний и обобщают или улучшают многие известные теоретические результаты, полученные ранее. В частности, для задачи минимизации расхода энергии при построении допустимого расписания

множества работ на параллельных машинах впервые рассмотрена неоднородная постановка задачи. В неоднородной постановке расход энергии каждой машины определяется собственной функцией зависимости мощности от скорости выполнения работ. Кроме того, допустимые интервалы выполнения каждой работы различны для разных машин. Для обоих вариантов задачи, в которых разрешены прерывания, с последующим выполнением работы на другой машине или с запрещением переноса работы на другую машину получены решения почти того же качества, что и в однородном случае. Более того, основываясь на результатах для задачи с прерываниями, в которой перенос работы на другую машину запрещен, построен приближенный алгоритм для задачи на одной машине без прерываний с лучшей известной оценкой точности. В последнем параграфе с использованием аналогичного метода, получен приближенный алгоритм нахождения энергетически эффективного расписания в цеховой задаче рабочего типа.

3.2.1 Вспомогательные утверждения

В этом параграфе докажем вспомогательные утверждения, которые будут использоваться в анализе точности алгоритмов, рассматриваемых в этом и следующем разделах.

Рассмотрим произвольное множество неотрицательных чисел $E = \{e_1, e_2, \ldots, e_n\}$. Для любого подмножества $S \subseteq E$ этих чисел положим

$$f(S) = \left(\sum_{j \in S} e_j^{1/\alpha} \right)^{\alpha}.$$

Лемма 14 *Пусть задано произвольное множество неотрицательных чисел* $E = \{e_1, e_2, \ldots, e_n\}$. *Предположим, что* $S \subseteq E$ *получено случайным выбором каждого элемента* e_j, $1 \le j \le n$, *независимо с вероятностью* Y_j. *Пусть* $e_n = e_{n+1}$. *Предположим, что* $S' \subseteq E \bigcup \{e_{n+1}\}$ *получено случайным выбором каждого элемента* e_j, $1 \le j \le n+1$, *независимо с вероятностями* $Y'_1, Y'_2, \ldots, Y'_{n+1}$, *где* $Y'_j = Y_j$, *для* $1 \le j \le n-1$, *и* $Y'_n + Y'_{n+1} = Y_n$. *Тогда,*

$$\mathbb{E}[f(S)] \le \mathbb{E}[f(S')]. \tag{3.5}$$

Доказательство. Обозначим через $Pr(T)$ вероятность события, что из множества $U = E \setminus \{e_n\}$ в множество S были выбраны все элементы множества T. Тогда $Pr(T) = \prod_{e_j \in T} Y_j \prod_{e_j \in U \setminus T} (1 - Y_j)$. Имеем

$$\mathbb{E}[f(S)] = \sum_{T \subseteq U} Pr(T) \left[(1 - Y_n) \cdot \left(\sum_{j \in T} e_j^{1/\alpha} \right)^{\alpha} + Y_n \cdot \left(\sum_{j \in T} e_j^{1/\alpha} + e_n^{1/\alpha} \right)^{\alpha} \right]. \tag{3.6}$$

Поскольку $Y_j = Y'_j$ для $1 \le j \le n-1$ выполнено

$$\mathbb{E}[f(S')] = \sum_{T \subseteq U} Pr(T) \left[(1-Y'_n) \cdot (1-Y'_{n+1}) \cdot \left(\sum_{j \in T} e_j^{1/\alpha} \right)^{\alpha} + Y'_n \cdot (1-Y'_{n+1}) \cdot \left(\sum_{j \in T} e_j^{1/\alpha} + e_n^{1/\alpha} \right)^{\alpha} \right.$$

$$+(1-Y'_n)\cdot Y'_{n+1}\cdot\left(\sum_{j\in T}e_j^{1/\alpha}+e_{n+1}^{1/\alpha}\right)^{\alpha}+Y'_n\cdot Y'_{n+1}\cdot\left(\sum_{j\in T}e_j^{1/\alpha}+e_n^{1/\alpha}+e_{n+1}^{1/\alpha}\right)^{\alpha}\Bigg]. \quad (3.7)$$

Для удобства обозначим $A=\sum_{j\in T}e_j^{1/\alpha}$ и $B=e_n^{1/\alpha}=e_{n+1}^{1/\alpha}$. Для доказательства леммы достаточно показать, что выражение в правой части равенства (3.6) меньше, чем выражение в правой части равенства (3.7).

$$(1-Y_n)A^{\alpha}+Y_n\cdot(A+B)^{\alpha}\le(1-Y'_n)\cdot(1-Y'_{n+1})\cdot A^{\alpha}+Y'_n\cdot(1-Y'_{n+1})\cdot(A+B)^{\alpha}$$
$$+(1-Y'_n)\cdot Y'_{n+1}\cdot(A+B)^{\alpha}+Y'_n\cdot Y'_{n+1}\cdot(A+2B)^{\alpha}.$$

Учитывая, что $Y'_n+Y'_{n+1}=Y_n$, перепишем верхнее неравенство как

$$Y'_nY'_{n+1}(A^{\alpha}-2(A+B)^{\alpha}+(A+2B)^{\alpha})\ge 0. \quad (3.8)$$

Если $Y'_n=0$ или $Y'_{n+1}=0$, то неравенство, очевидно, выполнено как равенство. В противном случае справедливость выражения в (3.8) вытекает из выпуклости функции $g(x)=x^{\alpha}$. □

В дальнейшем также потребуется оценка сверху на математическое ожидание функции $g(x)=x^{\alpha}$, когда переменная x равна сумме n независимых случайных величин с распределением Бернулли. Для получения такой верхней оценки воспользуемся результатом из теории вероятностей, который был первоначально доказан Хёфдингом в [107] для конечной суммы независимых случайных величин с распределением Бернулли, а позднее был обобщен в [54] на более общие распределения.

Утверждение 5 [54] *Пусть $X=\sum_{i=1}^{t}X_i$ — сумма t (где t, возможно, равно бесконечности) независимых случайных величин, $0\le X_i\le 1$ для $i=1,\dots,t$ и $\mu=\mathbb{E}[X]$. Для любой выпуклой функции f верно неравенство*

$$\mathbb{E}[f(X)]\le\mathbb{E}[f(Y)],$$

где Y является случайной величиной с биномиальным распределением $Y\sim B(t,\mu/t)$ в случае $t<\infty$ или случайной величиной с распределением Пуассона $Y\sim P(\mu)$ в противном случае.

Используя утверждение 5, докажем следующую лемму.

Лемма 15 *Для любого $\alpha\ge 1$ и произвольной функции $f(x)=x^{\alpha}$ с параметром $a\in[0,1]$ выполнено*

$$\mathbb{E}[f(B_a)]\le\mathbb{E}[f(P_a)],$$

где P_a — случайная величина с распределением Пуассона и параметром a, B_a — сумма n независимых случайных величин с распределением Бернулли, $\mathbb{E}[B_a]=a$.

Доказательство. Рассмотрим случайную величину B'_a, которая является суммой случайной величины B_a и бесконечного числа случайных величин Y'_j, $j = n+1, \ldots, \infty$ таких, что $Pr(Y_j = 1) = 0$. Тогда $\mathbb{E}[B'_a] = \mathbb{E}[B_a] = a$ и $\mathbb{E}[f(B'_a)] = \mathbb{E}[f(B_a)]$. Поскольку функция $f(x)$ выпуклая, то по утверждению 5 получим $\mathbb{E}[f(B_a)] \leq \mathbb{E}[f(P_a)]$. □

Лемма 16 *Для любых действительных $\alpha \geq 1$ и $\lambda \geq 0$ выполнено:*

(a) если $0 \leq \lambda \leq 1$, то $\mathbb{E}[P_\lambda^\alpha] \leq \lambda \mathbb{E}[P_1^\alpha]$;

(b) если $\lambda > 1$, то $\mathbb{E}[P_\lambda^\alpha] \leq \lambda^\alpha \mathbb{E}[P_1^\alpha]$.

Доказательство. Напомним, что $\mathbb{E}[P_\lambda^\alpha] = \sum_{k=0}^{\infty} k^\alpha \frac{\lambda^k e^{-\lambda}}{k!}$.

(a) Замечая, что $e^{-(1-\lambda)} \geq 1 - (1-\lambda) = \lambda \geq \lambda^{k-1}$ для $k \geq 2$ и $0 \leq \lambda \leq 1$, получим $e^{-1} \geq \lambda^{k-1} e^{-\lambda}$ для всех $k \geq 2$. Для $\lambda = 0$ утверждение леммы тривиально. Предположим, что $\lambda > 0$, тогда

$$\mathbb{E}[P_1^\alpha] - \frac{1}{\lambda}\mathbb{E}[P_\lambda^\alpha] = \sum_{k=0}^{\infty} k^\alpha \frac{e^{-1} - \lambda^{k-1} e^{-\lambda}}{k!} = (e^{-1} - e^{-\lambda}) + \sum_{k=2}^{\infty} k^\alpha \frac{e^{-1} - \lambda^{k-1} e^{-\lambda}}{k!}$$

$$\geq (e^{-1} - e^{-\lambda}) + \sum_{k=2}^{\infty} k \frac{e^{-1} - \lambda^{k-1} e^{-\lambda}}{k!} = \sum_{k=0}^{\infty} k \frac{e^{-1}}{k!} - \frac{1}{\lambda} \sum_{k=0}^{\infty} k \frac{\lambda^k e^{-\lambda}}{k!} = 1 - \frac{1}{\lambda} \cdot \lambda = 0.$$

(b) Для доказательства леммы в случае $\lambda > 1$ воспользуемся следующими двумя хорошо известными фактами.

- Сумма независимых случайных величин X_1 и X_2, имеющих распределение Пуассона с параметрами λ_1 и λ_2, имеет распределение Пуассона с параметром $\lambda_1 + \lambda_2$.
- Для любой случайной величины X величина $\mathbb{E}[X^p]^{1/p}$ определяет норму и, следовательно, удовлетворяет неравенству Минковского $||X + Y||_p \leq ||X||_p + ||Y||_p$.

Сначала докажем утверждение для рациональных чисел. Предположим, что $\lambda = \frac{A}{B}$ рациональное число, то есть $A, B \in Z_+$ и $A > B \geq 1$. Пусть случайная величина X имеет распределение Пуассона с параметром λ и $Y_1, \ldots, Y_A$ — независимые случайные величины, которые имеют распределение Пуассона с параметром $1/B$. Дополнительно, пусть $r = \binom{A-1}{B-1}$, и Y_S — следующая случайная величина

$$Y_S = \frac{\sum_{i \in S} Y_i}{r}.$$

Тогда

$$X = \sum_{i=1}^{A} Y_i = \sum_{S \subseteq [A]: |S| = B} Y_S,$$

где $[A] = \{1, \ldots, A\}$. Пусть P_1 — пуассоновская случайная величина с параметром 1, тогда, применяя неравенство Минковского, получим

$$\mathbb{E}[X^\alpha]^{1/\alpha} \le \sum_{S \subseteq [A]:|S|=B} \mathbb{E}[Y_S^\alpha]^{1/\alpha} = \frac{\binom{A}{B}}{r} \mathbb{E}[P_1^\alpha]^{1/\alpha} = \lambda \mathbb{E}[P_1^\alpha]^{1/\alpha}.$$

Утверждение леммы для произвольного рационального числа $\lambda > 1$ доказано.

Для доказательства утверждения леммы для произвольного действительного числа $\lambda > 1$ рассмотрим последовательность рациональных чисел, стремящихся к λ. Поскольку требуемое неравенство выполнено для всех рациональных чисел, переходя к пределу, получим, что неравенство справедливо и для действительного числа λ. □

Лемма 17 *Для любого множества положительных чисел $S = \{e_1, e_2, \ldots, e_n\}$ и некоторой константы $\alpha > 1$ справедливо неравенство*

$$\left(\sum_{e_i \in S} e_i^{1/\alpha} \right)^\alpha \le |S|^{\alpha-1} \sum_{e_i \in S} e_i.$$

Доказательство. Заметим, что функция $h(x) = x^{1/\alpha}$ вогнута при $\alpha > 1$. Следовательно, при любых $q_1, q_2, \ldots, q_n$ таких, что $q_1, q_2, \ldots, q_n > 0$ и $q_1 + q_2 + \cdots + q_n = 1$, для функции $h(x)$ выполнено неравенство Йенсена [5]

$$\sum_{e_i \in S} q_i h(e_i) \le h\left(\sum_{e_i \in S} q_i e_i \right). \tag{3.9}$$

Полагая $q_i = \frac{1}{|S|}$ для всех $i = 1, \ldots, n$, получим

$$\frac{1}{|S|} \sum_{e_i \in S} e_i^{1/\alpha} \le \left(\frac{1}{|S|} \sum_{e_i \in S} e_i \right)^{1/\alpha}.$$

Возведя обе части неравенства в степень α, придем к утверждению леммы. □

Лемма 18 *Пусть ξ_i, $i = 1, \ldots, n$ — последовательность независимых случайных величин, имеющих дискретные распределения с множествами возможных неотрицательных значений. Тогда для произвольной константы $\alpha \ge 1$ выполнено*

$$\mathbb{E}\left(\sum_{i=1}^n \xi_i^{1/\alpha} \right)^\alpha \le \left(\sum_{i=1}^n (\mathbb{E}\xi_i)^{1/\alpha} \right)^\alpha.$$

Доказательство. Чтобы избежать громоздких обозначений, докажем утверждение для двух независимых случайных величин ξ и η. Пусть случайная величина η принимает с ненулевой вероятностью значения $b_1, \ldots, b_k$. Дифференцированием по α легко

проверить, что для любых $\alpha \geq 1$ и $A \geq 0$ функция $\left(x^{1/\alpha} + A\right)^{\alpha}$ вогнута на интервале $[0, +\infty)$. Дважды применяя неравенство Йенсена для вогнутых функций, получаем

$$\mathbb{E}(\xi^{1/\alpha} + \eta^{1/\alpha})^{\alpha} = \sum_{i=1}^{k} \mathbb{E}\left(\xi^{1/\alpha} + b_i^{1/\alpha}\right)^{\alpha} \mathsf{P}(\eta = b_i) \leq$$

$$\sum_{i=1}^{k} \left((\mathbb{E}\xi)^{1/\alpha} + b_i^{1/\alpha}\right)^{\alpha} \mathsf{P}(\eta = b_i) = \mathbb{E}\left((\mathbb{E}\xi)^{1/\alpha} + \eta^{1/\alpha}\right)^{\alpha} \leq \left((\mathbb{E}\xi)^{1/\alpha} + (\mathbb{E}\eta)^{1/\alpha}\right)^{\alpha},$$

где первое равенство следует из независимости случайных величин ξ и η. Неравенство для произвольного числа независимых случайных величин может быть получено тем же способом при последовательном применении неравенств Йенсена к каждой из случайных величин. □

3.2.2 Неоднородная задача на параллельных машинах с прерываниями без переноса работ

В этом параграфе рассмотрим задачу $P|pmtn*, r_{ij}, d_{ij}, W_{ij}, \alpha_i|E$. Задано множество $\mathcal{J}$, состоящее из n работ, и множество $\mathcal{M}$, состоящее из m параллельных машин. Для каждой машины M_i задана собственная функция зависимости мощности от скорости выполнения работ $P(S) = S^{\alpha_i}$. Для каждой работы $J_j \in \mathcal{J}$ заданы момент поступления $r_{i,j}$, директивный срок $d_{i,j}$ и объем $W_{i,j}$, если работа j выполняется на машине $M_i \in \mathcal{M}$. Все перечисленные параметры зависят как от работы, так и от машины и могут быть различными для одной и той же работы. Выполнение работы может быть прервано и продолжено позднее на той же самой машине, т.е. выполнение работы на нескольких машинах не допускается. Требуется найти допустимое расписание, минимизирующее общий расход энергии.

Соответствующая однородная задача является NP-трудной, даже если все работы имеют общие времена поступления и директивные сроки [30]. В следующем параграфе представим задачу $P|pmtn*, r_{ij}, d_{ij}, W_{ij}, \alpha_i|E$ как задачу целочисленного линейного программирования (задача ЦЛП) с экспоненциальным числом переменных и полиномиальным числом ограничений. При снятии ограничения на целочисленность переменных задача ЦЛП превращается в задачу линейного программирования (задача ЛП). Предположим, что решение задачи ЛП известно. Тогда допустимое решение исходной задачи можно получить вероятностным округлением дробных значений переменных. Чтобы найти оптимальное решение задачи ЛП за полиномиальное время рассмотрим другую более компактную формулировку задачи $P|pmtn*, r_{ij}, d_{ij}, W_{ij}, \alpha_i|E$ как задачи ЦЛП и докажем, что линейные релаксации этих двух формулировок эквивалентны.

Задача линейного программирования

Сначала покажем, что при незначительной потере точности можно рассматривать расписания, удовлетворяющие следующим свойствам:

- на каждой машине для каждой работы задано множество интервалов одинаковой длины, в которых она может выполняться;
- число таких интервалов ограничено полиномом от числа работ и величины $\frac{1}{\varepsilon}$ для заданного фиксированного параметра точности ε.

Лемма 19 *Существует допустимое расписание σ такое, что $E(\sigma) \leq ((1+\frac{\varepsilon}{1-\varepsilon})(1+\frac{2}{n-2}))^{\alpha} \cdot OPT$ и каждый временной интервал обслуживания каждой работы $J_j \in \mathcal{J}$, выполняемой на машине $M_i \in \mathcal{M}$, начинается и заканчивается в моменты $r_{i,j} + k\frac{\varepsilon}{n^3}(d_{i,j} - r_{i,j})$, где $k \geq 0$ некоторое целое число.*

Доказательство. Пусть σ^* — произвольное оптимальное расписание для некоторого примера задачи $P|pmtn*, r_{ij}, d_{ij}, W_{ij}, \alpha_i|E$. Сначала преобразуем его в допустимое расписание σ, в котором на каждой машине M_i выполнение любой назначенной на нее работы J_j требует по крайней мере $\frac{\varepsilon}{n}(d_{i,j} - r_{i,j})$ единиц времени.

Так как все времена поступления работ и их директивные сроки целые, то их можно разделить на интервалы единичной длины. В каждом единичном интервале увеличим скорость машины на величину $1 + \frac{\varepsilon}{1-\varepsilon}$. При этом в каждом единичном интервале появится интервал длины ε, в котором машина простаивает, и общий расход энергии увеличится в $(1 + \frac{\varepsilon}{1-\varepsilon})^{\alpha-1}$ раз. Для каждой работы J_j в каждом единичном интервале внутри интервала $[r_{i,j}, d_{i,j}]$ зарезервируем $\frac{\varepsilon}{n}$ единиц времени на той машине, на которой она выполняется в расписании σ^*. Соответственно уменьшим скорость выполнения каждой работы так, чтобы время ее выполнения увеличилось на величину $\frac{\varepsilon}{n}(d_{i,j} - r_{i,j})$. Заметим, что по построению существует допустимое расписание для рассматриваемого примера, в котором все работы могут быть выполнены с найденными скоростями.

Зафиксируем найденные скорость выполнения и длительность каждой работы. Упорядочим работы по неубыванию директивных сроков. Назначим каждую работу на ту же машину, на которой она выполнялась в расписании σ^*. На каждой машине построим расписание по следующему правилу: в каждый момент времени выполняется незаконченная работа с наименьшим индексом. Полученное расписание σ также является допустимым, и общее число прерываний в нем не больше n.

Преобразуем полученное расписание σ в расписание σ', удовлетворяющее условиям леммы. Пусть работа J_j выполняется в расписании σ на машине M_i. Разобьем интервал $[r_{i,j}, d_{i,j}]$ в подинтервалы длины $\frac{\varepsilon}{n^3}(d_{i,j} - r_{i,j})$, т.е. в интервалы $[r_{i,j} + k\frac{\varepsilon}{n^3}(d_{i,j} - r_{i,j}), r_{i,j} + (k+1)\frac{\varepsilon}{n^3}(d_{i,j} - r_{i,j})]$, где $k \geq 0$ некоторое целое число. Так как в расписании σ длительность работы J_j не меньше чем $\frac{\varepsilon}{n}(d_{i,j} - r_{i,j})$, то эта работа должна выполняться целиком или частично по крайней мере в n^2 подинтервалах. Так как выполнение каждой работы в расписании σ прерывается не более n раз, то она может частично выполняться не более чем в $2n$ интервалах. Интервал

$[r_{i,j} + k\frac{\varepsilon}{n^3}(d_{i,j} - r_{i,j}), r_{i,j} + (k+1)\frac{\varepsilon}{n^3}(d_{i,j} - r_{i,j}))$ назовем целым для работы J_j, если только эта работа выполняется в течение всего интервала. Увеличив скорость выполнения работ, построим расписание σ', в котором все работы выполняются только в своих целых интервалах. Число таких интервалов не меньше $n^2 - 2n$. Следовательно, общий расход энергии в расписании σ' превышает расход энергии в расписании σ не больше чем в $(1 + \frac{2}{n-2})^{\alpha-1}$ раз. □

Пусть расписание σ удовлетворяет условиям леммы 19 и работа J_j выполняется в расписании σ на машине M_i. Тогда интервал $[r_{i,j}, d_{i,j}]$ можно разбить на полиномиальное от числа работ и величины $\frac{1}{\varepsilon}$ интервалов равной длины, в которых либо в течение всего интервала выполняется работа J_j, либо она в нем не выполняется. В дальнейшем будем рассматривать только расписания, удовлетворяющие этому свойству.

Конфигурацией c назовем расписание одной работы на одной машине. Более точно, конфигурация каждой работы задает множество интервалов, в которые эта работа выполняется. Пусть для работы J_j задана некоторая конфигурация c, тогда длительность работы J_j равна произведению числа заданных интервалов на их длину. В силу выпуклости функции зависимости мощности от скорости выполнения работ среди всех расписаний, удовлетворяющих лемме 19, минимальный расход энергии достигается на расписании, в котором скорость выполнения работы J_j постоянна. Следовательно, оптимальная скорость выполнения работы S_j в каждой конфигурации равна отношению ее объема к ее длительности. Обозначим через $\mathcal{C}_{ij}$ множество всех возможных конфигураций работы J_j на машине M_i.

Рассмотрим некоторую машину M_i. Заметим, что каждая работа определяет свой набор подинтервалов и, следовательно, свою последовательность точек вида $r_{i,j} + k\frac{\varepsilon}{n^3}(d_{i,j} - r_{i,j})$, введенных в лемме 19. Пусть $t_{i,1}, t_{i,2}, \ldots, t_{i,\ell_i}$ — упорядоченная по времени последовательность всех таких точек. Рассмотрим интервалы $[t_{i,p}, t_{i,p+1})$, $1 \le p \le \ell_i - 1$. В каждом из таких интервалов согласно лемме 19 либо в течение всего интервала выполняется одна работа, либо в нем не выполняется ни одной работы. Обозначим множество таких интервалов через $\mathcal{I}$.

Введем бинарную переменную $x_{i,j,\mathsf{c}}$, которая равна 1, если работа J_j выполняется на машине M_i в соответствии с конфигурацией c, и 0, иначе. Если конфигурация задана, то можно вычислить расход энергии $E_{i,j,\mathsf{c}}$ на выполнение работы J_j на машине M_i. Пусть интервал $I \in \mathcal{I}$. Для упрощения обозначений будем писать, что $I \in (i, j, \mathsf{c})$, если в конфигурации c существует интервал $[r_{i,j} + k\frac{\varepsilon}{n^3}(d_{i,j} - r_{i,j}), r_{i,j} + (k+1)\frac{\varepsilon}{n^3}(d_{i,j} - r_{i,j})]$, который содержит интервал I.

Рассмотрим следующую задачу целочисленного линейного программирования.

Задача ILP_1

$$\min \sum_{i,j,\mathsf{c}} E_{i,j,\mathsf{c}} \cdot x_{i,j,\mathsf{c}}$$

$$\sum_{i,c} x_{i,j,\mathsf{c}} \geq 1, \qquad \forall j \in \mathcal{J}, \tag{3.10}$$

$$\sum_{(i,j,\mathsf{c}):I \in (i,j,\mathsf{c})} x_{i,j,\mathsf{c}} \leq 1, \qquad \forall I \in \mathcal{I}, \tag{3.11}$$

$$x_{i,j,\mathsf{c}} \in \{0,1\} \qquad \forall M_i \in \mathcal{M}, j \in \mathcal{J}, \mathsf{c} \in \mathcal{C}_{ij}. \tag{3.12}$$

Неравенство (3.10) гарантирует, что для каждой работы выбрана ровно одна конфигурация. Неравенство (3.11) определяет, что не больше одной работы выполняется в каждом интервале $[t_{i,p}, t_{i,p+1})$, $1 \leq p \leq \ell_i - 1$.

Рандомизированное округление

Положив $x_{i,j,\mathsf{c}} \geq 0$, ослабим условие целочисленности в ограничении (3.12). Хотя полученная задача линейного программирования (задача LP_1) имеет огромное (экспоненциальное) число переменных, ее структура достаточна проста. Сначала предположим, что оптимальное решение задачи LP_1 известно. Выберем для каждой работы одну ее конфигурацию c случайно с вероятностью, пропорциональной $x_{i,j,\mathsf{c}}$. Напомним, что каждая конфигурация задает для работы множество интервалов, в которых она выполняется. При этом в процессе выбора в один интервал может попасть несколько работ. В таком случае увеличим скорость выполнения работ внутри интервала в необходимое число раз так, чтобы выполнить все работы в течение этого интервала. Формальное описание данной процедуры представлено ниже в алгоритме 3.5.

Алгоритм 3.5

1: Решить задачу линейного программирования.
2: Для каждой работы $J_j \in \mathcal{J}$ случайно выбрать одну конфигурацию c с вероятностью $x_{i,j,\mathsf{c}}$.
3: Для каждой работы J_j вычислить скорость ее выполнения S_j согласно конфигурации выбранной на шаге 2.
4: Для каждого интервала I определить множество работ $JOB(I)$, которые должны быть выполнены в интервале I согласно выбору на шаге 2.
5: Для каждого интервала I выполнить все работы множества $JOB(I)$ со скоростью $\sum_{J_j \in JOB(I)} S_j$.

Отметим, что трудоемкость алгоритма 3.5 зависит от реализации шага 1. В дальнейшем рассмотрим другую более компактную формулировку задачи $P|pmtn*, r_{ij}, d_{ij}, W_{ij}, \alpha_i|E$ как задачи целочисленного линейного программирования и установим эквивалентность соответствующих им задач линейного программирования, получаемых при ослаблении условий целочисленности.

Теорема 15 *Пусть $\alpha_i \geq 1$ для всех $i = 1, \ldots, m$. Алгоритм 3.5 является $((1 + \frac{\varepsilon}{1-\varepsilon})(1+\frac{2}{n-2}))^{\alpha}\tilde{B}_{\alpha}$-приближенным алгоритмом для задачи $P|pmtn*, r_{ij}, d_{ij}, W_{ij}, \alpha_i|E$, где $\alpha = \max_{i\in\mathcal{M}} \alpha_i$.*

Доказательство. Для каждого интервала $I \in \mathcal{I}$ оценим средний ожидаемый расход энергии машины. По определению интервал соответствует некоторой машине $M_i \in \mathcal{M}$. Для каждой работы $J_j \in \mathcal{J}$ пусть n_j обозначает число ненулевых значений переменной $x_{i,j,\mathsf{c}}$ таких, что $I \in (i, j, \mathsf{c})$. Обозначим через $X_{j,k}$ значение k-й из упомянутых выше переменной и через $S_{j,k}$ соответствующую ей скорость выполнения работы J_j, $1 \leq k \leq n_j$.

Пусть Y_j будет вероятность того, что работа J_j назначена на выполнение в интервал I на шаге 2 алгоритма 3.5. Имеем $Y_j = \sum_{k=1}^{n_j} X_{j,k}$. Ограничение (3.11) влечет $\sum_{j=1}^{n} Y_j \leq 1$. Предположим, что работа J_j выполняется в интервале I согласно конфигурации, соответствующей k-й переменной. Тогда расход энергии, требуемой на выполнение работы J_j, согласно выбранной конфигурации равен $e_{j,k} = |I|S_{j,k}^{\alpha_i}$. Следовательно, значение целевой функции в оптимальном решении задачи линейного программирования можно записать как $LP_I^* = \sum_{j=1}^{n}\sum_{k=1}^{n_j} e_{j,k}X_{j,k}$.

Предположим, что на шаге 2 алгоритма 3.5 для выполнения в интервале I было выбрано множество работ H. Вероятность такого события равна

$$Pr(H) = \prod_{J_j\in H} Y_j \prod_{J_j\in\mathcal{J}\setminus H} (1 - Y_j).$$

Пусть $E(H)$ — расход энергии машиной M_i в интервале I, если в нем выполняется множество работ H. Тогда математическое ожидание полного расхода энергии E_I в интервале I в решении, полученном алгоритмом 3.5, равно

$$E_I = \sum_{H\subseteq\mathcal{J}} E(H) \prod_{J_j\in H} Y_j \prod_{J_j\in\mathcal{J}\setminus H} (1 - Y_j).$$

Оценим величину $E(H)$ при условии, что множество работ H назначено алгоритмом в интервал I. Тогда для каждой работы $J_j \in H$ на шаге 2 алгоритма 3.5 была выбрана одна из конфигураций, соответствующая некоторой паре (j, k) с вероятностью $X_{j,k}$. Обозначим через $U(H)$ множество всех комбинаций пар (j, k), которые определяют выполнение множества работ H в течение интервала I. Пусть алгоритм назначил на выполнение множество работ H в интервал I согласно конфигурациям из U, где $U \subseteq U(H)$, тогда общий расход энергии в течение интервала I в расписании, полученном алгоритмом, равен $|I|\left(\sum_{(j,k)\in U} S_{j,k}\right)^{\alpha_i}$. Отсюда

$$E(H) = \sum_{U\subseteq U(H)} \left(\prod_{(j,k)\in U} \frac{X_{j,k}}{Y_j}\right) |I| \left(\sum_{(j,k)\in U} S_{j,k}\right)^{\alpha_i}.$$

Для каждой работы $J_j \in H$ обозначим через $\tilde{e}_j$ случайную величину, принимаю-

щую значение $e_{j,k}$ с вероятностью $\frac{X_{j,k}}{Y_j}$. Тогда можно записать следующее равенство

$$E(H) = \mathbb{E}\left[\left(\sum_{J_j \in H} \tilde{e}_j^{1/\alpha_i}\right)^{\alpha_i}\right].$$

По лемме 18 получим

$$E(H) \leq \left(\sum_{J_j \in H} \mathbb{E}\left[\tilde{e}_j\right]^{1/\alpha_i}\right)^{\alpha_i}.$$

Положим $e_j = \mathbb{E}\left[\tilde{e}_j\right]$. Оценим величину E_I, как

$$E_I \leq \sum_{H \subseteq \mathcal{J}} \left(\sum_{J_j \in H} e_j^{1/\alpha_i}\right)^{\alpha_i} \prod_{J_j \in H} Y_j \prod_{J_j \in \mathcal{J} \setminus H} (1 - Y_j). \tag{3.13}$$

Заметим, что все коэффициенты в задаче (3.10 - 3.12) рациональные числа. Следовательно, величины Y_j, $j = 1, \ldots, n$, также рациональные числа. Тогда существует целое положительное число Q такое, что $Y_j = \frac{q_j}{Q}$, $1 \leq j \leq n$, для некоторого $q_j \in \mathbb{N}$. Пусть $Y = 1/Q$ и $q = \sum_{j=1}^{n} q_j$. Заметим, что $q \leq Q$. Применим лемму 14 итеративно к правой части неравенства (3.13). Получим

$$E_I \leq \sum_{H \subseteq \{1,2,\ldots,q\}} \left(\sum_{J_j \in H} e_j^{1/\alpha_i}\right)^{\alpha_i} Y^{|H|}(1-Y)^{q-|H|},$$

откуда по лемме 17 имеем

$$\begin{aligned} E_I &\leq \sum_{H \subseteq \{1,2,\ldots,q\}} |H|^{\alpha_i - 1} \left(\sum_{J_j \in H} e_j\right) Y^{|H|}(1-Y)^{q-|H|} \\ &= \sum_{k=1}^{q} \sum_{H \subseteq \{1,2,\ldots,q\}, |H|=k} \left(\sum_{J_j \in H} e_j\right) k^{\alpha_i - 1} Y^k (1-Y)^{q-k}. \end{aligned}$$

Меняя порядок суммирования в последнем неравенстве, придем к следующей оценке на величину E_I :

$$\begin{aligned} E_I &\leq \left(\sum_{j=1}^{q} e_j\right) \sum_{k=1}^{q} \binom{q-1}{k-1} k^{\alpha_i - 1} Y^k (1-Y)^{q-k} \\ &= \left(\frac{\sum_{j=1}^{q} e_j}{q}\right) \sum_{k=1}^{q} \binom{q}{k} k^{\alpha_i} Y^k (1-Y)^{q-k} \\ &= \left(\frac{\sum_{j=1}^{n} q_j e_j}{q}\right) \sum_{k=1}^{q} \binom{q}{k} k^{\alpha_i} Y^k (1-Y)^{q-k} \\ &= \frac{Q}{q} LP_I^* \sum_{k=1}^{q} \binom{q}{k} k^{\alpha_i} Y^k (1-Y)^{q-k} = \frac{Q}{q} LP_I^* \cdot \mathbb{E}[B_{q/Q}^{\alpha_i}], \end{aligned}$$

где $B_{q/Q}$ случайная величина с биномиальным распределением и математическим ожиданием $\frac{q}{Q}$, соответствующая сумме q независимых одинаково распределенных случайных величин с распределением Бернулли. Следовательно,

$$E_I \quad \leq \quad \frac{Q}{q} LP_I^* \cdot \mathbb{E}[B_{q/Q}^{\alpha_i}] \leq \frac{Q}{q} LP_I^* \cdot \mathbb{E}[P_{q/Q}^{\alpha_i}] \leq \frac{Q}{q} LP_I^* \cdot \frac{q}{Q} \mathbb{E}[P_1^{\alpha_i}],$$

где второе неравенство следует из леммы 15, а последнее неравенство — из леммы 16(а). Суммируя по всем интервалам и машинам, окончательно получим

$$E \leq LP^* \cdot \mathbb{E}[P_1^{\alpha}] = LP^* \cdot \tilde{B}_{\alpha}.$$

□

Компактная задача линейного программирования

Рассмотрим другую формулировку задачи $P|pmtn*, r_{ij}, d_{ij}, W_{ij}, \alpha_i|E$ как задачи целочисленного линейного программирования, которую будем называть задачей ILP_2. Напомним, что по лемме 19 существует $((1+\frac{\varepsilon}{1-\varepsilon})(1+\frac{2}{n-2}))^{\alpha}$-приближенное расписание, удовлетворяющее следующим свойствам:

- на каждой машине M_i для каждой работы J_j интервал $[r_{i,j}, d_{i,j})$ разбит на равные промежутки времени, в которых либо работа J_j выполняется в течение всего интервала времени, либо не выполняется вовсе;
- для каждой машины M_i и каждой работы J_j число выделенных ей промежутков равно $\frac{n^3}{\varepsilon}$ и их длина равна $\frac{\varepsilon}{n^3}(d_{i,j} - r_{i,j})$.

В свою очередь, концы промежутков каждой работы разбивают время работы каждой машины на множество интервалов $\mathcal{I}$. Введем новые бинарные переменные $y_{i,j,q}$ и $z_{i,j,q,t}$. Переменная $y_{i,j,q}$ равна 1, если работа J_j выполняется на машине M_i ровно в q выделенных промежутках, и 0, иначе. Переменная $z_{i,j,q,t}$ равна 1, если работа J_j выполняется на машине M_i ровно в q выделенных промежутках и, в том числе, выполняется в t-ом выделенном промежутке, и 0 иначе. Обозначим через $p_{i,j,q} = q\frac{\varepsilon}{n^3}(d_{i,j} - r_{i,j})$ и $E_{i,j,q} = \frac{w_{i,j}^{\alpha_i}}{p_{i,j,q}^{\alpha_i - 1}}$ общее время выполнения и расход энергии работы J_j, если она выполняется ровно в q выделенных промежутках на машине M_i.

Задача ILP_2

$$\min \sum_{i,j,q} E_{i,j,q} \cdot y_{i,j,q}$$

$$\sum_{i,q} y_{i,j,q} = 1, \qquad \forall J_j \in \mathcal{J}, \tag{3.14}$$

$$\sum_{t} z_{i,j,q,t} = q \cdot y_{i,j,q}, \qquad \forall M_i \in \mathcal{M}, J_j \in \mathcal{J}, q \in \{1, 2, \ldots, \tfrac{n^3}{\varepsilon}\}, \tag{3.15}$$

$$\sum_{j,q} \sum_{t: I \subseteq t} z_{i,j,q,t} \leq 1, \qquad \forall M_i \in \mathcal{M}, I \in \mathcal{I}, \tag{3.16}$$

$$y_{i,j,q}, z_{i,j,q,t} \in \{0,1\}, \qquad \forall i \in \mathcal{M}, J_j \in \mathcal{J}, q, t \in \{1, 2, \ldots, \tfrac{n^3}{\varepsilon}\}. \tag{3.17}$$

Ограничение (3.14) гарантирует, что каждая работа целиком выполняется на одной машине. Ограничение (3.15) связывает переменные $y_{i,j,q}$ и $z_{i,j,q,t}$. Действительно, если $y_{i,j,q} = 1$, то ровно q переменных $z_{i,j,q,t}$ должны быть равны 1. Запись $I \subseteq t$ в ограничении (3.16) означает, что интервал $I \in \mathcal{I}$ лежит в t-ом выделенном промежутке для работы J_j на машине M_i. Таким образом, ограничение (3.16) гарантирует, что не более одной работы выполняется на каждой машине в каждый момент времени. Заметим, что число переменных ограничено полиномом от n и $1/\varepsilon$.

Задачи ILP_1 и ILP_2 эквивалентны и их оптимальные решения соответствуют расписаниям с минимальным расходом энергии среди всех расписаний, удовлетворяющих условиям леммы 19. Рассмотрим соответствующие им задачи линейного программирования LP_1 и LP_2, которые получаются ослаблением условий целочисленности в ограничениях (3.12) и (3.17) соответственно. Покажем, что по любому допустимому решению задачи LP_1 можно получить допустимое решение задачи LP_2 с тем же значением целевой функции, и наоборот.

Лемма 20 *Задачи LP_1 и LP_2 эквивалентны.*

Доказательство. Рассмотрим произвольное допустимое решение задачи LP_1. Напомним, что значение переменной $x_{i,j,\mathsf{c}}$ определяет часть работы $J_j \in \mathcal{J}$, которая выполняется на машине M_i в промежутках из конфигурации $\mathsf{c} \in \mathcal{C}_{ij}$. Обозначим через $\mathcal{C}_{ijq} \subseteq \mathcal{C}_{ij}$ множество конфигураций работы J_j на машине M_i, в которых она выполняется ровно в q промежутках. Тогда $E_{i,j,\mathsf{c}} = E_{i,j,q}$ для всех $\mathsf{c} \in \mathcal{C}_{ijq}$. Положим $y_{i,j,q} = \sum_{\mathsf{c} \in \mathcal{C}_{ijq}} x_{i,j,\mathsf{c}}$. Непосредственно отсюда получаем

$$\sum_{i,q} y_{i,j,q} = \sum_{i,q} \sum_{\mathsf{c} \in \mathcal{C}_{ijq}} x_{i,j,\mathsf{c}} = \sum_{i,\mathsf{c}} x_{i,j,\mathsf{c}} = 1$$

для каждой $J_j \in \mathcal{J}$ и

$$\sum_{i,j,q} E_{i,j,q} \cdot y_{i,j,q} = \sum_{i,j,q} \sum_{\mathsf{c} \in \mathcal{C}_{ijq}} E_{i,j,q} \cdot x_{i,j,\mathsf{c}} = \sum_{i,j,q} \sum_{\mathsf{c} \in \mathcal{C}_{ijq}} E_{i,j,\mathsf{c}} \cdot x_{i,j,\mathsf{c}} = \sum_{i,j,\mathsf{c}} E_{i,j,\mathsf{c}} \cdot x_{i,j,\mathsf{c}}.$$

Полагая $z_{i,j,q,t} = \sum_{c \in \mathcal{C}_q : t \in c} x_{i,j,c}$, получим допустимое решение задачи LP_2.

Рассмотрим произвольное допустимое решение задачи LP_2. Определим множество конфигураций, назначая ненулевые значения переменным $x_{i,j,\mathsf{c}}$, соответствующим этим конфигурациям, так, чтобы число конфигураций с ненулевыми значениями переменных было ограничено полиномом от n и $\frac{1}{\varepsilon}$.

Для этого в решении задачи LP_2 рассмотрим произвольную ненулевую переменную $y_{i,j,q}$ и соответствующие ей ненулевые переменные $z_{i,j,q,t}$. Чтобы определить значения переменных $x_{i,j,\mathsf{c}}$, построим двудольный граф $G = (A \cup B, E)$ на $q + \frac{n^3}{\varepsilon}$ вершинах. Доля A содержит q вершин, т.е. $A = \{a_1, a_2, \ldots, a_q\}$. Интуитивно ясно, что каждая вершина соответствует одному из q промежутков конфигурации, которая соответствует переменной $y_{i,j,q}$. Доля B содержит $\frac{n^3}{\varepsilon}$ вершин, каждая из которых соответствует промежутку, выделенному работе J_j на машине M_i, т.е. $B = \{b_1, b_2, \ldots, b_{\frac{n^3}{\varepsilon}}\}$. Определим множество ребер E и веса на них так, что сумма весов ребер инцидентных каждой вершине $a_k \in A$ равна $y_{i,j,q}$ и сумма весов ребер, инцидентных каждой вершине $b_t \in B$, равна $z_{i,j,q,t}$. Напомним, что $q \cdot y_{i,j,q} = \sum_t z_{i,j,q,t}$. Сначала построим ребра из вершины a_1. Пусть $l = \min\{k | \sum_{t=1}^{k} z_{i,j,q,t} \geq y_{i,j,q}\}$. Соединим вершину a_1 с вершинами $b_1, \ldots, b_l$. Вес ребра (a_1, b_k) положим равным $z_{i,j,q,k}$ для $k = 1, \ldots, l-1$ и вес ребра (a_1, b_l) положим равным $y_{i,j,q} - \sum_{t=1}^{l-1} z_{i,j,q,t}$. Если $y_{i,j,q} - \sum_{t=1}^{l-1} z_{i,j,q,t} < z_{i,j,q,l}$, то вершину b_l дополнительно соединим ребром с вершиной a_2. Вес ребра (b_1, a_2) положим равным $z_{i,j,q,l} - (y_{i,j,q} - \sum_{\ell=1}^{k-1} z_{i,j,q,t})$. Продолжим добавлять ребра от вершины a_2 к вершинам $b_{l+1}, b_{l+2}, \ldots$ с весами $z_{i,j,q,l+1}, z_{i,j,q,l+2}, \ldots$ соответственно, пока сумма весов не превысит $y_{i,j,q}$. Затем перейдем к вершине a_3, и так далее. Заметим, что в построенном двудольном графе степень каждой вершины в доле B равна один или два.

Применяя алгоритм Гонзалеза-Сани [97], найдем r совершенных паросочетаний $M_1, M_2, \ldots, M_r$ и набор положительных весов $\lambda_1, \lambda_2, \ldots, \lambda_r$ таких, что $\sum_{i=1}^{r} \lambda_i = y_{i,j,q}$ и $\sum_{i:e \in M_i} \lambda_i = w_e$, где w_e — вес ребра e в графе G.

Заметим, что каждое полученное паросочетание соответствует допустимой конфигурации работы J_j. Число таких конфигураций не превышает числа ребер в графе G и, следовательно, ограничено полиномом от n и $1/\varepsilon$. Полагая $x_{i,j,\mathsf{c}} = \lambda_i$ для каждой конфигурации c, соответствующей паросочетанию M_i, получим допустимое решение задачи LP_1. □

3.2.3 Задача на одной машине без прерываний работ

В этом параграфе представим приближенный алгоритм для задачи $1|r_i, d_i|E$, в которой множество работ должно быть выполнено на одной машине и прерывания в процессе обслуживания работ запрещены.

В [36] для задачи $1|r_i, d_i|E$, предложен $2^{5\alpha-4}$-приближенный алгоритм, основанный на серии преобразований исходного примера к примеру задачи минимизации L_α-нормы вектора загрузки различных одностадийных машин.

Наш алгоритм основан на сведении задачи $1|r_i, d_i|E$ к задаче $P|pmtn*, r_{ij}, d_{ij}, W_{ij}, \alpha_i|E$, рассмотренной в предыдущей главе. Первый этап сведения

совпадает с первым преобразованием примера задачи $1|r_i, d_i|E$, предложенным в [36]. Для полноты картины опишем этот этап.

Разобьем множество работ на подмножества и интервал выполнения работ на подинтервалы следующим образом. Положим $t_1 = \min\{d_j : J_j \in \mathcal{J}\}$. Рассмотрим $\mathcal{J}_1 \subseteq \mathcal{J}$ подмножество работ, которые поступили на выполнение не позднее момента t_1, т.е. $\mathcal{J}_1 = \{J_j \in \mathcal{J} : r_j \leq t_1\}$. Затем положим $t_2 = \min\{d_j : J_j \in \mathcal{J} \setminus \mathcal{J}_1\}$ и $\mathcal{J}_2 = \{J_j \in \mathcal{J} : t_1 < r_j \leq t_2\}$. Продолжим описанную процедуру до тех пор, пока не назначим по подмножествам все работы. Пусть число построенных подмножеств равно k. Дополнительно, пусть $t_0 = \min\{r_j : J_j \in \mathcal{J}\}$ и $t_{k+1} = \max\{d_j : J_j \in \mathcal{J}\}$.

Рассмотрим интервалы $[t_{i-1}, t_i)$, $1 \leq i \leq k+1$. Пусть $\mathcal{I}_j$ обозначает множество тех интервалов $[t_{i-1}, t_i)$, в которых может быть выполнена работа $J_j \in \mathcal{J}$.

Утверждение 6 *Пусть σ — оптимальное решение рассматриваемой задачи среди расписаний, в которых каждая работа J_j выполняется внутри одного из интервалов $I \in \mathcal{I}_j$. Тогда $E(\sigma) \leq 2^{\alpha-1}OPT$.*

Доказательство. Сначала покажем, что в любом допустимом расписании задачи $1|r_i, d_i|E$ нет работ, интервал выполнения которых включает два последовательных момента t_ℓ и $t_{\ell+1}$. Рассмотрим произвольную работу $J_j \in \mathcal{J}_\ell$, которая может выполняться более чем в двух интервалах, т.е. $|\mathcal{I}_j| \geq 2$. По определению имеем $r_j \leq t_\ell$ и $t_{\ell'} < d_j \leq t_{\ell'+1}$, где $\ell < \ell'$. Рассмотрим произвольный момент t_p, $\ell < p \leq \ell'$ такой, что $r_j < t_p \leq d_j$. Пусть $J_{j'} \in \mathcal{J}_p$ работа, определяющая момент t_p, т.е. $t_p = d_{j'} = \min\{d_i : J_i \in \mathcal{J} \setminus \mathcal{J}_1 \bigcup \cdots \bigcup \mathcal{J}_{p-1}\}$. По определению для J'_j выполнено $t_{p-1} < r_{j'} \leq t_p$. Следовательно, работа J_j не может выполняться в допустимом расписании без прерываний с момента t_{p-1} до момента t_p, так как в интервале $[t_{p-1}, t_p]$ должна быть выполнена работа $J_{j'}$.

По оптимальному расписанию σ^* построим расписание σ', в котором каждая работа J_j выполняется внутри одного из интервалов $I \in \mathcal{I}_j$. Для этого рассмотрим произвольную работу $J_j \in \mathcal{J}$, которая в расписании σ^* выполняется в двух интервалах, например, $(t_{\ell-1}, t_\ell]$ и $(t_\ell, t_{\ell+1}]$. Пусть $e_{j,\ell}$ и $e_{j,\ell+1}$ время выполнения J_j в $(t_{\ell-1}, t_\ell]$ и $(t_\ell, t_{\ell+1}]$ соответственно. Без ограничения общности предположим, что $e_{j,\ell} \geq e_{j,\ell+1}$. В расписании σ' поместим работу J_j целиком в интервал $(t_{\ell-1}, t_\ell]$ и положим ее время выполнения равным $\frac{(e_{j,\ell}+e_{j,\ell+1})}{2}$, увеличив скорость ее выполнения в два раза. Расход энергии на выполнение работы J_j в расписании σ^* равен $(e_{j,\ell} + e_{j,\ell+1})S_j^\alpha$, а в расписании σ' равен $\frac{(e_{j,\ell}+e_{j,\ell+1})}{2}(2S_j)^\alpha$. Повторим описанную процедуру для всех работ, выполняемых в расписании σ^* в двух интервалах. Суммируя соотношение на расход энергии по всем таким работам, получим $E(\sigma') \leq 2^{\alpha-1}OPT$. Так как σ оптимальное расписание, в котором каждая работа J_j выполняется внутри одного из интервалов $I \in \mathcal{I}_j$, то $E(\sigma) \leq E(\sigma') \leq 2^{\alpha-1}OPT$. □

По произвольному примеру задачи $1|r_i, d_i|E$ построим пример задачи $P|pmtn*, r_{ij}, d_{ij}, W_{ij}, \alpha_i|E$. Построим интервалы $[t_{i-1}, t_i]$, $1 \leq i \leq k+1$, способом, описанным вначале данного параграфа. Каждому интервалу $[t_{i-1}, t_i]$, $1 \leq i \leq k+1$,

сопоставим машину M_i. Для каждой работы $J_j \in \mathcal{J}$, которая может выполняться в интервале $[t_{i-1}, t_i]$, $1 \le i \le k+1$, положим $r_{i,j} = \max\{0, r_j - t_{i-1}\}$, $d_{i,j} = \min\{t_i - t_{i-1}, d_j - t_{i-1}\}$ и $w_{i,j} = w_j$. Для каждой машины M_i, $1 \le i \le k+1$, положим $\alpha_i = \alpha$.

Применим к полученному примеру алгоритм 3.5. Вообще говоря, алгоритм 3.5 найдет допустимое относительно директивных сроков расписание с прерываниями работ по интервалам $[t_{i-1}, t_i)$. Перестроим расписание, найденное алгоритмом 3.5 в допустимое расписание без прерываний. Заметим, что для каждой машины M_i и для каждой доступной на ней работы либо $r_{i,j} = 0$, либо $d_{i,j} = t_i - t_{i-1}$. Таким образом, все работы на этой машине имеют согласованные времена поступления и директивные сроки. Упорядочивая работы по неубыванию директивных сроков и по возрастанию времен поступления в случае, когда директивные сроки совпадают, получим допустимое расписание без прерываний. Окончательно получаем следующий результат.

Теорема 16 *Для задачи* $1|r_i, d_i|E$ *существует* $(2^{\alpha-1}((1+\frac{\varepsilon}{1-\varepsilon})(1+\frac{2}{n-2}))^{\alpha}\tilde{B}_{\alpha})$*-приближенный алгоритм.*

3.2.4 Неоднородная задача на параллельных машинах с прерываниями и перемещениями работ

В данном параграфе рассмотрим задачу $P|pmtn, r_{ij}, d_{ij}, W_{ij}, \alpha_i|E$. Множество работ должно быть выполнено на параллельных неоднородных машинах. Выполнение работы может быть прервано и продолжено позднее на любой машине. Пусть в некотором расписании σ на машине M_i выполнено x единиц работы J_j. Величину $\rho_{ij}(\sigma) = x/W_{i,j}$ назовем долей работы J_j на машине M_i в расписании σ. Расписание допустимо, если сумма долей каждой работы равна 1.

Для рассматриваемой задачи предлагается вполне полиномиальный приближенный алгоритм с абсолютной погрешностью ε для любого фиксированного $\varepsilon > 0$. Алгоритм основан на решении соответствующей задачи линейного программирования с экспоненциальным числом переменных и полиномиальным числом ограничений.

Конфигурацией c назовем назначение n_{c}, $0 \le n_{\mathsf{c}} \le m$, работ на m машин такое, что каждая машина обслуживает не более одной работы, с заданными скоростями обслуживания каждой работы. Обозначим через $\mathcal{C}$ множество всех возможных конфигураций. Очевидно, что расписание допустимо, если в каждый момент времени реализуется не более одной конфигурации. Мощность множества $\mathcal{C}$ неограничена, так как скорость может принимать любое положительное действительное значение. Следующий результат позволяет ограничиться конечным числом возможных скоростей за счет некоторой потери точности.

Лемма 21 *Существует допустимое расписание, в котором число различных скоростей, используемых машинами в течение всего интервала выполнения работ, конечно, и суммарный расход энергии не превосходит величины* $OPT + \varepsilon$ *для любого фиксированного* $\varepsilon > 0$.

Доказательство. Определим нижнюю и верхнюю границы на скорость работы машин в оптимальном расписании. Учитывая, что в оптимальном расписании каждая машина выполняет любую работу или ее часть с постоянной скоростью, в качестве нижней оценки на скорость можно взять величину

$$S_{LB} = \min_{J_j \in \mathcal{J}} \{ \frac{\min_{i \in \mathcal{M}} \{W_{i,j}\}}{\sum_{M_i \in \mathcal{M}} (d_{i,j} - r_{i,j})} \}.$$

В свою очередь, в качестве верней оценки можно рассмотреть скорость, которая необходима для выполнения общего объема всех работ в минимальном по длине интервале, то есть

$$S_{UB} = \max_{M_i \in P} \{ \frac{\sum_{J_j \in J} W_{i,j}}{min_{J_j \in J} (d_{i,j} - r_{i,j})} \}.$$

В интервале между величинами S_{LB} и S_{UB} для заданной константы $\delta > 0$ выберем дискретное множество значений $\mathcal{S}_\delta = \{(1+\delta)S_{LB}, (1+\delta)^2 S_{LB}, \ldots, (1+\delta)^k S_{LB}\}$, где k — наименьшее целое такое, что $(1+\delta)^k S_{LB} \geq S_{UB}$. В дальнейшем будем рассматривать расписания, в которых скорости машин могут принимать только перечисленное выше множество значений. Заметим, что $k = \lceil \log_{1+\delta} \frac{S_{UB}}{S_{LB}} \rceil$, и число различных возможных скоростей ограничено полиномом от размера входа и величины $1/\log_2(1+\delta)$.

Рассмотрим произвольное оптимальное расписание σ^* задачи $P|pmtn, r_{ij}, d_{ij}, W_{ij}, \alpha_i|E$ с произвольным набором скоростей. Пусть расписание σ получается из расписания σ^* округлением скоростей машин в каждый момент времени до ближайшего дискретного значения. Тогда расход энергии машины M_i увеличится не более чем в $(1+\delta)^{\alpha_i}$ раз и, следовательно, $E(\sigma) \leq (1+\delta)^\alpha OPT$. Выбирая $\delta = (1 + \frac{\varepsilon}{OPT})^{1/\alpha} - 1$, получим $E(\sigma) \leq OPT + \varepsilon$. □

Заметим, что для $\delta = (1 + \frac{\varepsilon}{OPT})^{1/\alpha} - 1$ число различных возможных скоростей экспоненциально от размера входа и $1/\varepsilon$.

Пусть $t_0 < t_1 < \ldots < t_\ell$ — упорядоченная по времени последовательность времен поступления и директивных сроков всех работ на всех машинах. Множество интервалов $(t_{i-1}, t_i]$, $1 \leq i \leq \ell$ обозначим через $\mathcal{I}$.

Для каждого $I \in \mathcal{I}$ и $\mathsf{c} \in \mathcal{C}$ введем переменную $x_{I,\mathsf{c}}$. Значение переменной $x_{I,\mathsf{c}}$ равно суммарному времени, в течение которого машины выполняют работы в интервале I согласно конфигурации c. Соответственно через $E_{I,\mathsf{c}} = \sum_{j \in (I,\mathsf{c})} S_{j,\mathsf{c}}^{\alpha_{i(j,\mathsf{c})}}$ обозначим мгновенную суммарную мощность, вырабатываемую машинами в конфигурации c, и через $S_{j,\mathsf{c}}$ — скорость выполнения работы J_j в данной конфигурации. Дополнительно для удобства записи обозначим через (I, c) множество работ, выполняемых в интервале I согласно конфигурации c, и через $i(j, \mathsf{c})$ индекс машины, на которой выполняется работа J_j в конфигурации c. Рассмотрим следующую задачу линейного программирования.

Задача LP_3

$$\min \sum_{I\in\mathcal{I},\mathsf{c}\in\mathcal{C}} E_{I,\mathsf{c}} \cdot x_{I,\mathsf{c}}$$

$$\sum_{\mathsf{c}\in\mathcal{C}} x_{I,\mathsf{c}} \le |I|, \qquad \forall I \in \mathcal{I}, \tag{3.18}$$

$$\sum_{I,\mathsf{c}:\ j\in(I,\mathsf{c})} \frac{S_{j,\mathsf{c}}}{w_{i(j,\mathsf{c}),j}} x_{I,\mathsf{c}} \ge 1, \qquad \forall j \in \mathcal{J}, \tag{3.19}$$

$$x_{I,\mathsf{c}} \ge 0, \qquad \forall I \in \mathcal{I},\ \mathsf{c} \in \mathcal{C}.$$

Рассмотрим расписание в интервале I, в котором конфигурации, назначенные в этот интервал, реализуются в произвольном порядке. Расписание допустимо, если суммарная длина всех конфигураций не превышает длину интервала. Выполнение данного условия следует из ограничения (3.18). Ограничение (3.19) гарантирует, что все работы будут полностью выполнены. Таким образом, оптимальное решение задачи LP_3 задает допустимое расписание σ задачи $P|pmtn, r_{ij}, d_{ij}, W_{ij}, \alpha_i|E$. Более того, из леммы 21 вытекает, что суммарный расход энергии в σ не превосходит величины $OPT + \varepsilon$.

Как уже отмечалось выше, число различных конфигураций экспоненциально зависит от размера входа и величины $1/\varepsilon$ и, следовательно, задача LP_3 имеет экспоненциальное число переменных. При этом число ограничений оценивается линейной функцией от числа работ. Рассмотрим двойственную к ней задачу LP_4.

Задача LP_4

$$\max \sum_{j\in\mathcal{J}} \lambda_j - \sum_{I\in\mathcal{I}} \mu_I |I|$$

$$\sum_{j\in(I,\mathsf{c})} \frac{S_{j,\mathsf{c}}}{w_{i(j,\mathsf{c}),j}} \lambda_j - \mu_I \le E_{I,\mathsf{c}}, \qquad \forall I \in \mathcal{I},\ \mathsf{c} \in \mathcal{C},$$

$$\mu_I, \lambda_j \ge 0, \qquad \forall I \in \mathcal{I},\ J_j \in \mathcal{J}.$$

Число переменных в двойственной задаче LP_4 ограничено линейной функцией от числа работ, однако она имеет экспоненциальное число ограничений. В [102] описывается схема решения подобных задач с использованием метода эллипсоидов [25, 15]. Напомним, что метод эллипсоидов состоит из последовательности итераций. На каждой итерации имеется эллипсоид, содержащий некоторое решение системы линейных уравнений задачи LP_4 и уравнения, определяющего верхнюю границу на величину оптимального решения. Итерация состоит в замене текущего эллипсоида меньшим, который также содержит некоторое решение, если оно существует. Для построения меньшего эллипсоида для некоторой известной на каждой итерации точки, которая лежит внутри эллипсоида, необходимо проверить, что данная точка является допустимым решением для заданной системы неравенств, или найти неравенство,

которому она не удовлетворяет. В случае, когда число линейных неравенств ограничено полиномом от размера входа данной задачи, вопрос проверки допустимости заданного решения решается простым перебором. Для решения задачи линейного программирования с экспоненциальным числом ограничений требуется построить отделяющий оракул, который для заданного произвольного решения задачи LP_4 за время, ограниченное полиномом от размера входа и величины $1/\varepsilon$, либо определяет, что оно является допустимым, либо находит ограничение, которое нарушается. Если такой оракул существует, то задача LP_4 разрешима за время, ограниченное полиномом от размера входа и величины $1/\varepsilon$, если выполняются два дополнительных технических условия [102]. Первое условие гласит, что значения всех двойственных переменных должны быть ограничены сверху некоторым рациональным значением R. Второе условие требует существования допустимой точки (решения) такой, что любая точка в шаре радиуса r является допустимой. При выполнении этих условий время работы метода эллипсоидов будет ограничено полиномиально от величины $\log \frac{R}{r}$.

В нашем случае первое условие следует из факта, что решение задачи должно быть вершиной соответствующего полиэдра, поскольку значение оптимального решения ограничено. Следовательно, R можно ограничить полиномом от параметров задачи. Перейдем ко второму условию. Положим $\lambda_j = 1$ для всех $j \in \mathcal{J}$ и выберем значения переменных μ_I достаточно большими, чтобы выполнялись неравенства $-\mu_I + 1 \leq \min_{\mathsf{c}\in\mathcal{C}}(E_{I,\mathsf{c}}) - 2\sum_{j\in(I,\mathsf{c})} \frac{S_{j,\mathsf{c}}}{w_{i(j,\mathsf{c}),j}}$. Получим точку (λ, μ) такую, что любое решение в шаре радиуса 1 является допустимым решением двойственной задачи LP_4. В свою очередь, построенное методом эллипсоидов[25] оптимальное решение задачи LP_4 позволяет найти оптимальное решение прямой задачи LP_3 за полиномиальное время, так как значения прямых переменных, соответствующих двойственным ограничениям, которые не были нарушены в ходе решения двойственной задачи, равны нулю.

Вернемся к построению отделяющего оракула для задачи LP_4. Для каждого $I \in \mathcal{I}$ требуется проверить, существует ли конфигурация c, на которой нарушается ограничение (3.20), т.е. для каждого интервала $I \in \mathcal{I}$ достаточно найти минимальное значение величины $E_{I,\mathsf{c}} - \sum_{j\in(I,\mathsf{c})} \frac{S_{j,\mathsf{c}}}{w_{i(j,\mathsf{c}),j}}\lambda_j$ на множестве всех возможных конфигураций. Если найденное значение меньше чем $-\mu_I$, то данное ограничение нарушается. В противном случае все ограничения для выбранного интервала $I \in \mathcal{I}$ выполнены.

Подставляя значение величины $E_{I,\mathsf{c}}$ в верхнее выражение, получим, что требуется минимизировать значение величины $\sum_{J_j\in(I,\mathsf{c})}(S_{j,\mathsf{c}}^{\alpha_{i(j,\mathsf{c})}} - \frac{S_{j,\mathsf{c}}}{w_{i(j,\mathsf{c}),j}}\lambda_j)$. Величина каждого слагаемого в последнем выражении зависит от выбора скорости, с которой выполняется работа $J_j \in (I, \mathsf{c})$. Для каждой работы $J_j \in \mathcal{J}$, задействованной в интервале I на машине M_i, рассмотрим функцию $f_{ji}(x) = x^{\alpha_{i(j,\mathsf{c})}} - \frac{x}{w_{i(j,\mathsf{c}),j}}\lambda_j$, которая достигает минимума при $x = \left(\frac{\lambda_j}{\alpha_{i(j,\mathsf{c})}\cdot w_{i(j,\mathsf{c}),j}}\right)^{1/(\alpha_{i(j,\mathsf{c})}-1)}$. В силу выпуклости функции $f_{ji}(x)$ получим, что минимум каждого слагаемого достигается на одном из двух дискретных значений скорости из множества $\mathcal{S}_\delta$, наиболее близком к значению $\left(\frac{\lambda_j}{\alpha_{i(j,\mathsf{c})}\cdot w_{i(j,\mathsf{c}),j}}\right)^{1/(\alpha_{i(j,\mathsf{c})}-1)}$, которое будем обозначать через $v_{i(j,\mathsf{c}),j}$. Следовательно, для

каждого интервала $I \in \mathcal{I}$ требуется найти конфигурацию c, которая минимизирует величину $\sum_{J_j \in (I,\mathsf{c})} (v_{i(j,\mathsf{c}),j}^{\alpha_{i(j,\mathsf{c})}} - \frac{v_{i(j,\mathsf{c}),j}}{w_{i(j,\mathsf{c}),j}} \lambda_j)$.

Напомним, что каждая конфигурация соответствует назначению n_c, $0 \le n_\mathsf{c} \le m$, работ на m машин и выбору скорости обслуживания каждой работы. Тогда минимизация величины $\sum_{J_j \in (I,\mathsf{c})} (v_{i(j,\mathsf{c}),j}^{\alpha_{i(j,\mathsf{c})}} - \frac{v_{i(j,\mathsf{c}),j}}{w_{i(j,\mathsf{c}),j}} \lambda_j)$ сводится к нахождению максимального взвешенного паросочетания в следующем двудольном графе $G = (A \cup B, E)$. Множество A взаимнооднозначно соответствует подмножеству работ $\mathcal{J}$, задействованных в интервале I, и множество B взаимнооднозначно соответствует множеству машин. Если работа J_j может выполняться на машине M_i в интервале I, то между соответствующими им вершинами есть ребро веса $-(v_{i,j}^{\alpha_i} - \frac{v_{i,j}}{w_{i,j}} \lambda_j)$. Каждое паросочетание в графе G определяет допустимую конфигурацию n_c, и, следовательно, проверка существования конфигурации c, на которой нарушается ограничение (3.20), может быть осуществлена за время, ограниченное полиномом от размера входа задачи и величины $1/\varepsilon$.

В результате получаем следующую теорему.

Теорема 17 *Для любого $\varepsilon > 0$ и любого примера задачи $P|pmtn, r_{ij}, d_{ij}, W_{ij}, \alpha_i|E$ за время, полиномиальное от размера входа и величины $1/\varepsilon$ можно найти решение со значением целевой функции, не превосходящим $OPT + \varepsilon$.*

3.2.5 Цеховая задача рабочего типа с прерываниями операций

В данном параграфе рассмотрим задачу $J|pmtn, r_{i,j}, d_{i,j}, W_{i,j}, \alpha_i|E$ минимизации энергии при построении допустимого расписания многостадийных работ в системе с различными фиксированными маршрутами. Такие задачи принято называть цеховыми задачами рабочего типа.

В многостадийных или цеховых задачах теории расписаний процесс обслуживания каждой работы состоит из нескольких последовательных стадий, на каждой из которых эта работа выполняется на той или иной машине. Таким образом, заданы множество работ $\mathcal{J} = \{J_1, \ldots, J_n\}$ и множество машин $\mathcal{M}$. Каждая работа J_j состоит из μ_j операций $O_{1j}, \ldots O_{\mu_j j}$. Каждая операция O_{ij} должна быть выполнена на определенной машине $M_{ij} \in \mathcal{M}$, при этом две операции одной работы не могут выполняться одновременно. В цеховой задаче рабочего типа порядок выполнения операций одной работы фиксирован, т.е. для каждой работы $J_j \in \mathcal{J}$ и $1 \le k \le \mu_j - 1$ задано отношение предшествования $O_{k,j} \to O_{k+1,j}$, означающее, что выполнение операции $O_{k+1,j}$ может начаться только после завершения операции $O_{k,j}$. Обозначим через μ число всех операций, т.е. $\mu = \sum\limits_{J_j \in \mathcal{J}} \mu_j$. Для каждой операции $O_{k,j}$ задан объем работ $w_{k,j}$. Заметим, что возможна ситуация, когда более одной операции одной и той же работы должно быть выполнено на некоторой машине. Для каждой операции $O_{k,j}$ заданы ее время поступления $r_{k,j}$ и директивный срок $d_{k,j}$. Без ограничения общности предположим, что для каждой работы $J_j \in \mathcal{J}$ ее времена поступления и директивные сроки согласованы с порядком выполнения операций, т.е. $r_{1,j} \le r_{2,j} \le \ldots \le r_{\mu_j,j}$

и $d_{1,j} \leq d_{2,j} \leq \ldots \leq d_{\mu_j,j}$. В процессе выполнения операций разрешены прерывания. Требуется найти допустимое относительно времен поступления и директивных сроков расписание, минимизирующее расход энергии.

Для рассматриваемой задачи построим приближенный алгоритм, основанный на решении соответствующей задачи линейного программирования с экспоненциальным числом переменных и полиномиальным числом ограничений. Для представления задачи $J|pmtn, r_{i,j}, d_{i,j}, W_{i,j}, \alpha_i|E$ как задачи целочисленного линейного программирования ограничимся расписаниями, в которых все операции и их фрагменты начинают и заканчивают обслуживание в специальные дискретные моменты времени.

Пусть $t_0, t_1, \ldots, t_\tau$ — упорядоченная по времени последовательность моментов, соответствующих временам поступления и директивным срокам работ, такая, что каждому времени поступления или директивному сроку соответствует один из моментов t_ℓ. Пусть $I_\ell = [t_{\ell-1}, t_\ell)$, для $1 \leq \ell \leq \tau$, и обозначим через $|I_\ell|$ длину интервала I_ℓ.

Лемма 22 *Существует допустимое расписание, расход энергии в котором не превышает величины* $(1+\varepsilon)^{\alpha-1} \cdot OPT$, *и каждый фрагмент каждой операции* $O_{k,j}$, $J_j \in \mathcal{J}$ *и* $1 \leq k \leq \mu_j$, *выполняемый в интервале* I_ℓ, $1 \leq \ell \leq \tau$, *начинается и заканчивается в моменты* $t_{\ell-1} + r\frac{\varepsilon}{\mu(1+\varepsilon)}|I_\ell|$, *где* $r \geq 0$ *некоторое целое число.*

Доказательство. Рассмотрим произвольное расписание σ и произвольный интервал I_ℓ, $1 \leq \ell \leq \tau$. Определим последовательность моментов $q_0 = t_{\ell-1} < q_1 < q_2 < \cdots < q_w = t_\ell$, совпадающих с множеством времен начала или завершения фрагментов операций на любой из машин в интервале I_ℓ в σ. При этом интервал I_ℓ разобьется на интервалы $[q_{p-1}, q_p)$ для $1 \leq p \leq w$, которые будем называть *слайсами.*

Заметим, что расписание σ в интервале I_ℓ можно рассматривать как допустимое расписание в задаче $J|pmtn|C_{\max}$ с заданными длительностями операций и длиной расписания не больше $|I_\ell|$. В [51] доказано, что для любого примера такой задачи существует оптимальное расписание с не более чем μ слайсами, то есть $w \leq \mu$.

Пусть σ^* — оптимальное расписание, в котором каждый интервал I_ℓ, $1 \leq \ell \leq \tau$ содержит не более μ слайсов. Создадим в каждом интервале I_ℓ промежуток простоя длины $\frac{\varepsilon}{1+\varepsilon}|I_\ell|$, в котором не выполняется ни одна операция. Для этого увеличим скорость каждой машины в каждом слайсе в $1+\varepsilon$ раз. Соответственно, расход энергии также увеличится в $(1+\varepsilon)^\alpha$ раз. Затем округлим длину каждого слайса до ближайшего значения $r\frac{\varepsilon}{\mu(1+\varepsilon)}|I_\ell|$. При этом длина каждого слайса увеличится не более чем на $\frac{\varepsilon}{\mu(1+\varepsilon)}|I_\ell|$. Так как число слайсов не превосходит μ, общее время работы машин в интервале I_ℓ увеличится не более чем на $\mu(\frac{\varepsilon}{\mu(1+\varepsilon)}|I_\ell|) = \frac{\varepsilon}{1+\varepsilon}|I_\ell|$ и не превысит длины созданного промежутка простоя. Таким образом, полученное расписание является допустимым и удовлетворяет условиям леммы. □

Моменты времени вида $t_{\ell-1} + r\frac{\varepsilon}{\mu(1+\varepsilon)}|I_\ell|$, введенные в формулировке леммы 22, будем называть *ключевыми.*

Лемма 23 *Существует допустимое расписание, расход энергии в котором не превышает величины* $(1+\varepsilon)^{\alpha-1}(1+\frac{2}{\mu-2})^{\alpha-1}(1+\frac{\varepsilon}{1-\varepsilon})^{\alpha-1}\cdot OPT$ *и для каждой операции* $O_{k,j}$, $J_j \in \mathcal{J}$ *и* $1 \le k \le \mu_j$, *существуют два ключевых момента* $b_{k,j}$ *и* $c_{k,j}$ *такие, что каждый фрагмент операции* $O_{k,j}$ *начинается и заканчивается в моменты* $b_{k,j} + h\frac{\varepsilon}{\mu^3}(c_{k,j} - b_{k,j})$ *в интервале* $[b_{k,j}, c_{k,j}]$, *где* $h \ge 0$ *некоторое целое число.*

Доказательство. Рассмотрим расписание σ, удовлетворяющее условиям леммы 22. Согласно этой лемме каждый интервал I_ℓ, $1 \le \ell \le \tau$, разбит на промежутки равной длины, число которых ограничено полиномом от μ и $1/\varepsilon$. В каждом из таких промежутков каждая операция $O_{k,j}$, $J_j \in \mathcal{J}$ и $1 \le k \le \mu_j$ либо выполняется в течение всего промежутка, либо не выполняется вовсе. Пусть $b_{k,j}$ и $c_{k,j}$ — момент начала выполнения первого фрагмента и момент завершения выполнения последнего фрагмента операции $O_{k,j}$ в расписании σ.

Сначала преобразуем расписание σ в расписание σ', в котором каждая операция $O_{k,j}$, $J_j \in \mathcal{J}$ и $1 \le k \le \mu_j$, выполняется по крайней мере в течение $\frac{\varepsilon}{\mu}(c_{k,j} - b_{k,j})$ единиц времени. Для этого в каждом промежутке ω увеличим скорость машины в $1+\frac{\varepsilon}{1-\varepsilon}$ раз и, соответственно, расход энергии — в $(1+\frac{\varepsilon}{1-\varepsilon})^{\alpha-1}$ раза. Обозначим полученное расписание через $\bar{\sigma}$. Для каждой операции $O_{k,j}$, $J_j \in \mathcal{J}$ и $1 \le k \le \mu_j$, выделим интервал длины $\frac{\varepsilon|s|}{\mu}$ в каждом промежутке ω в $(b_{k,j}, c_{k,j}]$. В расписании σ' уменьшим скорость выполнения операции $O_{k,j}$, назначив ее в те интервалы, в которых она выполнялась в расписании $\bar{\sigma}$ и в зарезервированные для нее интервалы длины $\frac{\varepsilon|s|}{\mu}$. Получим, что время выполнения операции $O_{k,j}$ в расписании σ' не меньше $\frac{\varepsilon}{\mu}(c_{k,j} - b_{k,j})$. Назовем операцию $O_{k,j}$ *доступной* в момент времени t, если $t \ge b_{k,j}$. На каждой машине упорядочим операции по невозрастанию величин $c_{k,j}$ и построим расписание операций с прерываниями по правилу: в каждый момент выполняй доступную операцию с наименьшим индексом. Получим допустимое расписание с не больше чем μ прерываниями, так как выполняющаяся операция может быть прервана только в момент, когда другая операция стала доступной.

Затем трансформируем расписание σ' в расписание σ'', которое удовлетворяет условиям леммы. Для каждой операции $O_{k,j}$, $J_j \in \mathcal{J}$ и $1 \le k \le \mu_j$, разделим интервал $(b_{k,j}, c_{k,j}]$ на промежутки $(b_{k,j} + h\frac{\varepsilon}{\mu^3}(c_{k,j} - b_{k,j}), b_{k,j} + (h+1)\frac{\varepsilon}{\mu^3}(c_{k,j} - b_{k,j})]$, где $h \ge 0$ некоторое целое число. Длина каждого нового промежутка равна $\frac{\varepsilon}{\mu^3}(c_{k,j} - b_{k,j})$. Поскольку время выполнения операции $O_{k,j}$ по крайней мере $\frac{\varepsilon}{\mu}(c_{k,j} - b_{k,j})$, она должна обрабатываться не менее чем в μ^2 промежутках. Так как число прерываний каждой операции не больше μ, то по крайней мере $\mu^2 - 2\mu$ из них полностью заняты операцией $O_{k,j}$. Разместим в расписании σ'' операцию $O_{k,j}$ в те интервалы, которые она занимала целиком, увеличив скорость ее выполнения в $1 + \frac{2}{\mu-2}$ раза. Принимая во внимание лемму 22 и оценку на увеличение расхода энергии в расписании σ' по отношению к оптимальному расписанию, получим утверждение леммы. □

Конфигурацией c назовем расписание одной работы, то есть допустимое расписание всех ее операций, удовлетворяющее условиям лемм 22 и 23. Более точно, отдельно для каждой работы J_j конфигурация определяет моменты начала и завер-

шения ее операций, удовлетворяющие условиям леммы 22, и для каждой операции — множество промежутков, в которых она выполняется, удовлетворяющее условиям леммы 23. Обозначим через $\mathcal{C}_j$ множество всех возможных конфигураций для работы $J_j \in \mathcal{J}$.

Для каждой машины M_i рассмотрим все временные точки вида $b_{k,j}+h\frac{\varepsilon}{\mu^3}(c_{k,j}-b_{k,j})$ всех операций, выполняемых на этой машине. Пусть $t_{i,1}, t_{i,2}, \ldots, t_{i,\ell_i}$ — упорядоченная последовательность этих точек. Рассмотрим интервалы $(t_{i,p}, t_{i,p+1}]$, $1 \le p \le \ell_i - 1$. В расписании, удовлетворяющем леммам 22 и 23, в каждом таком интервале либо выполняется одна работа, либо машина простаивает. Обозначим через $\mathcal{I}$ множество всех таких интервалов на всех машинах. Из лемм 22 и 23 следует, что размер множества $\mathcal{I}$ ограничен полиномом от размера входа и величины $1/\varepsilon$.

Заметим, что если для работы J_j задана некоторая конфигурация c, то можно посчитать расход энергии $E_{j,\mathsf{c}}$ на выполнение работы J_j. Для удобства будем писать $I \in (j,\mathsf{c})$, если в интервале $I \in \mathcal{I}$ выполняется операция работы J_j согласно конфигурации c, т.е. существует операция $O_{k,j}$, два момента $b_{k,j}$ и $c_{k,j}$, и промежуток $(b_{k,j} + h\frac{\varepsilon}{\mu^3}(c_{k,j} - b_{k,j}), b_{k,j} + (h+1)\frac{\varepsilon}{\mu^3}(c_{k,j} - b_{k,j})]$ в c, который содержит I. С учетом введенных обозначений задачу $J|pmtn, r_{i,j}, d_{i,j}, W_{i,j}, \alpha_i|E$ можно записать как задачу целочисленного линейного программирования.

Задача ILP_5

$$\min \sum_{j,\mathsf{c}} E_{j,\mathsf{c}} \cdot x_{j,\mathsf{c}},$$

$$\sum_{\mathsf{c}} x_{j,\mathsf{c}} \ge 1, \qquad \forall J_j \in \mathcal{J}, \tag{3.20}$$

$$\sum_{\mathsf{c}\in\mathcal{C}_j : I\in(j,\mathsf{c})} x_{j,\mathsf{c}} \le 1, \qquad \forall I \in \mathcal{I}, \tag{3.21}$$

$$x_{j,\mathsf{c}} \in \{0,1\}, \qquad \forall J_j \in \mathcal{J}, \mathsf{c} \in \mathcal{C}_j. \tag{3.22}$$

Ограничения (3.20) совместно с целевой функцией гарантируют, что каждая работа выполняется согласно одной конфигурации. Ограничения (3.21) означают, что не более одной работы выполняется в каждом интервале $I \in \mathcal{I}$. Рассмотрим задачу LP_5, полученную из задачи ILP_5 заменой ограничения (3.22) на ограничение $x_{j,c} \ge 0$, для всех $J_j \in \mathcal{J}$ и $\mathsf{c} \in \mathcal{C}_j$.

Как и в параграфе 3.2.2, преобразуем решение задачи LP_5 в допустимое решение задачи $J|pmtn, r_{i,j}, d_{i,j}, W_{i,j}, \alpha_i|E$. Выберем для каждой работы одну ее конфигурацию c случайно с вероятностью, пропорциональной $x_{i,j,\mathsf{c}}$. Напомним, что каждая конфигурация задает для работы множество интервалов, в которых она выполняется. При этом в процессе выбора в один интервал может попасть несколько работ. В таком случае увеличим скорость выполнения работ внутри интервала в необходимое число раз так, чтобы выполнить все работы в течение этого интервала. Формальное описание данной процедуры представлено ниже в алгоритме 3.6.

На первом шаге алгоритма требуется решить задачу линейного программирования LP_5. Число ограничений в задаче LP_5 полиномиально от размера входа и

Алгоритм 3.6

1: Решить задачу линейного программирования LP_5.
2: Для каждой работы $J_j \in \mathcal{J}$, случайно выбрать одну конфигурацию c с вероятностью $x_{j,\mathsf{c}}$.
3: Для каждой операции $O_{j,k}$ вычислить скорость ее выполнения $S_{j,k}$ согласно конфигурации, выбранной на шаге 2.
4: Для каждой машины M_i и интервала I определить множество операций $OPERATION(M_i, I)$, которые должны быть выполнены на машине M_i в интервале I согласно выбору на шаге 2.
5: Выполнить все работы каждого множества $OPERATION(M_i, I)$ со скоростью $\sum_{O_{j,k} \in OPERATION(M_i,I)} S_{j,k}$ на машине M_i в интервале I.

величины $1/\varepsilon$, однако в ней экспоненциальное число переменных. Поэтому для ее решения будем использовать схему с применением метода эллипсоидов, описанную в предыдущем параграфе. Сначала построим двойственную к задаче LP_5 задачу LP_6.

Задача LP_6

$$\min \sum_j \lambda_j - \sum_I \kappa_I,$$

$$\lambda_j - \sum_{I \in (j,\mathsf{c})} \kappa_I \le E_{j,\mathsf{c}}, \qquad \forall J_j \in \mathcal{J}, \mathsf{c} \in \mathcal{C}_j, \tag{3.23}$$

$$\lambda_j, \kappa_I \ge 0. \tag{3.24}$$

Рассмотрим произвольное решение (λ_j, κ_I) задачи LP_6 и покажем, как построить отделяющий оракул для этой задачи. Для каждой работы $J_j \in \mathcal{J}$ найдем конфигурацию c, на которой выражение $E_{j,\mathsf{c}} + \sum_{I \in (j,\mathsf{c})} \kappa_I$ принимает минимальное значение. Если $\min_{\mathsf{c}}\{E_{j,\mathsf{c}} + \sum_{I \in (j,\mathsf{c})} \kappa_I\} < \lambda_j$, то для найденных j и c ограничение (3.23) нарушается, в противном случае полученное решение является допустимым.

Для нахождения конфигурации, минимизирующей значение выражения $E_{j,\mathsf{c}} + \sum_{I \in (j,\mathsf{c})} \kappa_I$, построим алгоритм динамического программирования. Обозначим через $J_{j(k)}$ работу, которая состоит из первых k операций работы J_j, т.е. $O_{1,j}, O_{2,j}, \ldots, O_{k,j}$, и через $\mathsf{c}_{k,I}$ конфигурацию работы $J_{j(k)}$, в которой после завершения интервала I не выполняется ни одной операции. Пусть $B_{k,I} = \min_{\mathsf{c}_{k,I}}\{E_{j(k),\mathsf{c}_{k,I}} + \sum_{I' \in (j(k),\mathsf{c}_{k,I})} \kappa'_I\}$. Обозначим через $\bar{\mathcal{I}}$ множество интервалов, у которых правая граничная точка является ключевым моментом. Пусть $C_{k,\ell,I',I}$ — вклад в $B_{k,I}$ операции $O_{k,j}$, когда рассматриваются конфигурации, в которых операция $O_{k,j}$ выполняется ровно в ℓ промежутках между правой граничной точкой интервала $I' \in \bar{\mathcal{I}}$ и правой граничной точкой интервала $I \in \bar{\mathcal{I}}$. При этом правая граничная точка интервала $I' \in \bar{\mathcal{I}}$ определяет момент $b_{k,j}$, а правая граничная точка интервала $I \in \bar{\mathcal{I}}$ определяет момент $c_{j,k}$. Пусть $D_{k,I',I} = \min_\ell\{C_{k,\ell,I',I}\}$. Запись $I' < I$ обозначает, что интервал I' предшествует интервалу I. Тогда для интервалов из множества $\bar{\mathcal{I}}$ выполнено следующее соотношение

динамического программирования

$$B_{k,I} = \min_{I' \in \bar{\mathcal{I}} | I' < I} \{B_{k-1,I'} + D_{k,I',I}\}.$$

Для определения величины $D_{k,I',I}$ требуется вычислить l величин $C_{k,\ell,I',I}$, которые могут быть найдены за полиномиальное время для любых k, ℓ, I' и I. Действительно, так как длины всех промежутков выполнения операции $O_{k,j}$ равны, то расход энергии не зависит от конкретных промежутков, включенных в конфигурацию. Следовательно, значение $E_{j,\mathsf{c}}$ фиксировано для заданного ℓ, и остается выбрать ℓ промежутков минимальной стоимости κ_I. Так как число промежутков ограничено полиномом от размера входа и величины $1/\varepsilon$, то это может быть сделано за полиномиальное время.

Таким образом, для каждой работы $J_j \in \mathcal{J}$ за полиномиальное время можно найти минимальное значение величины $E_{j,\mathsf{c}} + \sum_{I \in (j,\mathsf{c})} \kappa_I$ среди всех конфигураций $\mathsf{c} \in \mathcal{C}_j$, и, следовательно, существует полиномиальный отделяющий оракул для задачи LP_6. Применив алгоритм эллипсоидов[25], можно точно решить двойственную задачу LP_6, и по ее решению можно получить точное решение прямой задачи LP_5 за время, ограниченное от размера входа и величины $1/\varepsilon$.

Анализ точности расписания, получаемого алгоритмом 3.6, совпадает с анализом точности алгоритма 3.5.

Теорема 18 *Алгоритм 3.6 является* $(1+\varepsilon)^{\alpha-1}(1+\frac{2}{\mu-2})^{\alpha-1}(1+\frac{\varepsilon}{1-\varepsilon})^{\alpha-1}\tilde{B}_\alpha$*-приближенным алгоритмом для задачи* $J|pmtn, r_{ij}, d_{ij}, W_{ij}, \alpha_i|E$*. Его время работы ограничено полиномом от размера входа и величины* $1/\varepsilon$.

Глава 4

Задачи построения расписаний с задержками передачи данных

В этой главе рассматриваются задачи построения расписания работ на параллельных машинах с частичным порядком на множестве работ и задержками при передаче данных. В 1987 г. Райвард-Смит [147] ввел в рассмотрение однородную коммуникационную модель выполнения множества заданий параллельной программы. В этой модели множество работ $\mathcal{J} = \{J_1, \ldots, J_n\}$ должно быть выполнено на m идентичных параллельных машинах. Пусть $G = (V, E)$ – ориентированный ациклический граф, $V = \mathcal{J}$, а множество E задает частичный порядок на множестве $\mathcal{J}$. Далее в этой главе будем использовать символ V для обозначения множества работ. Для каждой дуги $(J_j, J_k) \in E$ задана задержка δ_{jk}. Если работы, связанные отношением предшествования, выполняются на разных машинах, то в расписании необходимо учесть время на передачу данных от одной машины к другой. Если обе работы выполняются на одной машине, то считается, что величина задержки несущественна, и она полагается равной нулю. Отметим, что процесс получения или отправки информации о работе J_j не препятствует обработке другой работы на этой машине. Таким образом, требуется найти золотую середину между двумя экстремальными решениями: выполнить все работы последовательно без задержек или использовать весь потенциал распараллеливания, но при этом потерять время на передачу данных от одной машины к другой. Эта модель интенсивно изучается, начиная с 90-х годов прошлого века.

В [108] показано, что задачи $P\infty|ct|C_{max}$ и $P\infty|ct|\sum C_j$ на неограниченном числе машин являются NP-трудными, даже если все задержки и длительности всех работ единичные. По трудности построения приближенных алгоритмов в задачах с задержками неформально можно выделить три подзадачи:

- задачи с единичными задержками и единичными длительностями работ, т.е. $\delta_{ij} = 1$ для всех $(J_i, J_j) \in E$ и $p_i = 1$ для всех $J_i \in V$;
- задачи с малыми задержками,

$$\max_{(J_i,J_j)\in E} \delta_{ij} \leq \min_{J_i \in V} p_i; \tag{4.1}$$

- задачи с большими задержками, $\min_{(J_i,J_j)\in E} \delta_{ij} \geq \max_{J_i\in V} p_i$.

Построение приближенного алгоритма с гарантированной оценкой точности, ограниченной константой, для последней задачи является одной из важных открытых проблем в теории расписаний. Далее в этой главе рассматриваются только две первые задачи. Для упрощения обозначений в задачах с малыми задержками во втором поле $\alpha|\beta|\gamma$-идентификатора будем использовать аббревиатуру *sct* (small communication time) и в задачах с единичными задержками — аббревиатуру *uct* (unit communication time). Например, задача минимизации длины расписания с малыми задержками на неограниченном числе параллельных машин будет обозначаться идентификатором $P\infty|sct|C_{max}$. Отметим, что для задач с малыми задержками на неограниченном числе параллельных машин любое активное расписание обладает следующим свойством.

Утверждение 7 *Пусть $C_j(\sigma)$ – время завершения работы J_j в расписании σ. Тогда $C_j(\sigma) \leq 2t$, где t – самый ранний момент завершения работы J_j по всем допустимым расписаниям.*

Как следствие, любое активное расписание является 2-приближенным алгоритмом для задач $P\infty|sct|C_{max}$ и $P\infty|sct|\sum w_jC_j$.

В [142] для задачи $P\infty|uct, p_j = 1, |C_{max}$ предложен 4/3-приближенный алгоритм. В [108] показано, что задача $P\infty|uct, p_j = 1|C_{max} \leq 6$ проверки существования расписания длины 6 является NP-полной. Таким образом, существование полиномиального ρ-приближенного алгоритма для задачи $P\infty|uct, p_j = 1|C_{max}$, с $\rho < 7/6$ влечет совпадение классов P и NP.

Задача $P|uct, p_j = 1|C_{max}$, в которой число машин ограничено и является частью входной информации, еще более трудна с вычислительной точки зрения. Как показано в [108], проверка существования расписания длины 4 в такой постановке является NP-полной задачей, что влечет несуществование для нее полиномиального ρ-приближенного алгоритма с $\rho < 5/4$ при условии несовпадения классов P и NP. Наилучший известный приближенный алгоритм с $\rho = (7/3 - 4)/(3m)$ предложен в [141].

Кроме критерия минимизации длины расписания, для задач с задержками рассматривались и другие классические критерии. Например, в [139] представлен 4/3-приближенный алгоритм для задачи $P\infty|uct, p_j = 1|\sum w_jC_j$, обобщающий результат из [142]. В той же статье представлен $10/3-4/(3m)$-приближенный алгоритм для задачи $P|uct, p_j = 1|\sum w_jC_j$.

Заметим, что упомянутые выше приближенные алгоритмы для задач минимизации длины расписания и минимизации взвешенной суммы моментов окончания работ основаны на использовании в качестве нижней оценки значения оптимальных решений задач линейного программирования фактически с одинаковым набором ограничений. В связи с этим возникает вопрос о возможности получения приближенного алгоритма, строящего хорошее приближенное решение для обоих критериев одновременно.

В разделе 4.1 представлены алгоритмы, которые строят расписание, ориентирующееся сразу на два критерия: минимизацию длины расписания и минимизацию взвешенной суммы моментов окончания работ с обобщением полученных результатов на задачу, в которой для каждой работы заданы не только время выполнения, но и время на доставку.

В разделе 4.2 рассматривается другое обобщение задач $P|sct|C_{max}$ и $P\infty|sct|C_{max}$. В [42] была введена иерархическая коммуникационная модель, в которой задержки не являются однородными. Машины разбиты на группы или кластеры. Передача данных между машинами одной группы осуществляется значительно быстрее, чем между двумя машинами из разных групп. Эта модель отражает иерархическую структуру современных компьютерных сетей и рабочих станций [35, 144]. Использование сетей из рабочих станций для параллельных вычислений возобновило интерес к параллельным вычислениям и также создало ряд новых алгоритмических проблем, связанных с использованием полной мощности механизма распараллеливания работ в таких системах. В разделе 4.2 будет приведен приближенный алгоритм для задачи с произвольными длительностями, малыми задержками передачи данных от одной группы машин к другим и дополнительным условием, что каждая группа состоит ровно из двух машин.

4.1 Одновременная минимизация длины расписания и взвешенной суммы моментов окончания работ

Рассмотрим следующую задачу теории расписаний. Множество работ $V = \{J_1, \ldots, J_n\}$ должно быть выполнено на m идентичных параллельных машинах. Пусть ациклический ориентированный граф $G = (V, E)$ задает частичный порядок на множестве работ V. Без ограничения общности предположим, что в графе G нет транзитивных замыканий. Для каждой работы $J_i \in V$ задана ее длительность p_i и неотрицательный вес w_i. Каждая дуга $(J_i, J_j) \in E$ ассоциируется с отношением предшествования между работами J_i и J_j, а ее вес δ_{ij} — с коммуникационной задержкой. В дальнейшем предположим, что задержки малы, т.е. $\max\limits_{(J_i,J_j)\in E} \delta_{ij} \leq \min\limits_{J_i \in V} p_i$. Таким образом, если работы J_i и J_j выполняются на одной машине, то работа J_j может стартовать непосредственно сразу после окончания работы J_i, т.е. в любом допустимом расписании должно выполняться $s_j \geq s_i + p_i$. Если работы J_i и J_j выполняются на разных машинах, то работа J_j не может начаться раньше времени $s_i + p_i + \delta_{ij}$. Работа J_i называется родителем работы J_j, а работа J_j — ребенком работы J_i. Множества всех детей и родителей работы J_i обозначим через $\Gamma^+(i)$ и $\Gamma^-(i)$ соответственно. Множество работ, у которых нет родителей, обозначим через Z, и множество работ, у которых нет детей, — через U. Напомним, что $C_{max}(\sigma) = \max_{J_i \in V} C_j(\sigma)$.

В этом разделе рассмотрим задачу $P\infty|sct|C_{max}, \sum w_j C_j$, в которой требуется найти "хорошее" расписание относительно двух критериев C_{max} и $\sum w_j C_j$. Напомним, в задаче предполагается, что одновременно может выполняться произвольное число

работ.

Пусть C^*_{max} и $(\sum w_jC_j)^*$ обозначают значения целевых функций в оптимальных расписаниях для критериев C_{max} и $\sum w_jC_j$ соответственно. В [162] Штайн и Вайн ввели понятие (α, β)-расписания. Допустимое расписание σ относительно критериев f_0 и f_1 называется (α, β)-*расписанием*, если $f_0(\sigma) \leq \alpha f_0(\sigma_0)$ и $f_1(\sigma) \leq \beta f_1(\sigma_1)$, где σ_0 – оптимальное расписание для критерия f_0, и σ_1 – оптимальное расписание для критерия f_1. Алгоритм, строящий такое расписание для любого примера двухкритериальной задачи, называется (α, β)-*приближенным алгоритмом*. В своей работе Штайн и Вайн предложили элегантный метод доказательства существования $(2, 2)$-расписаний для широкого класса задач, в которых требуется одновременно минимизировать критерии C_{max} и $\sum w_jC_j$. Впоследствии оценки на величины α и β в (α, β)-расписаниях были улучшены в [37] и обобщены на любые пары критериев, в которых первый требует минимизации либо длины расписания, либо максимального времени нахождения работы в обслуживании, либо максимального запаздывания, а второй – минимизации либо суммарного времени нахождения работы в обслуживании, либо суммы моментов окончания работ [148]. Результаты, представленные в упомянутых работах, верны для задач, в которых допустимые расписания удовлетворяют следующим условиям.

1. Если в допустимом расписании удалить все работы, заканчивающиеся после момента t, то расписание останется допустимым для остальных работ.

2. Пусть расписания σ_1 и σ_2 допустимы для множеств работ $\mathcal{J}_1$ и $\mathcal{J}_2$ соответственно. Пусть расписание σ получено добавлением расписания σ_2 в конец расписания σ_1 и удалением из σ_2 всех работ из $\mathcal{J}_1 \cap \mathcal{J}_2$. Тогда расписание σ является допустимым для множества работ $\mathcal{J}_1 \cup \mathcal{J}_2$.

Отметим, что кроме чисто теоретического интереса, который они представляют, результаты работ [162, 37, 148] можно использовать для построения (α, β)-приближенных алгоритмов. Действительно, если известны x-приближенный алгоритм для критерия f_0 и y-приближенный алгоритм для критерия f_1, то, используя метод, введенный в [162, 37, 148], можно получить $(x\alpha, y\beta)$-приближенный алгоритм для обоих критериев одновременно.

Безусловно, таким способом можно получить (α, β)-приближенные алгоритмы для различных двукритериальных задач теории расписаний. Однако этот метод, как и всякий общий метод, не учитывает специфику задач. В этом разделе на примере задачи $P\infty|sct|C_{max}, \sum w_jC_j$ покажем, что объединение описанного подхода с методами построения приближенных расписаний, разработанными для конкретных задач, позволяет улучшить качество решений, получаемых (α, β)-приближенными алгоритмами. Заметим, что задачи с задержками не удовлетворяют второму условию Штайна-Вайна, поэтому мы дополнительно представим оценки на существование (α, β)-расписаний для задач с единичными задержками.

4.1.1 Неограниченное число машин

В начале раздела приведем формулировку однокритериальных задач $P\infty|sct|C_{max}$ и $P\infty|sct|\sum w_jC_j$ как задач целочисленного линейного программирования. Затем, ослабляя условие целочисленности, найдем точные решения соответствующих задач линейного программирования. Комбинируя эти решения, получим псевдорасписание. На последнем этапе преобразуем полученное псевдорасписание в расписание, допустимое относительно отношения предшествования и коммуникационных задержек.

Целочисленное линейное программирование (ЦЛП)

Пусть $t_1, \ldots, t_n$ — переменные, определяющие моменты начала выполнения работ $J_1, \ldots, J_n$ соответственно. Для каждой дуги (J_i, J_j) введем переменную $x_{ij} \in \{0, 1\}$, которая определяет наличие или отсутствие задержки между работами J_i и J_j. Тогда следующий набор ограничений определяет допустимые расписания в задачах $P\infty|sct|C_{max}$ и $P\infty|sct|\sum w_jC_j$.

$$t_i + p_i + x_{ij}\delta_{ij} \leq t_j \text{ для всех } (J_i, J_j) \in E; \tag{4.2}$$

$$\sum_{J_j \in \Gamma^+(i)} x_{ij} \geq |\Gamma^+(i)| - 1 \text{ для всех } J_i \in V \setminus U; \tag{4.3}$$

$$\sum_{J_j \in \Gamma^-(i)} x_{ji} \geq |\Gamma^-(i)| - 1 \text{ для всех } J_i \in V \setminus Z; \tag{4.4}$$

$$x_{ij} \in \{0, 1\} \text{ для всех } (J_i, J_j) \in E; \tag{4.5}$$

$$t_i \geq 0 \text{ для всех } J_i \in V. \tag{4.6}$$

Задачи целочисленного линейного программирования отличаются друг от друга только способами вычисления целевых функций.

- Пусть $ILP_{\max}$ — задача целочисленного линейного программирования для задачи $P\infty|sct|C_{\max}$. В задаче ILP_{max} требуется минимизировать величину $C_{\max}$ при ограничениях (4.2)-(4.6) и дополнительном ограничении $t_i + p_i \leq C_{max}$ для всех $J_i \in V$.

- Пусть ILP_Σ — задача целочисленного линейного программирования для задачи $P\infty|sct|\sum w_jC_j$. В задаче ILP_Σ требуется минимизировать величину $\sum w_jC_j$ при ограничениях (4.2)-(4.6) и дополнительном ограничении $t_i + p_i = C_i$ для всех $J_i \in V$.

Напомним, что задачи $ILP_{\max}$ и ILP_Σ являются NP-трудными. Для получения эффективных нижних оценок ослабим ограничение целочисленности (4.5), заменив его на ограничение $x_{ij} \in [0, 1]$ для всех $(J_i, J_j) \in E$. Обозначим полученные задачи линейного программирования $LP_{\max}$ и LP_Σ соответственно. Поскольку число переменных и число ограничений в обеих задачах ограничено полиномом от числа работ, обе задачи можно решить за полиномиальное время [25, 102]. Решение задачи LP_{max}

сопоставит каждой дуге $(J_i, J_j) \in E$ значение $x_{ij} = e'_{ij}$ такое, что $0 \le e'_{ij} \le 1$, и получит нижнюю оценку $C_{max}^{LP_{max}}$ на величину C_{max}^*. Аналогично, решение задачи LP_Σ сопоставит каждой дуге $(J_i, J_j) \in E$ значение $x_{ij} = e''_{ij}$ такое, что $0 \le e''_{ij} \le 1$, и найдет нижнюю оценку $(\sum w_j C_j)^{LP_\Sigma}$ на величину $(\sum w_j C_j)^*$. Заметим, что если значения переменных x_{ij} фиксированы, то оптимальные значения переменных t_j для любого регулярного критерия определяются однозначно:

$$t_j = \begin{cases} 0, & \text{если } J_j \in Z, \\ \max_{J_i \in \Gamma^-(j)} t_i + p_i + x_{ij}\delta_{ij}, & \text{если } J_j \in V \setminus Z. \end{cases} \tag{4.7}$$

Решения задач LP_{max} и LP_Σ будем называть псевдорасписаниями и обозначать через $\sigma^{LP_{max}}$ и σ^{LP_Σ} соответственно. Время окончания работы J_j в псевдорасписании $\sigma^{LP_{max}}$ обозначим через $C_j^{LP_{max}}$, а время окончания работы J_j в псевдорасписании σ^{LP_Σ} — через $C_j^{LP_\Sigma}$.

Процедура объединения Штайна-Вайна

Центральным ядром алгоритма является объединение двух полученных псевдорасписаний $\sigma^{LP_{max}}$ и σ^{LP_Σ} в новое псевдорасписание σ^{LP}. Напомним, что впервые процедуру объединения расписаний предложили Штайн и Вайн в [162].

Процедура $Combine(\sigma^{LP_{max}}, \sigma^{LP_\Sigma}, t)$

1: Определить множество работ V', завершившихся после момента t в расписании σ^{LP_Σ}, и множество дуг E_f таких, что $(J_i, J_j) \in E_f$ тогда и только тогда, когда $J_i \in V \setminus V'$ и $J_j \in V'$.
2: Положить

$$x_{ij} = \begin{cases} e''_{ij}, & \text{если } J_j \in V \setminus V', \\ \max\{e'_{ij}, e''_{ij}\}, & \text{если } (J_i, J_j) \in E_f, \\ e'_{ij}, & \text{если } J_j \in V. \end{cases}$$

3: По полученным значениям переменных x_{ij} вычислить значения переменных t_j по формуле (4.7).

Заметим, что решение, построенное процедурой *Combine*, удовлетворяет ограничениям (4.3) и (4.4). Обозначим полученное псевдорасписание через σ. Положим $C_j(\sigma) = t_j + p_j$. Параметр t, используемый в процедуре $Combine(\sigma^{LP_{max}}, \sigma^{LP_\Sigma}, t)$, назовем точкой разрыва.

Процедура округления

Пусть e_{ij} — значения переменных x_{ij}, $(J_i, J_j) \in E$. Следующий результат непосредственно вытекает из ограничений (4.3) и (4.4).

Утверждение 8 [142] *Каждая работа J_i имеет не более одного родителя с $e_{ji} < \frac{1}{2}$ и не более одного ребенка с $e_{ij} < \frac{1}{2}$.*

Основываясь на этом наблюдении, рассмотрим следующую процедуру округления.

Процедура *Round*(σ)

1: Положить
$$x_{ij} = \begin{cases} 0, & \text{если } e_{ij} < \frac{1}{2}, \\ 1, & \text{если } e_{ij} \geq \frac{1}{2}. \end{cases}$$

2: Вычислить моменты начала работ
$$s_j = \begin{cases} 0, & \text{если } J_j \in Z, \\ \max_{J_i \in \Gamma^-(j)} s_i + p_j + x_{ij}\delta_{ij}, & \text{если } J_j \in V \setminus Z. \end{cases}$$

3: Положить $\sigma^h = \{(J_i, s_i), J_i \in V\}$.

Заметим, что процедура *Round*(σ) по любому псевдорасписанию σ строит допустимое расписание σ^h для задачи $P\infty|sct|C_{max}, \sum w_j C_j$. На первом шаге для каждой работы $J_i \in V \setminus U$ определяется не более одной работы из $\Gamma^+(j)$, которая выполняется на той же машине, что и работа J_i, и которая может стартовать сразу после ее завершения. Пусть $G(\sigma)$ — ориентированный граф, полученный из графа G добавлением к нему фиктивной вершины J_0 и дуг из вершины J_0 в вершины множества U. Зададим веса вершин и ребер в графе $G(\sigma^h)$ следующим образом. Определим вес вершины J_0 равным 0 и вес вершины J_i равным p_i, $i = 1, \ldots, n$. Положим вес каждой дуги, выходящей из J_0, равным 0 и вес каждой дуги $(J_i, J_j) \in E$ равным $e_{ij}\delta_{ij}$, где e_{ij} — значение переменной x_{ij}, полученное на шаге 1 процедуры *Round*. Длина пути из одной вершины в другую равна сумме весов всех вершин и ребер из этого пути. При этом величина $C_j(\sigma^h) = s_j + p_j$ равна длине максимального пути из вершины J_0 в вершину J_j. Путь P^{cr} максимальной длины в графе $G(\sigma^h)$ назовем *критическим.*

Как было показано в [139, 142] для единичных длительностей работ и единичных задержек, время завершения каждой работы в расписании σ^h можно оценить через время завершения этой работы в расписании σ. Такой же результат остается справедливым и для задач с малыми задержками.

Лемма 24 *Пусть σ^h – расписание, полученное процедурой Round(σ), тогда $C_j(\sigma^h) \leq \frac{4}{3}C_j(\sigma)$.*

Доказательство проведем по индукции. Неравенство справедливо для всех работ $J_i \in Z$. Рассмотрим работу $J_j \in V \setminus Z$ и предположим, что данное неравенство справедливо для всех ее родителей. Пусть $J_i \in \Gamma^-(j)$ такая, что $C_j(\sigma^h) = C_i(\sigma^h) + p_i + x_{ij}\sigma_{ij}$. Пусть e_{ij} — значение переменной x_{ij} в σ.

1. Предположим, что $e_{ij} < \frac{1}{2}$. Тогда $C_j(\sigma^h) = C_i(\sigma^h) + p_i$. По индукционной гипотезе получим

$$C_j(\sigma^h) \leq \frac{4}{3}C_i(\sigma) + p_i \leq \frac{4}{3}(C_i(\sigma) + p_i + e_{ij}\delta_{ij}) \leq \frac{4}{3}C_j(\sigma).$$

2. Предположим, что $e_{ij} \geq \frac{1}{2}$. Тогда $C_j(\sigma^h) = C_i(\sigma^h) + p_i + \delta_{ij}$. По индукционной гипотезе получим

$$C_j(\sigma^h) \leq \frac{4}{3}C_i(\sigma) + p_i + \delta_{ij} = \frac{4}{3}(C_i(\sigma) + p_i + \frac{\delta_{ij}}{2}) + \frac{1}{3}\delta_{ij} - \frac{1}{3}p_i.$$

Поскольку для малых задержек выполняется $\delta_{ij} \leq p_i$, с учетом предположения $e_{ij} \geq \frac{1}{2}$ получим

$$C_j(\sigma^h) \leq \frac{4}{3}(C_i(\sigma) + p_i + e_{ij}\delta_{ij}) \leq \frac{4}{3}C_j(\sigma).$$

□

Алгоритм 4.1 для задачи $P\infty|sct|C_{max}, \sum w_jC_j$ состоит из трех основных шагов.

Алгоритм 4.1

1: Выбрать параметр $t \geq 0$.
2: Решить задачи LP_{max} и LP_{Σ}.
3: Найти $\sigma = Combine(\sigma^{LP_{max}}, \sigma^{LP_{\Sigma}}, t)$.
4: $Round(\sigma)$.

Как обсуждалось ранее, допустимость полученного решения следует из утверждения 8 и шага 2 процедуры $Combine(\sigma^{LP_{max}}, \sigma^{LP_{\Sigma}}, t)$, на котором гарантируется выполнение ограничений (4.3) и (4.4) в построенном расписании.

Анализ точности алгоритма для $t = C_{\max}^{LP_{\max}}$.

Легко увидеть, что в зависимости от выбора параметра t алгоритм 4.1 будет получать решения, значения которых ближе к оптимальному либо для одного либо для другого критерия. Выберем $t = C_{\max}^{LP_{\max}}$ и покажем, что при этом t получаемое решение не более чем в $\frac{16}{9}$ раз хуже оптимальных решений для обоих критериев.

Напомним, что V' – это множество работ, завершившихся после момента t в расписании $\sigma^{LP_{\Sigma}}$. Без ограничения общности будем считать, что $\min\limits_{J_i \in V} p_i = 1$. Тогда по условию задачи $\max\limits_{(J_i,J_j)\in E} e_{ij} \leq 1$. Пусть $K = \sum\limits_{J_i \in P_{cr}(\sigma_h)} p_i$, $M = \sum\limits_{J_i \in P_{cr}(\sigma_h) \bigcap V'} p_i$ и $L = \sum\limits_{J_i \in P_{cr}(\sigma_h) \setminus V'} p_i$. Далее, пусть $\xi = \sum\limits_{(J_i,J_j)\in P_{cr}(\sigma_h)} e'_{ij}$. Тогда имеем $C_{\max}^{LP_{\max}} \geq K + \xi$.

Пусть $s_{min}(W, \sigma) = \min_{J_i \in W} s_i(\sigma)$ и $C_{\max}(W, \sigma) = \max_{J_i \in W} C_i(\sigma)$ для некоторого $W \subseteq V$. Под длиной частичного расписания на множестве работ W в σ будем понимать длину интервала $[s_{min}(W, \sigma), C_{\max}(W, \sigma)]$. Обозначим через C'_{max} длину частичного расписания, включающего работы из $P_{cr}(\sigma_h) \bigcap V'$ и через $\bar{C}_{max}$ длину частичного расписания, включающего работы из $P_{cr}(\sigma_h) \setminus V'$. С учетом $\max\limits_{(J_i,J_j)\in E} e_{ij} \leq 1$ длину всего расписания можно оценить, как

$$C_{max} \leq C'_{max} + \bar{C}_{max} + 1. \tag{4.8}$$

Так как на шаге 1 процедуры *Round* длина каждой задержки увеличивается не более чем в два раза, то $C'_{max} \leq M + 2\xi$. Учитывая (4.1), получим $C'_{max} \leq 2M - 1$ и $\bar{C}_{max} \leq 2L - 1$.

Подставляя неравенства $\bar{C}_{max} \leq 2L - 1$ и $C'_{max} \leq M + 2\xi$ в (4.8), получим

$$C_{\max} \leq 2L + M + 2\xi = K + (K - M) + 2\xi \leq 2C_{\max}^{LP_{\max}} - M. \tag{4.9}$$

Так как все работы в $P_{cr}(\sigma_h)\backslash V'$ заканчиваются до момента $C_{\max}^{LP_{\max}}$, то из леммы 24 имеем $\bar{C}_{max} \leq \frac{4}{3}C_{\max}^{LP_{\max}}$. Складывая последнее неравенство с неравенством $C'_{max} \leq 2M - 1$, получим

$$C_{\max} \leq \frac{4}{3}C_{\max}^{LP_{\max}} + 2M. \tag{4.10}$$

Сложив дважды неравенство (4.9) с неравенством (4.10), получим оценку на длину расписания $C_{max} \leq \frac{16}{9}C_{\max}^{LP_{\max}}$.

Оценим величину $\sum w_j C_j$ через моменты завершения каждой работы C_j в расписании σ^h. Если $J_j \in V \setminus V'$, то $C_j(\sigma^h) \leq \frac{4}{3}C_j^{LP_\Sigma}$. В случае $J_j \in V'$ имеем $C_j(\sigma^h) \leq C_{max} \leq \frac{16}{9}C_{\max}^{LP_{\max}} \leq \frac{16}{9}C_j^{LP_\Sigma}$.

Окончательно получим следующий результат.

Теорема 19 *Алгоритм 4.1 с параметром* $t = C_{\max}^{LP_{\max}}$ *является* $(\frac{16}{9}, \frac{16}{9})$*-приближенным алгоритмом для задачи* $P\infty|sct|C_{max}, \sum w_j C_j$.

Выбор наилучшей точки разрыва

В предыдущем разделе псевдорасписание σ^{LP_Σ} обрезалось в точке $t = C_{\max}^{LP_{\max}}$ и затем дополнялось псевдорасписанием работ $\sigma^{LP_{\max}}$. Покажем, что выбор собственной точки разрыва для каждой индивидуальной задачи позволяет улучшить точность получаемых решений.

Рассмотрим псевдорасписания $\sigma^{LP_{max}}$ и σ^{LP_Σ}. Определим $T = C_{\max}^{LP_{\max}}$ и пусть $\sigma = Combine(\sigma^{LP_{max}}, \sigma^{LP_\Sigma}, \alpha T)$ для некоторого положительного α. Тогда неравенство (4.10) примет вид

$$C_{max} \leq \frac{4}{3}\alpha C_{\max}^{LP_{\max}} + 2M. \tag{4.11}$$

Складывая дважды неравенство (4.9) с (4.11), получим $3C_{\max} \leq (\frac{4}{3}\alpha + 4)C_{\max}^{LP_{\max}}$ и

$$C_{\max} \leq (\frac{4}{9}\alpha + \frac{4}{3})C_{\max}^{LP_{\max}} = (\frac{4}{9}\alpha + \frac{4}{3})T. \tag{4.12}$$

Так как любое активное расписание является 2-приближенным относительно обоих критериев $C_{\max}$ и $\sum w_j C_j$, то $\frac{4}{3}$-приближенные алгоритмы для однокритериальных задач $P|sct, m \geq n|C_{max}$ и $P|sct, m \geq n|\sum w_j C_j$ соответственно являются $(\frac{4}{3}, 2)$-приближенным алгоритмом и $(2, \frac{4}{3})$-приближенным алгоритмом для задачи $P\infty|sct|C_{max}, \sum w_j C_j$. Для избежания тривиальных оценок предположим, что $0 < \alpha < 3/2$.

Оценим величину $\sum w_j C_j$, получаемую алгоритмом 4.1, при $t = \alpha T$. Для этого применим метод оценки взвешенной суммы моментов завершения работ через функцию плотности распределения [37].

Нормализуем веса w_j так, чтобы решение задачи LP_Σ удовлетворяло равенству $\sum_j w_j C_j^{LP_\Sigma} = 1$. Рассмотрим функцию

$$g(z) \doteq \sum_j w_j C_j^{LP_\Sigma} \delta(C_j^{LP_\Sigma} - z),$$

где $\delta(\cdot)$ — дельта-функция Дирака, т.е. $\delta(x) = 0$ для всех $x \neq 0$ и $\int_{-\infty}^{\infty} \delta(x)dx = 1$. С учетом нормализованных весов получим $\int_0^{\infty} g(z)dz = \sum_j w_j C_j^{LP_\Sigma} = 1$ и $g(z) \geq 0$. Таким образом $g(z)$ – функция плотности распределения.

Рассмотрим работу J_j, для которой выполнено $C_j^{LP_\Sigma} \leq \alpha T$. После применения процедуры $Combine(\sigma^{LP_{max}}, \sigma^{LP_\Sigma}, \alpha T)$ получим $C_j(\sigma^h) = C_j^{LP_\Sigma}$, и по лемме 24 время завершения этой работы в σ^h не превосходит $\frac{4}{3} C_j^{LP_\Sigma}$. Для работ с $C_j^{LP_\Sigma} > \alpha T$ из неравенства (4.12) имеем $C_j^{LP_\Sigma} \leq (\frac{4}{9}\alpha + \frac{4}{3})T$. Суммируя по всем работам, получим

$$\begin{aligned}
\sum w_j C_j &\leq \int_0^{\alpha T} \frac{4}{3} g(z)dz + \int_{\alpha T}^{\infty} \frac{(\frac{4}{3} + \frac{4}{9}\alpha)T}{z} g(z)dz \\
&= \int_0^{\infty} \frac{4}{3} g(z)dz + \int_{\alpha T}^{\infty} \frac{(\frac{4}{3} + \frac{4}{9}\alpha)T - \frac{4}{3}z}{z} g(z)dz \\
&= \frac{4}{3} + \frac{4}{3} \int_{\alpha T}^{\infty} \frac{(1 + \frac{1}{3}\alpha)T - z}{z} g(z)dz. \qquad (4.13)
\end{aligned}$$

Для построения хорошего решения относительно критерия $\sum w_j C_j$ требуется выбрать значение параметра α, которое минимизирует последнее слагаемое в правой части неравенства. При этом оценка (4.12) на длину расписания также зависит от параметра α. В частности, чтобы гарантированно получить длину расписания не больше $(\frac{4}{3} + \frac{4}{9}\phi)T$, необходимо ограничить интервал выбора значения параметра α от 0 до ϕ. Дополнительно заметим, что каждое решение задачи LP_Σ можно представить как соответствующую функцию плотности распределения $g(z)$. Таким образом, оценивая следующую величину

$$\max_g \min_{0 \leq \alpha \leq \phi} \int_{\alpha T}^{\infty} \frac{(1 + \frac{1}{3}\alpha)T - z}{z} g(z)dz,$$

где $g(z)$ произвольная функция плотности распределения на $[0, \infty)$, получим оценку на отношение $(\sum w_j C_j)/(\sum w_j C_j)^{LP_\Sigma}$. Используя замену переменных $z = Tx$, перепишем последнее выражение как

$$\max_f \min_{0 \leq \alpha \leq \phi} \int_{\alpha}^{\infty} \frac{1 + \frac{1}{3}\alpha - x}{x} f(x)dx, \qquad (4.14)$$

где $f(x) = Tg(Tx)$ также является функцией плотности распределения на $[0, \infty)$.

Предположим, что $f_{\max}$ – это функция плотности распределения, на которой достигается максимальное значение $\theta_{\max}$ интеграла в выражении (4.14). Определим

функцию $A(\alpha) = \int_{\alpha}^{\phi} \frac{1+\frac{1}{3}\alpha - x}{x} f_{\max}(x)dx$. По определению величина $\theta_{\max}$ равна минимальному значению функции $A(\alpha)$ для $\alpha \in [0, \phi]$. Тогда

$$\int_0^{\phi} \theta_{max} d\alpha \leq \int_0^{\phi} A(\alpha)d\alpha.$$

Положим $\phi \leq 3/2$ и рассмотрим выражение $\int_0^{\phi}(1-\frac{2}{3}\alpha)^{-\frac{5}{2}}A(\alpha)d\alpha$.
Так как $(1-\frac{2}{3}\alpha)^{-\frac{5}{2}} \geq 0$ для всех $\alpha \in [0, \phi]$, то

$$\int_0^{\phi} \left(1-\frac{2}{3}\alpha\right)^{-\frac{5}{2}} \theta_{\max} d\alpha \leq \int_0^{\phi} \left(1-\frac{2}{3}\alpha\right)^{-\frac{5}{2}} A(\alpha)d\alpha.$$

Преобразуя левую часть неравенства и подставляя выражение для $A(\alpha)$ в его правую часть, получим

$$\theta_{\max}\left(\left(1-\frac{2}{3}\phi\right)^{-\frac{3}{2}} - 1\right) \leq \int_0^{\phi} \left(1-\frac{2}{3}\alpha\right)^{-\frac{5}{2}} \left[\int_{\alpha}^{\phi} \frac{1+\frac{1}{3}\alpha - x}{x} f_{\max}(x)dx\right] d\alpha.$$

Перепишем последнее неравенство, поменяв порядок интегрирования в правой части,

$$\theta_{\max}\left(\left(1-\frac{2}{3}\phi\right)^{-\frac{3}{2}} - 1\right) \leq \int_0^{\phi} \left[\int_0^{x} \left(1-\frac{2}{3}\alpha\right)^{-\frac{5}{2}} \frac{1+\frac{1}{3}\alpha - x}{x} d\alpha\right] f_{\max}(x)dx.$$

Рассмотрим внутренний интеграл в последнем неравенстве

$$\int_0^{x} \left(1-\frac{2}{3}\alpha\right)^{-\frac{5}{2}} \frac{1+\frac{1}{3}\alpha - x}{x} d\alpha$$

$$= \frac{1-x}{x}\int_0^{x} \left(1-\frac{2}{3}\alpha\right)^{-\frac{5}{2}} d\alpha + \frac{1}{3x}\int_0^{x} \left(1-\frac{2}{3}\alpha\right)^{-\frac{5}{2}} \alpha d\alpha$$

$$= \frac{1-x}{x}\left(\left(1-\frac{2}{3}x\right)^{-\frac{3}{2}} - 1\right) + \frac{1}{3}\left(1-\frac{2}{3}x\right)^{-\frac{3}{2}} - \frac{1}{x}\left(\left(1-\frac{2}{3}x\right)^{-\frac{1}{2}} - 1\right) = 1.$$

Отсюда получим

$$\theta_{\max}\left(\left(1-\frac{2}{3}\phi\right)^{-\frac{3}{2}} - 1\right) \leq \int_0^{\phi} f_{\max}(x)dx = 1.$$

Следовательно, $\theta_{\max} \leq ((1-\frac{2}{3}\phi)^{-\frac{3}{2}} - 1)^{-1}$ и, выбирая наилучшую точку разрыва, получим

$$\sum w_j C_j \leq \frac{4}{3} + \frac{4}{3}\max_f \min_{0\leq\alpha\leq\phi} \int_{\alpha}^{\infty} \frac{1+\frac{1}{3}\alpha - x}{x} f(x)dx$$

$$\leq \frac{4}{3}\left(\frac{(1-\frac{2}{3}\phi)^{-\frac{3}{2}}}{(1-\frac{2}{3}\phi)^{-\frac{3}{2}} - 1}\right) = \frac{4}{3-3(1-\frac{2}{3}\phi)^{\frac{3}{2}}}.$$

ϕ	(α, β)	ϕ	(α, β)	ϕ	(α, β)
0.8	(1.689, 1.958)	0.927	(1.746, 1.746)	1.2	(1.867, 1.465)
0.9	(1.734, 1.785)	1	(1.834, 1.524)	1.4	(1.956, 1.357)

Таблица 4.1: Значения величин α и β в зависимости от значения параметра ϕ.

Таким образом, для любого примера задачи $P\infty|sct|C_{\max}, \sum w_j C_j$ при выборе оптимального значения параметра α в промежутке $[0, \phi]$ алгоритм 4.1 гарантировано найдет $(\frac{4}{3} + \frac{4}{9}\phi, \frac{4}{3-3(1-\frac{2}{3}\phi)^{\frac{3}{2}}})$-расписание.

Покажем, что оптимальное значение параметра α можно найти за полиномиальное время. Действительно, процедура *Combine* разбивает множество работ на два непересекающихся подмножества согласно их моментам завершения в псевдорасписании σ^{LP_Σ}. Очевидно, что существует не более n таких разбиений, одно из которых соответствует оптимальному выбору α. Таким образом, достаточно рассмотреть только моменты времени, соответствующие различным моментам завершения работ в псевдорасписании σ^{LP_Σ}.

Алгоритм 4.2

1: Решить задачи LP_{max} и LP_Σ.
2: **for** $j = 1, \ldots, n$ **do**
3: Положить $t = C_j^{LP_\Sigma}$.
4: Если $t \leq \phi T$, то найти $\sigma_j = Combine(\sigma^{LP_{max}}, \sigma^{LP_\Sigma}, t)$.
5: $Round(\sigma_j)$.
6: Выдать расписание с наименьшим значением критерия $\sum w_i C_i$.

Заметив, что значение дроби $\frac{4}{3-3(1-\frac{2}{3}\phi)^{\frac{3}{2}}}$ при изменении ϕ в интервале $(0, \frac{3}{2})$ монотонно убывает и при $\phi = (1 - 3^{-\frac{2}{3}})\frac{3}{2} \approx 0.721$ равно 2, получим следующий результат.

Теорема 20 *Для любого $\phi \in [0.721, 1.5]$ алгоритм 4.2 является (α, β)-приближенным алгоритмом для задачи $P\infty|sct|C_{max}, \sum w_j C_j$, где $\alpha = \frac{4}{3} + \frac{4}{9}\phi$ и $\beta = \frac{4}{3-3(1-\frac{2}{3}\phi)^{\frac{3}{2}}}$.*

Для определенности в таблице 4.1.1 приведены конкретные значения величин α и β для некоторых значений ϕ.

4.1.2 Одновременная минимизация длины расписания и взвешенной суммы моментов окончания работ с доставками

Полученные результаты легко обобщаются на задачу, в которой для каждой работы J_j дополнительно задано время доставки q_j. Положим $\bar{L}_j = C_j + q_j$ и рассмотрим целевые функции $\bar{L}_{\max} = \max_j \bar{L}_j$ и $\sum_j w_j \bar{L}_j$. Заметим, что задачи с доставками и критериями $\bar{L}_{\max}$ и $\sum_j w_j \bar{L}_j$ эквивалентны задачам минимизации максимального запаздывания $L_{\max}$ и минимизации суммарного запаздывания $\sum_j w_j L_j$ соответственно,

в том смысле, что их оптимальные решения совпадают. С другой стороны, если все q_j положить равными нулю, то получим рассмотренные задачи минимизации длины расписания и взвешенной суммы моментов окончания работ.

Чтобы применить описанную в предыдущих разделах процедуру построения приближенных решений, переформулируем задачи целочисленного линейного программирования, заменив способы вычисления целевых функций.

- Пусть ILP_1 — задача целочисленного линейного программирования для задачи $P\infty|sct|\bar{L}_{max}$. В ILP_1 требуется минимизировать величину $\bar{L}_{max}$ при ограничениях (4.2)-(4.6) и дополнительном ограничении $t_i + p_i + q_i \leq \bar{L}_{max}$ для всех $J_i \in V$.

- Пусть ILP_2 — задача целочисленного линейного программирования для задачи $P\infty|sct|\sum w_j\bar{L}_j$. В ILP_2 требуется минимизировать величину $\sum w_j\bar{L}_j$ при ограничениях (4.2)-(4.6) и дополнительном ограничении $t_i + p_i + q_i = \bar{L}_i$ для всех $J_i \in V$.

Пусть σ^{LP_1} и σ^{LP_2} — решения задач ILP_1 и ILP_2, в которых ослаблено условие целочисленности на переменные x_{ij}. Применим алгоритм 4.2 на этих псевдорасписаниях. Пусть J_i — работа, на которой в расписании σ, построенном алгоритмом 4.2, достигается значение $\bar{L}_{max}$. Тогда, рассматривая критический путь из вершины J_0 в вершину J_i в графе $G(\sigma)$ и повторяя рассуждения из предыдущего раздела, получим $C_i \leq (\frac{4}{9}\phi + \frac{4}{3})C_i^{LP_{\max}}$. Отсюда $\bar{L}_{max} = \bar{L}_i = C_i + q_i \leq (\frac{4}{9}\phi + \frac{4}{3})C_i^{LP_1} + q_i \leq (\frac{4}{9}\phi + \frac{4}{3})(C_i^{LP_1} + q_i) \leq (\frac{4}{9}\phi + \frac{4}{3})\bar{L}_{max}^{LP_1}$. Пусть $\beta = \frac{4}{3-3(1-\frac{2}{3}\phi)^{\frac{3}{2}}}$. Аналогично оценим величину $\sum w_j\bar{L}_j$ как $\sum w_j\bar{L}_j = \sum w_jC_j + \sum w_jq_j \leq \beta\sum w_jC_j^{LP_2} + \sum w_jq_j \leq \beta(\sum w_j\bar{L}_j)^{LP_2}$.

Теорема 21 *Для любого $\phi \in [0.721, 1.5]$ алгоритм 4.2 является (α, β)-приближенным алгоритмом для задачи $P\infty|sct|\bar{L}_{max}, \sum w_j\bar{L}_j$, где $\alpha = \frac{4}{3} + \frac{4}{9}\phi$ и $\beta = \frac{4}{3-3(1-\frac{2}{3}\phi)^{\frac{3}{2}}}$.*

4.1.3 Верхние оценки на существование (α, β)-приближенных расписаний

Для задачи с единичными длительностями и задержками покажем, что существует (α, β)-расписание лучше того, которое находит алгоритм 4.2. Так как обе задачи $P\infty|uct, p_j = 1|C_{max}$ и $P\infty|uct, p_j = 1|\sum w_jC_j$ являются NP-трудными в сильном смысле, то для их решения, скорее всего, не существует точных полиномиальных алгоритмов. Однако можно использовать знание о существовании точных решений этих задач для того, чтобы показать существование решения двухкритериальной задачи, одновременно близкого к оптимальным решениям обеих однокритериальных задач. Предположим, что $\sigma^*_{\max}$ и σ^*_{Σ} – оптимальные расписания в задачах $P\infty|uct, p_j = 1|C_{max}$ и $P\infty|uct, p_j = 1|\sum w_jC_j$ соответственно.

Теорема 22 *Для любого $\phi \in [0, 1]$ существует $(1+\frac{\phi}{2}, \frac{4}{\phi(4-\phi)})$-приближенное решение для задачи $P\infty|uct, p_j = 1|C_{max}, \sum w_jC_j$.*

Доказательство. Выберем значение параметра α такое, что $\alpha C^*_{\max}$ – целое. Пусть $t_1, \ldots, t_n$ – времена начала выполнения работ $J_1, \ldots, J_n$ в расписании $\sigma^*_{\max}$ или σ^*_Σ, а $x_{ij} = 0$, если работы J_i и J_j выполняются на одной машине, и $x_{ij} = 1$ в противном случае. Тогда расписания $\sigma^*_{\max}$ и $\sigma^*_\sum$ удовлетворяют ограничениям (4.2)-(4.4). Пусть e'_{ij} и e''_{ij} — значения переменной x_{ij} в расписаниях $\sigma^*_{\max}$ и σ^*_Σ соответственно.

Положим

$$x_{ij} = \begin{cases} e''_{ij}, & \text{если } t_j(\sigma^*_\Sigma) \le \alpha C^*_{\max}, \\ e'_{ij}, & \text{если } t_j(\sigma^*_\Sigma) > \alpha C^*_{\max}. \end{cases}$$

Пусть σ – полученное расписание. Легко показать, что $C_{max}(\sigma) \le (1 + \frac{\alpha}{2})C^{OPT}_{max}$. Действительно, пусть P^{cr} – критический путь в расписании σ. Тогда $C_{max} \le \alpha C^{OPT}_{max} + C^{OPT}_{max} - \sum_{J_j \in P^{cr}: C_j(\sigma) \le \alpha C^{OPT}_{max}} p_j \le C^{OPT}_{max} + \frac{\alpha}{2} C^{OPT}_{max}$. Последнее неравенство следует из того, что доля времени, потраченная на выполнение любой цепочки работ, всегда не меньше доли времени, потраченной на задержки между работами. Следовательно, повторяя рассуждения из предыдущего раздела, получим

$$\sum w_j C_j \le 1 + \int_{\alpha T}^{\infty} \frac{(1 + \frac{1}{2}\alpha)T}{z} g(z)dz. \tag{4.15}$$

Положим $\theta_{\max} = \max_f \min_{0 \le \alpha \le \phi} \int_\alpha^\infty \frac{1 + \frac{1}{2}\alpha - x}{x} f(x)dx$. Пусть $\phi < 2$, тогда для $\theta_{\max}$ выполнено неравенство

$$8 \int_0^\infty (2-\alpha)^{-3} \theta_{\max} d\alpha \le 8 \int_0^\infty (2-\alpha)^{-3} \left[\int_\alpha^\infty \frac{1 + \frac{1}{2}\alpha - x}{x} f(x)dx \right] d\alpha.$$

Выражение в правой части равно 1 и, следовательно, $\theta_{\max} \le \frac{1}{4(2-\phi)^{-2}-1}$. Подставляя полученное выражение в 4.15, получим результат теоремы. □

Следствие 4 *Существует* $(1.445, 1.445)$*-приближенное решение для задачи* $P\infty|uct, p_j = 1|C_{max}, \sum w_j C_j$.

4.1.4 Ограниченное число машин

Для задач с задержками и одним критерием приближенное решение с неограниченным числом машин можно использовать для получения приближенного решения с заданным (фиксированным) числом машин. Покажем, что это можно сделать и в двухкритериальной задаче.

Расширим линейные программы LP_{max} и LP_Σ, добавив в них неравенство Шульца [152]

$$\sum_{j \in A} (t_j + 1) \ge \frac{1}{2m}|A|^2 + \frac{1}{2}|A| \text{ для всех } A \subseteq V. \tag{4.16}$$

Обозначим полученные задачи линейного программирования через LP^+_{max} и LP^+_Σ. Заметим, что решения этих задач не являются нижними оценками на оптимальное решение соответствующих задач с неограниченным числом машин. Однако, следующее утверждение выполнено для задач, в которых рассматривается m машин.

Утверждение 9 [153] *Любое допустимое решение задачи* $P|uct, p_j = 1|C_{\max}$ *или задачи* $P|uct, p_j = 1|\sum w_j C_j$ *удовлетворяет ограничениям (4.9), (4.10) и (4.16).*

Шульц также доказал, что задачи LP^+_{max} и LP^+_Σ разрешимы за полиномиальное время [152].

Рассмотрим последовательное применение процедур $Combine(\sigma^{LP^+_{\max}}, \sigma^{LP^+_\Sigma}, t)$ и *Round* к псевдорасписаниям $\sigma^{LP^+_{\max}}$ и $\sigma^{LP^+_\Sigma}$. Обозначим через σ^∞ полученное расписание. Положив $t = \alpha C^{LP^+_{\max}}_{\max}$, получим $C_{\max}(\sigma^\infty) \le (\frac{4}{3} + \frac{4}{9}\alpha) C^{LP^+_{\max}}_{\max}$. Заметим, что расписание σ^∞ допустимо относительно порядка предшествования и задержек, но в нем может выполняться больше m работ одновременно. В [139] предложен простой алгоритм *GLS*, преобразующий расписание σ^∞ в допустимое расписание σ^h. Будем говорить, что не назначенная в частичное расписании σ работа J_i *доступна* в момент времени t, если обслуживание всех ее родителей завершено и

$$t \ge C_i(\sigma) + x_{ij} \text{ для всех } (J_i, J_j) \in E,$$

где $x_{ij} = 1$, если в расписании σ^∞ есть единичная задержка между работами J_i и J_j, и $x_{ij} = 0$ в противном случае.

Пусть задан произвольный список $\mathcal{L}$ работ из V. В каждый момент времени t, в котором выполняется меньше m работ, алгоритм $GLS(\sigma^\infty, \mathcal{L})$ ставит в расписание в интервал $[t, t+1)$ первую доступную работу из списка $\mathcal{L}$ и продолжает так до тех пор, пока либо не останется доступных работ, либо число работ, выполняемых в интервале $[t, t+1)$, станет равно m.

Утверждение 10 [139] *Пусть* σ^h *— расписание, полученное алгоритмом GLS из расписания* σ^∞ *с использованием списка* $\mathcal{L} = (J_1, \ldots, J_n)$. *Тогда*

$$C_j(\sigma^h) \le C_j(\sigma^\infty) + \frac{j}{m}, \tag{4.17}$$

где m *– число машин.*

Заметим, что утверждение 10 верно для любого списка $\mathcal{L}$. Так как последнее слагаемое в выражении (4.17) является нижней оценкой на длину расписания в задаче $P|uct, p_j = 1|C_{\max}$, получаем $C^h_{\max} \le (\frac{7}{3} + \frac{4}{9}\alpha) C^{opt,m}_{max}$, где $C^{opt,m}_{max}$ — длина самого короткого из всех допустимых расписаний на m машинах. В дополнение для любого допустимого решения задачи LP^+_Σ в [153] установлен следующий результат.

Утверждение 11 [153] *Пусть значения переменных* $t_1, \ldots, t_n$ *удовлетворяют неравенству (4.16) и* $C_i = t_i + 1$ *для всех* $i = 1, \ldots, n$. *Предположим, что* $A \subseteq V$, $J_j \in A$ *и* $C_k \le C_j$ *для всех* $J_k \in A$. *Тогда* $\frac{|A|}{m} \le 2C_j$.

Рассмотрим следующий алгоритм построения (α, β)-расписания для задачи $P|uct, p_j = 1|C_{\max}, \sum w_j C_j$.

Алгоритм 4.3

1: Решить задачи LP^+_{max} и LP^+_Σ.
2: Построить список работ $\mathcal{L}$, в котором работы занумерованы в порядке неубывания величин $C_j(\sigma^{LP^+_\Sigma})$.
3: Найти расписание $\sigma = Combine(\sigma^{LP^+_{\max}}, \sigma^{LP^+_\Sigma}, t)$.
4: Найти расписание $\sigma^\infty = Round(\sigma)$.
5: Найти расписание $\sigma^h = GLS(\sigma^\infty, \mathcal{L})$.
6: Выдать расписание σ^h.

Оценим параметры α и β в расписании σ^h, полученном алгоритмом 4.3, с параметром $t = \alpha C^{LP^+_{max}}_{\max}$. Пусть $T = C^{LP^+_{max}}_{\max}$.

Рассмотрим работу J_j, для которой выполнено $C_j^{LP^+_\Sigma} \le \alpha T$. После применения процедуры $Combine(\sigma^{LP_{max}}, \sigma^{LP_\Sigma}, \alpha T)$ получим $C_j(\sigma) = C_j^{LP^+_\Sigma}$, и из леммы 24 вытекает, что $C_j(\sigma^\infty) \le \frac{4}{3} C_j^{LP^+_\Sigma}$. Тогда из утверждения 11 следует, что

$$C_j(\sigma^h) \le \frac{4}{3} C_j^{LP^+_\Sigma} + 2C_j^{LP^+_\Sigma} = \frac{10}{3} C_j^{LP^+_\Sigma}.$$

Для работ с $C_j^{LP_{\Sigma^+}} > \alpha T$ из неравенства (4.12) имеем $C_j^\infty \le (\frac{4}{9}\alpha + \frac{4}{3})T$.
Пусть $C_j^{LP_{\Sigma^+}} = z$, тогда из утверждений 10 и 11 получим

$$C_j(\sigma^h) \le C_j^\infty + \frac{j}{m} \le (\frac{4}{9}\alpha + \frac{4}{3})T + 2z.$$

Нормализуем веса w_j так, чтобы решение задачи LP^+_Σ удовлетворяло равенству $\sum_j w_j C_j^{LP^+_\Sigma} = 1$. Полагая $g(z) \doteq \sum_j w_j C_j^{LP_\Sigma} \delta(C_j^{LP_\Sigma} - z)$, получим

$$\sum w_j C_j \le \int_0^{\alpha T} \frac{10}{3} g(z)dz + \int_{\alpha T}^\infty \frac{(\frac{4}{3} + \frac{4}{9}\alpha)T + 2z}{z} g(z)dz$$

$$= \int_0^\infty \frac{10}{3} g(z)dz + \int_{\alpha T}^\infty \frac{(\frac{4}{3} + \frac{4}{9}\alpha)T + 2z - \frac{10}{3}z}{z} g(z)dz$$

$$= \frac{10}{3} + \frac{4}{3} \int_{\alpha T}^\infty \frac{(1 + \frac{1}{3}\alpha)T - z}{z} g(z)dz.$$

Заметим, что интеграл во втором слагаемом в последнем выражении совпадает со вторым слагаемым в выражении (4.13) и при изменении параметра α в интервале $(0, \phi)$, $\phi \le \frac{3}{2}$, ограничен величиной $((1 - \frac{2}{3}\phi)^{-\frac{3}{2}} - 1)^{-1}$.

Отсюда получим

$$\sum w_j C_j \le \frac{10}{3} + \frac{4}{3}((1 - \frac{2}{3}\phi)^{-\frac{3}{2}} - 1)^{-1}.$$

Теорема 23 *Для любого $\phi \in [0, 1.5]$ алгоритм 4.3 является (α, β)-приближенным алгоритмом для задачи $P|uct, p_j = 1|C_{\max}, \sum w_j C_j$ со значениями $\alpha = \frac{7}{3} + \frac{4}{9}\phi$ и $\beta = \frac{10}{3} + \frac{4}{3}((1 - \frac{2}{3}\phi)^{-\frac{3}{2}} - 1)^{-1}$.*

4.2 Задачи с задержками в иерархической коммуникационной системе

Рассмотрим задачи построения расписания множества работ, связанных отношениями предшествования, при наличии задержек в иерархической коммуникационной системе.

В задаче $P\infty(P2)|sct|C_{max}$ множество работ $V = \{J_1, \ldots, J_n\}$ должно быть выполнено на m идентичных параллельных машинах. Каждая машина имеет два процессора и, следовательно, может выполнять две работы одновременно. Число машин в системе неограничено, то есть $m \geq n$. Пусть ациклический ориентированный граф $G = (V, E)$ задает частичный порядок на множестве работ V. Без ограничения общности предположим, что в графе G нет транзитивных замыканий. Для каждой работы $J_i \in V$ задана ее длительность p_i и неотрицательный вес w_i. Каждая дуга $(J_i, J_j) \in E$ ассоциируется с отношением предшествования между работами J_i и J_j, а ее вес δ_{ij} с коммуникационной задержкой. В дальнейшем предположим, что задержки малы, т.е. $\max_{(J_i,J_j)\in E} \delta_{ij} \leq \min_{J_i \in V} p_i$. Таким образом, если работы J_i и J_j выполняются на одной машине, то работа J_j может стартовать сразу после окончания работы J_i, т.е. в любом допустимом расписании должно выполняться $s_j \geq s_i + p_i$. Если работы J_i и J_j выполняются на разных машинах, то работа J_j не может начаться раньше времени $s_i + p_i + \delta_{ij}$. Требуется построить расписание, допустимое относительно отношения предшествования и задержек, которое имеет минимальную длину.

Пусть $\Phi = \frac{\min_{J_i \in V} p_i}{\max_{(J_i,J_j)\in E} \delta_{ij}} \geq 1$. Для неиерархической задачи $P\infty|sct|C_{max}$ Мунье и Ханен [141] предложили $\frac{2(1+\Phi)}{2\Phi+1}$-приближенный алгоритм, который является прямым обобщением $\frac{4}{3}$-приближенного алгоритма Мунье и Кёниг для задачи с единичными длительностями работ и единичными задержками.

В [43] было показано, что задача $P\infty(P2)|sct|C_{max}$ является NP-трудной в сильном смысле, даже если длительности всех работ и все задержки равны 1. Более того, существование ρ-приближенного алгоритма с $\rho < \frac{5}{4}$ для задачи $P\infty(P2)|uct, p_j = 1|C_{max}$, в которой длительности всех работ и все задержки равны 1, влечет совпадение классов P и NP. Для задачи $P\infty(Pm)|uct, p_j = 1|C_{max}$ с единичными длительностями и задержками при наличии в каждой группе произвольного фиксированного числа машин в [42] предложен $(2 - \frac{2}{2m+1})$-приближенный алгоритм. Однако в отличие от результатов для неиерархической модели, результат этой статьи не переносится на задачу с малыми задержками. Хотя предложенный ниже алгоритм также основан на округлении дробных значений в решении соответствующей задачи линейного программирования, формулировка задачи ЛП и последующая процедура округления значительно сложнее, чем для иерархической задачи с единичными длительностями и единичными задержками.

4.2.1 Линейное программирование

В этом разделе сформулируем задачу линейного программирования, решение которой дает нижнюю оценку величины оптимального решения в задаче $P\infty(P2)|sct|C_{max}$, и в дальнейшем эта оценка будет использована для построения приближенного решения. Рассмотрим адаптированную версию линейной программы, предложенной в [43] для задачи $P\infty(P2)|uct, p_j = 1|C_{max}$.

Как и в предыдущем разделе, множество всех детей и родителей работы J_i обозначим через $\Gamma^+(i)$ и $\Gamma^-(i)$ соответственно. Множество работ, у которых нет родителей, обозначим через Z, а множество работ, у которых нет детей, — через U. Без ограничения общности предположим, что $\max_{(J_i,J_j)\in E} \delta_{ij} = 1$ и $\min_{J_i\in V} p_i = \Phi \geq 1$.

Пусть $t_1, \ldots, t_n$ — переменные, определяющие моменты начала выполнения работ $J_1, \ldots, J_n$ соответственно. Для каждой дуги (J_i, J_j) введем переменную $x_{ij} \in \{0, 1\}$, которая определяет наличие или отсутствие задержки между работами J_i и J_j. Рассмотрим следующую задачу линейного программирования (задача LP):

$$\min\ C_{\max},$$

$$t_i + p_i + x_{ij}\delta_{ij} \leq t_j \ \text{ для всех } \ (J_i, J_j) \in E; \tag{4.18}$$

$$\sum_{J_j\in\Gamma^+(i)} x_{ij} \geq |\Gamma^+(i)| - 2 \ \text{ для всех } \ J_i \in V \setminus U; \tag{4.19}$$

$$\sum_{J_j\in\Gamma^-(i)} x_{ji} \geq |\Gamma^-(i)| - 2 \ \text{ для всех } \ J_i \in V \setminus Z; \tag{4.20}$$

$$\begin{aligned} &x_{ji} + x_{jk} + x_{lk} + x_{lm} \geq 1 \\ &\quad \text{для всех } \ J_i, J_j, J_k, J_l, J_m \in V \ \text{ таких,} \\ &\quad \text{что } \ (J_j, J_i), (J_j, J_k), (J_l, J_k), (J_l, J_m) \in E; \end{aligned} \tag{4.21}$$

$$\begin{aligned} &x_{ij} + x_{kj} + x_{lk} + x_{ml} \geq 1 \\ &\quad \text{для всех } \ J_i, J_j, J_k, J_l, J_m \in V \ \text{ таких,} \\ &\quad \text{что } \ (J_i, J_j), (J_k, J_j), (J_k, J_l), (J_m, J_l) \in E; \end{aligned} \tag{4.22}$$

$$x_{ij} \in [0, 1] \ \text{ для всех } \ (J_i, J_j) \in E; \tag{4.23}$$

$$t_i \geq 0 \ \text{ для всех } \ J_i \in V; \tag{4.24}$$

$$t_i + p_i \leq C_{\max}. \tag{4.25}$$

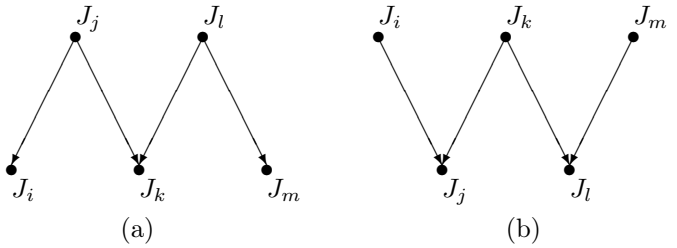

Рис. 4.1: Специальные подграфы, рассматриваемые в LP

Ограничения (4.18)-(4.20) сходны с ограничениями (4.2)-(4.4) из предыдущего раздела и указывают на то, что не более двух детей одной работы могут начать выполнение непосредственно после ее завершения, и каждая работа может стартовать непосредственно после завершения не более чем двух ее родителей.

Неравенства (4.21) и (4.22) задают ограничения для специальных подграфов орграфа G, изоморфных ориентированным графам, изображенным на рисунке 4.1. Допустимость этих неравенств будет показана в следующем параграфе.

Поскольку число переменных и число ограничений в задаче LP ограничено полиномом от числа работ ($O(n^2)$ переменных и $O(n^5)$ ограничений), ее можно решить за полиномиальное время [25, 102]. Решение задачи LP сопоставит каждой дуге $(J_i, J_j) \in E$ значение $x_{ij} = e_{ij}$ $0 \leq e_{ij} \leq 1$, которое необязательно определяет допустимое решение задачи $P\infty(P2)|uct, p_j = 1|C_{max}$. Алгоритм построения допустимого решения будет представлен в параграфе 4.2.3.

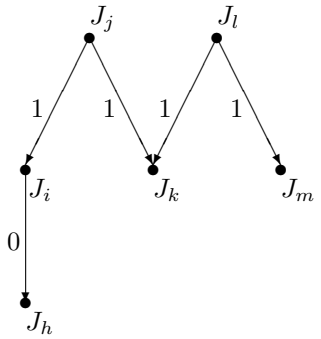

Рис. 4.2: Пример, на котором решение задачи ЦЛП не является нижней оценкой на оптимальное решение задачи $P\infty(P2)|sct|C_{max}$.

В следующем параграфе докажем, что решение задачи LP дает нижнюю оценку $\Theta^{\inf}$ на длину оптимального расписания задачи $P\infty(P2)|sct|C_{max}$. Важно отметить, что в отличие от аналогичной задачи с единичными длительностями и задержками, задача с малыми задержками не может быть сформулирована как целочисленная версия рассматриваемой задачи линейного программирования с $x_{ij} \in \{0, 1\}$. Действительно, рассмотрим пример задачи с шестью работами, изображенный на рис. 4.2. Пусть длительности работ равны $p_i = p_j = p_m = p_h = 1$, $p_l = p_k = 1.5$, $\delta_{ji} = \delta_{jk} = \delta_{lk} = \delta_{lm} = 1$, и $\delta_{ih} = 0$. Если потребовать целочисленность переменных x,

то одна из переменных, соответствующая дугам $(J_j, J_i), (J_j, J_k), (J_l, J_k), (J_l, J_m)$, примет значение 1, и длина полученного расписания будет равна 3.5. В оптимальном расписании одна машина выполняет все работы кроме работы J_h: первый процессор последовательно без простоя выполняет работы J_j, J_i и J_m, второй процессор — работы J_l и J_k. Работа J_h выполняется на другой машине сразу после завершения работы J_i. Длина такого расписания равна 3.

4.2.2 Нижняя оценка

Покажем, что решение задачи LP дает нижнюю оценку на величину оптимального решения задачи $P\infty(P2)|sct|C_{max}$.

Лемма 25 *Величина $\Theta^{\inf}$ является нижней оценкой на величину оптимального решения задачи $P\infty(P2)|sct|C_{max}$.*

Доказательство. Пусть I — произвольный пример задачи $P\infty(P2)|sct|C_{max}$. Покажем, что для любого допустимого расписания работ σ в примере I существует решение задачи LP такое, что момент начала выполнения каждой работы в решении задачи линейного программирования не больше, чем момент начала выполнения каждой работы в рассматриваемом расписании.

Пусть $s_i(\sigma)$ – момент начала выполнения работы J_i в расписании σ. Построим по расписанию σ решение $\mathcal{L} = (t_i(\mathcal{L}), x_{ij}(\mathcal{L}))$, которое удовлетворяет всем ограничениям задачи LP.

Для всех $(J_i, J_j) \in E$ положим

$$x_{ij}(\mathcal{L}) = \begin{cases} \min\{(s_j(\sigma) - s_i(\sigma) - p_i)/\delta_{ij}, 1\}, & \text{если } \delta_{ij} \neq 0; \\ 1, & \text{если } \delta_{ij} = 0. \end{cases}$$

Для всех $J_i \in Z$ положим $t_i(\mathcal{L}) = 0$ и для всех $J_j \in V \setminus Z$ положим $t_j(\mathcal{L}) = \max\limits_{(J_i,J_j)\in E}\{t_i(\mathcal{L}) + p_i + x_{ij}(\mathcal{L})\delta_{ij}\}$.

Покажем, что $t_i(\mathcal{L}) \leq s_i(\sigma)$ для всех работ $J_i \in V$. Очевидно, что неравенство выполнено для всех работ $J_i \in Z$. Пусть неравенство верно для всех предшественников работы $J_j \in V \setminus Z$. Рассмотрим работу $J_i \in \Gamma^-(j)$ такую, что $t_j(\mathcal{L}) = t_i(\mathcal{L}) + p_i + x_{ij}(\mathcal{L})\delta_{ij}$. По определению $x_{ij}(\mathcal{L})$ имеем

$$t_j(\mathcal{L}) \leq t_i(\mathcal{L}) + p_i + \frac{s_j(\sigma) - s_i(\sigma) - p_i}{\delta_{ij}}\delta_{ij} = t_i(\mathcal{L}) + s_j(\sigma) - s_i(\sigma) \leq s_j(\sigma).$$

Последнее неравенство верно в силу индукционного предположения для работы J_i.

Также заметим, что если для некоторой дуги $(J_i, J_j) \in E$ работы J_i и J_j выполняются на разных машинах в расписании σ, то

$$x_{ij}(\mathcal{L}) = \min\left\{\frac{s_j(\sigma) - s_i(\sigma) - p_i}{\delta_{ij}}, 1\right\} \leq \min\left\{\frac{c_{ij}}{c_{ij}}, 1\right\} = 1.$$

Докажем, что решение $\mathcal{L}$ удовлетворяет всем ограничениям задачи LP.

Ограничения (4.18),(4.24) и (4.25) удовлетворяются по построению решения $\mathcal{L}$, поскольку $1 \geq x_{ij}(\mathcal{L}) \geq 0$.

Так как в любом допустимом расписании у каждой работы J_i не более двух детей J_{j_1} и $J_{j_2} \in \Gamma^+(i)$ выполняются на той же машине, что и J_i, в интервале $[t_i+p_i, t_i+p_i+1]$, то для не более двух ее потомков значение $x_{ij}(\mathcal{L})$, $J_j \in \Gamma^+(i)$ может быть меньше 1. Отсюда следует, что неравенство (4.19) также выполнено. Аналогично доказывается, что решение $\mathcal{L}$ удовлетворяет ограничению (4.20).

Пусть в G есть ориентированный подграф, изоморфный изображенному на рисунке 4.1(a).

Если работы J_i, J_j, J_k, J_l, J_m выполняются более чем на одной машине в расписании σ, то по крайней мере одна из переменных в неравенстве (4.20) имеет значение 1, и, следовательно, неравенство (4.20) выполнено.

Пусть все работы J_i, J_j, J_k, J_l, J_m обслуживаются некоторой машиной M. Без ограничения общности предположим, что

$$s_j(\sigma) + p_j \leq s_l(\sigma) + p_l. \tag{4.26}$$

Если работы J_j иJ_l выполняются на одном процессоре, то

$$\begin{aligned} x_{jk}(\mathcal{L}) = &\ \min\left\{\tfrac{s_k(\sigma)-s_j(\sigma)-p_j}{\delta_{jk}}, 1\right\} \\ \geq &\ \min\left\{\tfrac{s_j(\sigma)+p_j+p_l-s_j(\sigma)-p_j}{\delta_{jk}}, 1\right\} = 1, \end{aligned}$$

и неравенство (4.20) выполнено.

Пусть работы J_j и J_l выполняются на одной машине, но на разных процессорах. Обозначим через π_j процессор, на котором выполняется работа J_j, и через π_l процессор, на котором выполняется работа J_l. Заметим, что машина M не может обслуживать три работы J_i, J_k и J_m одновременно. Следовательно, одна из этих работ должна стартовать после окончания другой из этих работ.

Рассмотрим возможное взаимное расположение работ J_i, J_k и J_m в расписании σ. Вначале рассмотрим случаи, в которых при построении решения $\mathcal{L}$ значение одной из переменных в ограничении (4.20) становится равным 1, что сразу влечет его выполнение. Последним рассмотрим случай, в котором при построении решения $\mathcal{L}$ все значения переменных могут быть меньше 1.

(a) Работа J_i выполняется после завершения работы J_k (или работы J_m). Тогда получим

$$\begin{aligned} x_{ji}(\mathcal{L}) = &\ \min\left\{\tfrac{s_i(\sigma)-s_j(\sigma)-p_j}{\delta_{ji}}, 1\right\} \\ \geq &\ \min\left\{\tfrac{s_j(\sigma)+p_j+p_k-s_j(\sigma)-p_j}{\delta_{ji}}, 1\right\} = 1. \end{aligned}$$

Если работа J_i выполняется после завершения работы J_m, с учетом неравенства (4.26) получаем такую же цепочку неравенств.

(b) Работа J_k выполняется после завершения работы J_i (или работы J_m). Тогда получим

$$\begin{aligned} x_{jk}(\mathcal{L}) = &\ \min\left\{\tfrac{s_k(\sigma)-s_j(\sigma)-p_j}{\delta_{jk}}, 1\right\} \\ \geq &\ \min\left\{\tfrac{s_j(\sigma)+p_j+p_i-s_j(\sigma)-p_j}{\delta_{jk}}, 1\right\} = 1. \end{aligned}$$

(c) Работа J_m выполняется после завершения работы J_k. Тогда получим

$$\begin{aligned} x_{lm}(\mathcal{L}) = &\ \min\left\{\tfrac{s_m(\sigma)-s_l(\sigma)-p_l}{\delta_{jk}}, 1\right\} \\ \geq &\ \min\left\{\tfrac{s_l(\sigma)+p_l+p_k-s_l(\sigma)-p_l}{\delta_{jk}}, 1\right\} = 1. \end{aligned}$$

(d) Работа J_m выполняется после завершения работы J_i, и работа J_i выполняется на процессоре π_l. В этом случае получим

$$\begin{aligned} x_{lm}(\mathcal{L}) = &\ \min\left\{\tfrac{s_m(\sigma)-s_l(\sigma)-p_l}{\delta_{lm}}, 1\right\} \\ \geq &\ \min\left\{\tfrac{s_l(\sigma)+p_l+p_i-s_l(\sigma)-p_l}{\delta_{lm}}, 1\right\} \\ \geq &\ \min\left\{\tfrac{p_i}{c_{lm}}, 1\right\} = 1. \end{aligned}$$

(e) Предположим, что процессор π_j последовательно выполняет работы J_j, J_i, J_m, а процессор π_l выполняет работы J_l и J_k. Тогда

$$\begin{aligned} x_{jk}(\mathcal{L}) + x_{lm}(\mathcal{L}) &= \min\left\{\tfrac{s_k(\sigma)-s_j(\sigma)-p_j}{\delta_{jk}}, 1\right\} + \min\left\{\tfrac{s_m(\sigma)-s_l(\sigma)-p_l}{\delta_{lm}}, 1\right\} \\ &\geq \min\left\{s_k(\sigma) - s_j(\sigma) - p_j + s_m(\sigma) - s_l(\sigma) - p_l, 1\right\} \\ &\geq \min\{s_l(\sigma) + p_l - s_j(\sigma) - p_j + s_j(\sigma) \\ &\qquad +p_j + p_i - s_l(\sigma) - p_l, 1\} \\ &\geq \min\{p_i, 1\} = 1. \end{aligned}$$

Поскольку рассмотрены все случаи взаимного расположения работ J_i, J_k и J_m в расписании σ, можно сделать вывод, что решение $\mathcal{L}$ удовлетворяет ограничению (4.20). Аналогично доказывается, что решение $\mathcal{L}$ удовлетворяет ограничению (4.21). □

4.2.3 Построение допустимого решения

В предыдущем параграфе установлено, что решение задачи LP является нижней оценкой на оптимальное решение задачи $P\infty(P2)|sct|C_{max}$. Таким образом, если все

переменные x_{ij} в решении задачи LP равны либо 0, либо 1, то полученное решение является оптимальным. К сожалению, решение задачи LP может содержать и дробные значения. В этом параграфе представим алгоритм 4.4, который по решению задачи LP строит допустимое расписание для задачи $P\infty(P2)|sct|C_{max}$, присваивая переменным x_{ij} одно из значений либо 0, либо 1. В дальнейшем дуга $(J_i, J_j) \in E$ называется *0-дугой*, если $x_{ij} = 0$ и *1-дугой*, если $x_{ij} = 1$. Кроме того, дуга $(J_i, J_j) \in E$ называется *настоящей 1-дугой*, если $x_{ij} = 1$ и $e_{ij} \geq \frac{1}{4}$. Формальное описание алгоритма приведено ниже.

Алгоритм 4.4

1: Положить
$$x_{ij} = \begin{cases} 0, & \text{если } e_{ij} < \frac{1}{4}, \\ 1, & \text{если } e_{ij} \geq \frac{1}{4}. \end{cases}$$
2: Построить подграф $G_0 = (V_0, E_0)$ из графа G удалением (настоящих) 1-дуг. Обозначим через A_i множество дуг, входящих в вершину J_i в графе G_0, и через A^i множество дуг, выходящих из вершины J_i в графе G_0.
3: Пометить все работы множества $V_0 \bigcap Z$ как просмотренные.
4: **while** существуют непросмотренные работы **do**
5: Найти работу J_i, все родители которой уже просмотрены.
6: **if** множество A_i содержит только 0-дуги **then**
7: положить $x_{ij} := 1$ для всех $(J_i, J_k) \in A^i$.
Назовем такие дуги *1-дугами первого типа.*
8: **if** множество A_i содержит по крайней мере одну 1-дугу **then**
9: положить $x_{ij} := 1$ для всех $(J_k, J_i) \in A_i$.
Назовем такие дуги *1-дугами второго типа.*
10: **while** существуют две последовательные 1-дуги второго типа **do**
11: для каждой пары 1-дуг второго типа (J_i, J_j) и (J_j, J_k), таких что A^k не содержит 1-дуг второго типа положить $x_{ij} = 0$.
12: Пусть $G' = (V, E')$ такое, что $E' = \{(J_i, J_j) \in E | x_{ij} = 0\}$. Назначить множество работ, лежащих в связной компоненте графа G' на отдельную машину.
13: Вычислить моменты начала работ
$$s_j = \begin{cases} 0, & \text{если } J_j \in Z, \\ \max_{J_i \in \Gamma^-(j)} s_i + p_i + x_{ij}\delta_{ij}, & \text{если } J_j \in V \setminus Z. \end{cases}$$
14: Положить $\sigma^h = \{(J_i, s_i), J_i \in V\}$.

Покажем, что решение, получаемое алгоритмом 4.4, является допустимым, и установим другие свойства полученного решения. Заметим, что согласно шагу 1 алгоритма 4.4 и ограничениям (4.19) и (4.20) каждая вершина в ориентированном графе G_0 имеет не больше двух входящих и двух выходящих дуг.

Лемма 26 *После завершения выполнения первого цикла* **while** *на шагах 4 – 9 алгоритма 4.4 в графе G_0 нет последовательных 0-дуг.*

Доказательство. Рассмотрим произвольную вершину $J_i \in V \setminus \{Z \cup U\}$. Когда она просматривается в цикле **while** на шагах 4 – 9 алгоритма, то либо все ее выходящие дуги становятся 1-дугами первого типа (шаг 7), либо все ее входящие дуги становятся 1-дугами второго типа (шаг 9). □

Лемма 27 *Если после завершения выполнения первого цикла* **while** *на шагах 4 – 9 алгоритма 4.4 дуга* (J_i, J_j) *– 1-дуга второго типа, то множество* A_i *содержит только 1-дуги.*

Доказательство. После выполнения первого цикла **while** все переменные, соответствующие дугам из A_i, имеют одинаковое значение. Если эти значения равны 0, то при рассмотрении вершины J_i дуга (J_i, J_j) стала 1-дугой , т.е. 1-дугой первого типа. Следовательно, значения всех переменных x_{ki}, соответствующих дугам из A_i, равны 1. □

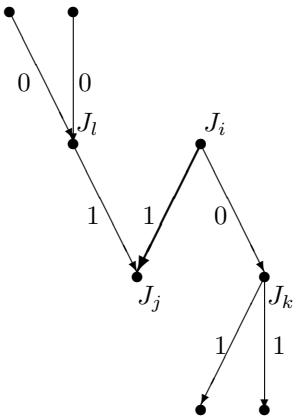

Рис. 4.3: Иллюстрация к лемме 28

Лемма 28 *Пусть* (J_i, J_j) *— 1-дуга второго типа. Если в графе* G_0 *существует дуга* (J_i, J_k)*, то после завершения выполнения первого цикла* **while** *на шагах 4 – 9 алгоритма 4.4 дуга* (J_i, J_k) *является 0-дугой. Более того, множество* A^k *либо пусто, либо содержит только 1-дуги первого типа.*

Доказательство. Пусть (J_i, J_j) — 1-дуга второго типа и в графе G_0 существует дуга (J_i, J_k). Так как (J_i, J_j) — 1-дуга второго типа, то в графе G_0 есть дуга (J_l, J_k), которая является 1-дугой первого типа. Следовательно, в A^k нет других дуг кроме (J_i, J_k), в противном случае граф, индуцированный работами J_i, J_j, J_k, J_l и J_m, был бы изоморфен графу на рисунке 4.1(b) и решение задачи LP не удовлетворяло бы ограничению (4.22). Описанная ситуация изображена на рисунке 4.3.

Докажем, что дуга (J_i, J_k) является 0-дугой. Действительно, дуга (J_i, J_k) не может быть 1-дугой второго типа, так как в множестве A_k нет других дуг кроме самой дуги

(J_i, J_k). Дуга (J_i, J_k) также не может быть 1-дугой первого типа, так как дуга (J_i, J_j) тогда тоже должна быть 1-дугой первого типа.

Заканчивая доказательство леммы, заметим, что согласно шагу 7 алгоритма 4.4 множество A^k либо пусто, либо содержит только 1-дуги первого типа. □

Лемма 29 *После завершения выполнения первого цикла* **while** *на шагах 4 – 9 алгоритма 4.4 граф, индуцированный 1-дугами второго типа, есть объединение цепей.*

Доказательство. Рассмотрим произвольную вершину J_i, в которую входит 1-дуга второго типа. Из ограничения (4.20) и определения 1-дуги второго типа следует, что в графе G_0 существует ровно одна другая входящая дуга, которая является 1-дугой первого типа. Следовательно, в любую вершину входит не больше одной 1-дуги второго типа. Кроме того из леммы 28 следует, что из произвольной вершины выходит не больше одной 1-дуги второго типа. □

Непосредственно из леммы 29 и шагов 10, 11 алгоритма 4.4 вытекает следующий результат.

Лемма 30 *После завершения выполнения второго цикла* **while** *на шагах 10, 11 алгоритма 4.4 в графе G_0 не существует двух последовательных 1-дуг второго типа.*

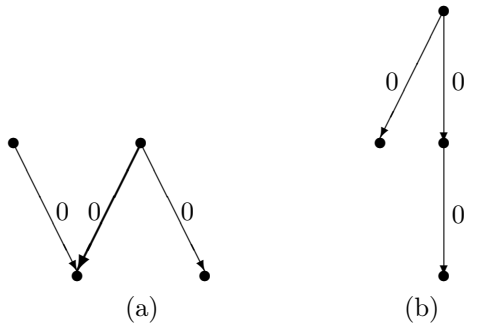

Рис. 4.4: Максимальные подграфы, индуцированные 0-дугами

Лемма 31 *После выполнения алгоритма 4.4 все максимальные связные подграфы в графе G, индуцированные 0-дугами, изоморфны графам, изображенным на рисунке 4.4.*

Доказательство. После выполнения шагов 1 и 2 алгоритма 4.4 в G_0 нет подграфов, изоморфных графам на рисунке 4.1, и алгоритм в процессе работы не изменяет значения настоящих 1-дуг. Кроме того, по лемме 26 после завершения выполнения первого цикла **while** на шагах 4 – 9 алгоритма 4.4 в графе G_0 нет последовательных 0-дуг. Таким образом, после завершения выполнения первого цикла **while** все подграфы, индуцированные 0-дугами, являются подграфами ориентированного графа, изображенного на рисунке 4.4(a).

На шагах 10, 11 алгоритма 4.4 только 1-дуга второго типа может стать 0-дугой. По лемме 29 перед выполнением этих шагов граф, индуцированный 1-дугами второго типа, является объединением цепей.

Пусть $(J_{i_1}, J_{i_2}, ..., J_{i_m})$ — такая цепь и $m \geq 2$. По условию работы алгоритма дуга $(J_{i_{m-1}}, J_{i_m})$ остается 1-дугой. Кроме того, алгоритм не создает двух последовательных 0-дуг внутри этой цепи. Пусть $(J_{i_{k-1}}, J_{i_k})$ для некоторого $k = 2, \ldots, m-1$ будет 0-дугой, полученной после завершения выполнения второго цикла **while**. Из леммы 27 следует, что множество A_{k-1} содержит только 1-дуги.

Предположим, что существует другая дуга $(J_{i_{k-1}}, J_l)$, выходящая из вершины $J_{i_{k-1}}$. По лемме 28 дуга $(J_{i_{k-1}}, J_l)$ – единственная дуга, входящая в вершину J_l в графе G_0, она является 0-дугой, и из вершины J_l не выходит ни одной 0-дуги.

Теперь рассмотрим дуги, инцидентные вершине J_{i_k}. Так как $k \leq m-1$, то у вершины J_{i_k} есть выходящая 1-дуга второго типа $(J_{i_k}, J_{i_{k+1}})$. Тогда по лемме 28, если в графе G_0 существует другая выходящая дуга (J_{i_k}, J_r), то она является 0-дугой, в графе G_0 в вершину J_r не входят другие дуги, и все выходящие дуги, если они есть, являются 1-дугами.

В итоге получим, что максимальный связный граф, индуцированный 0-дугами и содержащий вершины $J_{i_{k-1}}$ и J_{i_k}, изображен на рисунке 4.4(b).

Учитывая, что значения настоящих 1-дуг и 1-дуг первого типа в процессе работы алгоритма 4.4 не изменяются, получим утверждение леммы. □

Теорема 24 *Алгоритм 4.4 строит допустимое расписание.*

Доказательство. Согласно шагу 13 алгоритма каждая работа $J_i \in V \setminus z$ начинает выполняться в момент $\max_{J_k \in \Gamma^-(j)} s_k + p_k + x_{ki}\delta_{ki}$. При этом все работы, которые попали в связную компоненту графа G', и только они выполняются на одной машине. Из леммы 31 следует, что максимальная связная компонента в графе G' является подграфом одного из двух графов, изображенных на рисунке 4.4. Легко проверить, что все работы для таких подграфов можно без нарушения ограничений разместить на двух процессорах, так что время начала каждой работы J_i равно $\max_{J_k \in \Gamma^-(j)} s_k + p_k + x_{ki}\delta_{ki}$. Таким образом, моменты начала выполнения работ удовлетворяют ограничениям предшествования и заданным задержкам. □

4.2.4 Анализ точности

В этом разделе оценим точность решения, получаемого алгоритмом 4.4 для иерархических задач с малыми задержками. Для этого докажем, что для любой работы $J_i \in V$ момент начала ее выполнения $s_i^h \leq \frac{12(\Phi+1)}{12\Phi+1} t_i^*$, где t_i^* — момент начала выполнения работы J_i в оптимальном решении задачи линейного программирования *LP*. Напомним, что из ограничения (4.18) для каждой дуги $(J_j, J_i) \in E$ выполнено $t_i^* \geq t_j^* + p_j + e_{ij}\delta_{ij}$.

Сначала установим еще несколько свойств решения, получаемого алгоритмом 4.4.

Утверждение 12 *Если (J_i, J_j) — 1-дуга первого типа, то множество A_i содержит только 0-дуги, более того, A_i содержит по крайней мере одну 0-дугу.*

Утверждение напрямую следует из определения 1-дуги первого типа.

Лемма 32 *После выполнения алгоритма 4.4 в графе G длина любой цепочки, состоящей из двух 1-дуг первого или второго типа (ненастоящих 1-дуг), не превосходит два. Более того, если (J_i, J_j) и (J_j, J_k) — две последовательные 1-дуги с $e_{ij} < 0.25$ и $e_{jk} < 0.25$, то $V_i \notin Z$.*

Доказательство. Пусть (J_i, J_j) и (J_j, J_k) две последовательные 1-дуги с $e_{ij} < 0.25$ и $e_{jk} < 0.25$. Заметим, что (J_j, J_k) является 1-дугой второго типа. Действительно, если бы (J_j, J_k) была 1-дугой первого типа, то в силу утверждения 12 все ее родители были бы либо 0-дуги, либо настоящие 1-дуги. По лемме 30 в решении, построенном алгоритмом. нет двух последовательных 1-дуг второго типа. Следовательно, (J_i, J_j) — 1-дуга первого типа. Тогда по утверждению 12 получим, что у работы J_i нет входящих 1-дуг (J_l, J_i) с $e_{li} < 0.25$. Более того, вторая часть утверждения 12 влечет $V_i \notin Z$. □

Лемма 33 *Для каждой работы $J_i \in V$ выполнено $s_i^h \leq \frac{12(\Phi+1)}{12\Phi+1} t_i^*$.*

Доказательство. Докажем утверждение по индукции.

Неравенство справедливо для всех работ $J_j \in Z$. Определим критический путь в вершину J_i, как самый длинный ориентированный путь в эту вершину из произвольной вершины $J \in Z$. В данном случае под длиной пути будем понимать сумму длительностей всех работ и задержек всех дуг, принадлежащих этому пути. Заметим, что если дуга (J_j, J_i) принадлежит критическому пути, то

$$s_i^h = s_j^h + p_j + c_{ji} x_{ji}. \tag{4.27}$$

Предположим, что утверждение леммы справедливо для всех предшественников работы J_i. Обозначим множество всех ее родителей, которые принадлежат какому-либо критическому пути в вершину J_i, через $\Gamma_{cr}^-(i)$. Если среди них есть вершина $J_j \in \Gamma_{cr}^-(i)$ такая, что либо $x_{ji} = 0$, либо $x_{ji} = 1$ и $e_{ji} \geq 0.25$, то оценим s_i^h через s_j^h. Такая ситуация будет разобрана в случае 1. В противном случае имеем $x_{ji} = 1$ и $e_{ji} < 0.25$. Ситуация, когда $\Gamma^-(j) = \emptyset$, рассмотрена в случае 2. Если в $\Gamma_{cr}^-(j)$ есть работа J_k такая, что либо $x_{kj} = 0$, либо $x_{kj} = 1$ и $e_{kj} \geq 0.25$, то получаем случай 3. В противном случае на критическом пути есть две последовательные 1-дуги с $e_{kj} < 0.25$ и $e_{ji} < 0.25$. По лемме 32 в любом критическом пути им должна предшествовать либо 0-дуга, либо настоящая 1-дуга. Эта ситуация будет рассмотрена в случае 4.

Случай 1. (а) Пусть $x_{ji} = 0$. Так как (J_j, J_i) – критическая дуга, то из (4.27) получаем $s_i^h = s_j^h + p_j$. По индукционной гипотезе имеем $s_j^h \leq \frac{12(\Phi+1)}{12\Phi+1} t_j^*$ и, следовательно,

$$s_i^h \leq \frac{12(\Phi+1)}{12\Phi+1} t_j^* + p_j \leq \frac{12(\Phi+1)}{12\Phi+1} (t_i^* - p_j) + p_j \leq \frac{12(\Phi+1)}{12\Phi+1} t_i^*.$$

(b) Пусть $x_{ji} = 1$ и $e_{ji} \geq 0.25$, т.е. $t_i^* \geq t_j^* + p_j + 0.25\delta_{ij}$. Так как (J_j, J_i) – критическая дуга, то из (4.27) получаем $s_i^h = s_j^h + p_j + \delta_{ij}$. По индукционной гипотезе имеем $s_i^h \leq \frac{12(\Phi+1)}{12\Phi+1} t_j^* + p_j + \delta_{ij}$ и, следовательно,

$$s_i^h \leq \frac{12(\Phi+1)}{12\Phi+1} t_i^* + p_j \left(1 - \frac{12(\Phi+1)}{12\Phi+1}\right) + \delta_{ij} \left(1 - \frac{12(\Phi+1)}{4(12\Phi+1)}\right).$$

Так как $\max\limits_{(J_l,J_m)\in E} \delta_{lm} \leq 1$ и $p_j \geq \min\limits_{J_x \in V} p_x = \Phi \geq 1$, то получаем

$$s_i^h \leq \frac{12(\Phi+1)}{12\Phi+1} t_i^* + \Phi \left(1 - \frac{12(\Phi+1)}{12\Phi+1}\right) + \left(1 - \frac{12(\Phi+1)}{4(12\Phi+1)}\right)$$

$$\leq \frac{12(\Phi+1)}{12\Phi+1} t_i^* - \frac{11\Phi}{12\Phi+1} + \frac{9\Phi-2}{12\Phi+1} \leq \frac{12(\Phi+1)}{12\Phi+1} t_i^*.$$

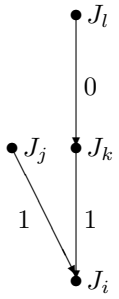

Рис. 4.5: Иллюстрация к случаю 2 из доказательства леммы 33

Случай 2. Пусть $x_{ji} = 1$, $e_{ji} < 0.25$ и $\Gamma^-(j) = \emptyset$, т.е. $J_j \in Z$. Тогда по утверждению 12 (J_j, J_i) — 1-дуга второго типа. Из этого следует, что существует дуга (J_k, J_i), которая является 1-дугой первого типа. Согласно шагу 7 алгоритма 4.4 и утверждению 12 существует 0-дуга (J_l, J_k), входящая в вершину J_k. Описанный подграф изображен на рисунке 4.5. Отношения предшествования заданные в таком графе дают две оценки на величину t_i^*: $t_i^* \geq p_j$ и $t_i^* \geq t_l^* + p_l + p_k \geq 2\Phi$. Так как (J_j, J_i) – критическая дуга и $J_j \in Z$, то из (4.27) получим $s_i^h = p_j + \delta_{ji} \leq \frac{p_j + \delta_{ji}}{\max\{2\Phi, p_j\}} t_i^*$. Завершая разбор случая 2, рассмотрим два подслучая.

(a) Если $2\Phi \geq p_j$, то

$$s_i^h \leq \frac{2\Phi+\delta_{ij}}{2\Phi} t_i^* \leq \frac{2\Phi+1}{2\Phi} t_i^* \leq \frac{12(\Phi+1)}{12\Phi+1} t_i^*.$$

(b) Если $2\Phi < p_j$, то

$$s_i^h \leq \frac{p_j+\delta_{ij}}{p_j} t_i^* \leq \frac{p_j+1}{p_j} t_i^* \leq \frac{2\Phi+1}{2\Phi} t_i^* \leq \frac{12(\Phi+1)}{12\Phi+1} t_i^*.$$

Случай 3. Пусть $x_{ji} = 1$, $e_{ji} < 0.25$ и $\Gamma^-(j) \neq \emptyset$.

(a) Пусть существует критическая дуга (J_k, J_j), которая является 0-дугой, тогда имеем $s_i^h = s_k^h + p_k + p_j + \delta_{ji}$ и $t_i^* \geq t_k^* + p_k + p_j$. По индукции получаем

$$\begin{aligned} s_i^h &\leq \tfrac{12(\Phi+1)}{12\Phi+1}(t_i^* - p_k - p_j) + \delta_{ji} + p_k + p_j \\ &\leq \tfrac{12(\Phi+1)}{12\Phi+1} t_i^* + \delta_{ji} + (p_k + p_j)\left(1 - \tfrac{12(\Phi+1)}{12\Phi+1}\right). \end{aligned}$$

Используя неравенство $12(\Phi+1)/(12\Phi+1) > 1$, получаем оценку

$$\begin{aligned} s_i^h &\leq \tfrac{12(\Phi+1)}{12\Phi+1} t_i^* + 1 + 2\Phi\left(1 - \tfrac{12(\Phi+1)}{12\Phi+1}\right) \\ &\leq \tfrac{12(\Phi+1)}{12\Phi+1} t_i^* + 1 - \tfrac{22\Phi}{12\Phi+1} \leq \tfrac{12(\Phi+1)}{12\Phi+1} t_i^*. \end{aligned}$$

(b) Пусть существует критическая дуга (J_k, J_j), которая является настоящей 1-дугой, тогда имеем $s_i^h = s_k^h + p_k + p_j + \delta_{ji} + \delta_{kj}$ и $t_i^* \geq t_k^* + p_k + p_j + 0.25\delta_{kj}$. Рассуждая аналогично предыдущему случаю, по индукции получаем

$$\begin{aligned} s_i^h &\leq \tfrac{12(\Phi+1)}{12\Phi+1} t_i^* + \delta_{ji} + \delta_{kj}\left(1 - \tfrac{3(\Phi+1)}{12\Phi+1}\right) + (p_k + p_j)\left(1 - \tfrac{12(\Phi+1)}{12\Phi+1}\right) \\ &\leq \tfrac{12(\Phi+1)}{12\Phi+1} t_i^* - \tfrac{22\Phi}{12\Phi+1} + \tfrac{21\Phi-1}{12\Phi+1} \leq \tfrac{12(\Phi+1)}{12\Phi+1} t_i^*. \end{aligned}$$

Случай 4. Осталось рассмотреть случай, когда в критическом пути есть две дуги (J_j, J_i) и (J_k, J_j) такие, что $e_{ji} < 0.25$ и $e_{kj} < 0.25$.

(a) Пусть (J_l, J_k) – критическая дуга, которая является 1 -дугой и $e_{lk} \geq 0.25$. Тогда $s_i^h = s_l^h + p_l + p_k + p_j + \delta_{lk} + \delta_{kj} + \delta_{ji}$ и $t_i^* \geq t_l^* + p_l + p_k + p_j + 0.25\delta_{lk}$. По индукции получаем

$$\begin{aligned} s_i^h &\leq \tfrac{12(\Phi+1)}{12\Phi+1}(t_i^* - p_l - p_k - p_j - 0.25\delta_{lk}) \\ &\quad + p_l + p_k + p_j + \delta_{lk} + \delta_{kj} + \delta_{ji} \\ &= \tfrac{12(\Phi+1)}{12\Phi+1} t_i^* + (p_l + p_k + p_j)\left(1 - \tfrac{12(\Phi+1)}{12\Phi+1}\right) \\ &\quad + \delta_{lk}\left(1 - \tfrac{3(\Phi+1)}{12\Phi+1}\right) + \delta_{kj} + \delta_{ji}. \end{aligned}$$

Используя неравенства $12(\Phi+1)/(12\Phi+1) > 1$ и $3(\Phi+1)/(12\Phi+1) < 1$, получаем оценку

$$\begin{aligned} s_i^h &\leq \tfrac{12(\Phi+1)}{12\Phi+1} t_i^* + 3\Phi\left(1 - \tfrac{12(\Phi+1)}{12\Phi+1}\right) + 3\left(1 - \tfrac{\Phi+1}{12\Phi+1}\right) \\ &\leq \tfrac{12(\Phi+1)}{12\Phi+1} t_i^* - \tfrac{33\Phi}{12\Phi+1} + \tfrac{33\Phi}{12\Phi+1} = \tfrac{12(\Phi+1)}{12\Phi+1} t_i^*. \end{aligned}$$

(b) Пусть (J_l, J_k) — критическая дуга, которая является 0 -дугой. Тогда $s_i^h = s_k^h + p_l + p_k + p_j + \delta_{kj} + \delta_{ji}$ и $t_i^* \geq t_k^* + p_l + p_k + p_j$. Заметим, что соотношение между величинами s_i^h и t_i^* меньше чем соотношение между теми же величинами в случае 4(a), а следовательно, полученная выше оценка верна

и для этого случая. □

Результат леммы 33 влечет оценки на точность алгоритма 4.4 для задач построения минимального по длине расписания на неограниченном числе параллельных машин с отношением предшествования между работами и малыми задержками.

Теорема 25 *Алгоритм 4.4 является* $12(\Phi+1)/(12\Phi+1)$*-приближенным алгоритмом для задачи* $P\infty(P2)|sct|C_{max}$.

В дополнение покажем, что описанный в разделе метод применим и к задаче минимизации взвешенной суммы моментов окончания работ. Задача $P\infty(P2)|sct|\sum w_jC_j$ отличается от задачи $P\infty(P2)|sct|C_{max}$ только целевой функцией.

Пусть LP' — задача линейного программирования, полученная заменой в задаче LP ограничения (4.25) на ограничение $t_i+p_i = C_i$ для всех $J_i \in V$, в которой требуется минимизировать величину $\sum w_jC_j$. Аналогично лемме 25 можно показать, что величина оптимального решения задачи LP' является оценкой на величину оптимального решения задачи $P\infty(P2)|sct|\sum w_jC_j$. Применим алгоритм 4.4 к оптимальному решению задачи LP'. Заметим, что алгоритм 4.4 строит расписание по решению задачи линейного программирования без использования вида целевой функции. Таким образом, для любой работы $J_i \in V$, момент начала ее выполнения $s_i \leq \frac{12(\Phi+1)}{12\Phi+1}t_i^*$, где t_i^* – момент начала выполнения работы J_i в оптимальном решении задачи линейного программирования LP'. Следовательно, получим следующий результат.

Теорема 26 *Алгоритм 4.4, примененный к оптимальному решению задачи* LP', *строит* $12(\Phi+1)/(12\Phi+1)$*-приближенное решение для задачи* $P\infty(P2)|sct|\sum w_jC_j$.

Глава 5

Маршрутизация машин в цеховых задачах открытого типа

В этой главе рассматривается цеховая задача открытого типа с маршрутизацией, которая является обобщением двух классических задач дискретной оптимизации: цеховой задачи открытого типа ($O||C_{\max}$) и метрической задачи коммивояжера (MTSP).

Задача $O||C_{\max}$

Заданы множество работ $\mathcal{J} = \{J_1, \ldots, J_n\}$, множество машин $\mathcal{M} = \{M_1, \ldots, M_m\}$. Каждая работа J_j состоит из m операций $O_{1,j}, O_{2,j}, \ldots, O_{m,j}$. Операция $O_{i,j}$ выполняется на машине M_i и ее длительность составляет $p_{i,j} \in \mathrm{Z}^+$ единиц времени. Операции каждой работы могут выполняться в произвольном порядке. Прерывания в процессе выполнения операций запрещены. Различные машины не могут выполнять операции одной работы одновременно, и каждая машина обрабатывает не более одной работы в каждый момент времени. Требуется минимизировать длину расписания. Для задачи $O||C_{\max}$ длина расписания совпадает с моментом завершения последней операции.

Задача MTSP

Задан неориентированный полный граф $G = (V, E)$. Вес $\tau_{ij} \in Z^+$ ребра $e_{ij} = [v_i, v_j]$ определяет расстояние между вершинами v_i и v_j. Расстояния удовлетворяют неравенству треугольника. Требуется найти гамильтонов цикл R (обход) минимального веса $|R| = \sum_{e_{ij} \in R} \tau_{ij}$.

Гамильтонов цикл в полном графе задается последовательностью вершин, в которой каждая вершина кроме первой появляется ровно один раз и последняя вершина совпадает с первой. Задача MTSP является NP-трудной в сильном смысле [93].

Цеховая задача открытого типа с маршрутизацией

Предположим, что работы разбиты на K групп, и каждая группа $\mathcal{J}_k$, $k = 0, \ldots, K$ расположена в узле v_k транспортной сети G. Для выполнения работы $J_j \in \mathcal{J}_k$ каждая машина должна переместиться в узел v_k. Изначально все машины расположены в одной вершине v_0 и должны вернуться в нее после выполнения всех работ. Все машины перемещаются с одинаковой скоростью и затрачивают время τ_{jk} на перемещение из вершины v_j в вершину v_k. Расстояния удовлетворяют неравенству треугольника.

Все машины могут начать движение или обслуживание работ с момента времени 0. Требуется найти минимальное по длине расписание для перемещения машин и выполнения работ.

Обязательные задержки между последовательным выполнением различных операций на одной машине или одной работы обычно связывают с переналадкой машины или транспортировкой работы. В общем случае время переналадки θ_{ijk} зависит как от машины M_i (работы), так и от обеих операций $O_{i,j}$ и $O_{i,k}$. В цеховой задаче открытого типа с маршрутизацией переналадки удовлетворяют следующим дополнительным ограничениям: $\theta_{ijk} = \theta_{jk} = \theta_{kj}$ и $\theta_{jk} \leq \theta_{jt} + \theta_{tk}$. Согласно системе обозначений, введенной в [99], для записи получающейся задачи необходимо внести оба условия в поле β, что сделает обозначения громоздкими и плохо читаемыми. Поэтому введем для рассматриваемых задач новые обозначения. Цеховую задачу открытого типа с маршрутизацией обозначим через $RO||\widetilde{C}_{max}$ или $ROm||\widetilde{C}_{max}$, если число машин фиксировано и равно m. Запись $|V| = k$ в поле β означает, что число вершин в графе G фиксировано и равно k. Заметим, что если в задаче нет ограничения на число вершин в графе G, то без ограничения общности можно считать, что каждая работа J_j расположена в отдельной вершине v_j, т.е. число групп равно n, и каждая группа содержит ровно одну работу.

Задачи теории расписаний с маршрутизацией привлекают повышенный интерес исследователей в последние 20 лет. Большинство работ в этом направлении фокусируется на одностадийных моделях [83, 87, 121]. Цеховые задачи, в которых машины перемещаются между работами, расположенными в узлах транспортной сети, впервые были рассмотрены в [38, 39, 40, 41]. Примеры задач, в которых машины должны перемещаться между деталями, возникают при обработке тяжелых или громоздких деталей или при построении расписания работы роботов, которые осуществляют ежедневное техническое обслуживание неподвижных объектов, расположенных в различных частях цеха [39]. Цеховая задача открытого типа с маршрутизацией также возникла при автоматическом планировании маршрутов экскурсий в Национальном Дворцовом музее в городе Тайбэй, входящем в пятерку крупнейших музеев мира [73, 167].

Цеховые задачи открытого типа с маршрутизацией впервые рассматривались в работах Авербаха, Бермана и Черных [40, 41]. Задача $RO||\widetilde{C}_{\max}$ является NP-трудной в сильном смысле, даже если работы требуется выполнить одной машиной или все работы лежат в одной вершине. В первом случае из задачи $RO||\widetilde{C}_{\max}$ получается задача MTSP, во втором случае — задача $O||C_{\max}$. Более того, в [41] показано, что задача $RO2||V| = 2|\widetilde{C}_{\max}$ с двумя машинами на двухвершинном графе NP-трудна в обычном смысле. В [40] установлено, что длина оптимального расписания лежит в интервале $[\bar{C}_R, \frac{6}{5}\bar{C}_R]$, где $\bar{C}_R$ — тривиальная нижняя оценка, точная формулировка которой будет приведена в следующем разделе. Там же приведен линейный $\frac{6}{5}$-приближенный алгоритм решения задачи $RO2||V| = 2|\widetilde{C}_{\max}$. Для задачи на двух машинах и произвольной транспортной сети ($RO2||\widetilde{C}_{\max}$) в [41] представлен $\frac{7}{4}$-приближенный алгоритм, и там же для общего случая ($RO||\widetilde{C}_{\max}$) предложен простой $\frac{m+4}{2}$-приближенный алгоритм.

В разделе 5.1 представлен $O(\sqrt{m})$-приближенный алгоритм (алгоритм 5.3) для задачи $RO||\widetilde{C}_{\max}$. Алгоритм использует комбинаторные свойства рассматриваемой задачи и имеет маленькую трудоемкость. В следующем разделе доказывается существование $O(\log m)$-приближенного алгоритма для той же задачи. Основная идея состоит в сведении задачи $RO||\widetilde{C}_{\max}$ к цеховой задаче рабочего типа $J|p_{ij}=1|C_{\max}$ с одинаковыми длительностями операций. Результат носит теоретический характер, поскольку все известные алгоритмы для задачи $J|p_{ij}=1|C_{\max}$ используют алгоритмический вариант леммы Ловаша, и их трудоемкость ограничена полиномом высокой степени от размера входа задачи. В третьем, четвертом и пятом разделах рассматривается двухмашинная цеховая задача открытого типа с маршрутизацией. В разделе 5.3 для общего случая двухмашинной задачи $RO2||\widetilde{C}_{\max}$ предложен (13/8)-приближенный алгоритм. В следующем разделе для той же задачи приведен (4/3)-приближенный алгоритм для случая, когда метрическая задача коммивояжера на сети может быть решена за полиномиальное время. И наконец, в разделе 5.5 для двухмашинной задачи на двухвершинной сети построен точный псевдополиномиальный алгоритм и вполне приближенная полиномиальная схема.

5.1 Произвольное число машин. $O(\sqrt{m})$-Приближенный алгоритм

В этом разделе представим $O(\sqrt{m})$-приближенный алгоритм для задачи $RO||\widetilde{C}_{\max}$.

Обозначим через d_j длину работы $J_j \in \mathcal{J}$ и через ℓ_i загрузку машины M_i, т.е. $d_j = \sum_{i=1}^{m} p_{ij}$ и $\ell_i = \sum_{j=1}^{n} p_{ij}$. Пусть $d_{\max} = \max_{J_j \in \mathcal{J}} d_j$, $\delta_{\max} = \max_{J_j \in \mathcal{J}}(d_j + 2\tau_{0j})$, и $\ell_{\max} = \max_i \ell_i$. Обозначим через OPT длину оптимального расписания и через T^* длину оптимального тура в графе G.

Так как каждая машина выполняет операции всех работ из $\mathcal{J}$, она должна посетить все вершины в графе G. Отсюда получим нижнюю оценку длины расписания

$$\bar{C}_R \doteq \max\{\ell_{\max} + T^*, \delta_{\max}\} \leqslant OPT. \tag{5.1}$$

Из оценки 5.1 видно, что для построения хорошего расписания необходимо найти хороший обход всех вершин в графе G, т.е. решить метрическую задачу коммивояжера. Как упоминалось выше, задача MTSP является NP-трудной в сильном смысле. Наилучший по точности приближенный алгоритм из известных на данный момент был независимо разработан Кристофидисом [74] и Сердюковым [20]. Для полноты изложения ниже приведем этот алгоритм (алгоритм 5.1).

Общий вес ребер в H не превышает T^*, и вес паросочетания M не превышает $\frac{T^*}{2}$. Из неравенства треугольника следует, что

$$|R_{CS}| \leqslant \frac{3}{2}T^* \ . \tag{5.2}$$

Время работы алгоритма $\mathcal{A}_{CS}$ определяется временем, требуемым для нахождения паросочетания минимального веса, и оценивается через $O(n^3)$ элементарных операций [88].

Алгоритм 5.1 Алгоритм Кристофидиса-Сердюкова

1: Найти в графе G остовное дерево H минимального веса.
2: Пусть X – множество вершин, имеющих нечетную степень в H, и пусть G_X – индуцированный подграф в G. Найти в графе G_X совершенное паросочетание M минимального веса.
3: Найти эйлеров обход в мультиграфе $H \bigcup M$. Порядок вершин, в котором они появляются первый раз в эйлеровом обходе, определяет цикл R_{CS} в графе G.
4: Выдать цикл R_{CS}.

В свою очередь, для цеховой задачи открытого типа известен простой 2-приближенный алгоритм. Невыполненная операция в частичном расписании σ называется *доступной* в момент t, если в этот момент не выполняется другая операция той же самой работы. Расписание работ для задачи $O||C_{max}$ называется *плотным*, если машина простаивает тогда и только тогда, когда у нее нет доступных операций. Алгоритм, строящий плотное расписание, называется *жадным*. Для задачи $O||C_{max}$ известно, что длина любого плотного расписания не превышает величины $\ell_{\max} + d_{\max}$ [53]. Приведем жадный алгоритм для задачи $RO|\tau_{ij} = \tau|\widetilde{C}_{\max}$, т.е. цеховой задачи открытого типа с маршрутизацией и с одинаковыми расстояниями между вершинами. На каждом шаге алгоритма определено частичное расписание σ в интервале $[0, t)$, где t — момент времени, который увеличивается в ходе работы алгоритма. Заметим, что если в частичном расписании σ в момент t у машины нет доступных операций, то они могут появиться только в моменты завершения обслуживания операций на других машинах. В ходе работы жадного алгоритма будем хранить такие моменты в множестве $TIME$.

Алгоритм 5.2

1: Положить $t = 0$, $k = 1$ и $TIME = \{0\}$.
2: **while** существуют невыполненные работы **do**
3: **if** машина M_i простаивает и есть доступная операция O_{ij} **then**
4: Назначить операцию O_{ij} на машину M_i.
5: Положить $s_{ij} = t$.
6: Добавить момент $t_k = t + p_{ij} + \tau$ в множество $TIME$.
7: Положить $k = k + 1$.
8: Удалить t из множества $TIME$. Положить $t = \min_k t_k$.
9: Выдать расписание σ.

Так как в каждый момент времени в обслуживании находится не более m работ, нахождение доступной операции на каждой итерации требует $O(\min\{m, n\})$ элементарных операций. Отсюда общая трудоемкость алгоритма 5.2 оценивается временем $O(nm \min\{m, n\})$.

Лемма 34 *Алгоритм 5.2 строит для задачи* $RO|\tau_{ij}=\tau|\widetilde{C}_{\max}$ *плотное расписание* σ *длины не более* $\ell_{\max}+d_{\max}+n\tau$.

Доказательство. Предположим, что операция O_{ij} заканчивается последней в расписании σ. Тогда $\widetilde{C}_{\max}(\sigma)=\ell_i+n\tau+W_i$, где W_i – суммарное время простоя машины M_i в интервале $[0,\widetilde{C}_{\max}(\sigma)]$. В каждый момент времени, когда простаивает машина M_i, должна выполняться отличная от O_{ij} операция работы J_j. Следовательно, $W_i\le d_j$ и $\widetilde{C}_{\max}(\sigma)\le\ell_{\max}+d_{\max}+n\tau$. □

5.1.1 Приближенный алгоритм

В этом параграфе представим приближенный алгоритм, который для любого примера задачи $RO||\widetilde{C}_{\max}$ строит расписание длины $\widetilde{C}_{\max}\le O(\sqrt{m})\bar{C}_R$.

Основная идея алгоритма состоит в преобразовании исходного примера задачи $RO||\widetilde{C}_{\max}$ к примеру задачи $RO|\tau_{ij}=\tau|\widetilde{C}_{\max}$, нахождении для нового примера хорошего приближенного расписания и перестроении этого расписания в допустимое расписание для исходной задачи без увеличения его длины.

На первом шаге строится обход R в графе G, длина которого не более чем в 1.5 раза больше длины оптимального обхода в этом графе.

На следующем шаге исходное множество работ заменяется на $O(\sqrt{m})$ новых "комбинированных" работ. Каждая комбинированная работа расположена в отдельной вершине и объединяет несколько исходных работ, которые размещаются последовательно на некотором отрезке в обходе R. Работы, включенные в одну комбинированную работу, назовем ее *компонентами.* Время выполнения новой операции комбинированной работы на каждой машине равно сумме времен выполнения ее компонент на этой машине плюс длина пути, связывающая эти работы в R. Расстояние между любой парой вершин в новом примере равно максимальной длине ребра в исходном графе G.

Таким образом, получим пример задачи $RO|\tau_{ij}=\tau|\widetilde{C}_{\max}$. Решим этот пример алгоритмом 5.2. Полученное расписание преобразуем в допустимое расписание исходного примера задачи $RO||\widetilde{C}_{\max}$.

Ниже дадим формальное описание алгоритма 5.3. Алгоритм 5.3 использует два положительных параметра α и β, точные значения которых будут установлены в следующем параграфе. Оператор $\oplus$ обозначает сочленение последовательностей.

Оценим трудоемкость алгоритма 5.3. На первом шаге алгоритм 5.3 использует алгоритм 5.1 для построения обхода R_{CS}. Напомним, что время работы алгоритма 5.1 равно $O(n^3)$, где n — число вершин. Следовательно, и трудоемкость первого шага оценивается той же величиной. Так как все исходные данные — целые числа, то первое и второе условие в цикле **while** на шаге 6 легко проверить, заменив указанные неравенства на эквивалентные неравенства с целыми числами в левой и правой частях. Время работы алгоритма 5.2, применяемого на шаге 15, равно $O(nm\min\{m,n\})$. Трудоемкость все остальных шагов не превышает $O(mn)$ и, следовательно, общая

Алгоритм 5.3

1: Используя алгоритм 5.1, найти в графе G обход R_{CS}, длина которого $T \leqslant \frac{3}{2}T^*$.
2: Занумеровать в графе G вершины по порядку их появления в обходе R_{CS}, начиная с вершины v_0, в которой располагаются машины до начала обслуживания работ.
3: Положить $d_0 := 0$, $i := 0$, $q := 0$ и доопределить $\tau_{n,n+1} := \tau_{0n}$.
4: **while** $q \leqslant n$ **do**
5: $\quad i := i + 1$; $P_i := \emptyset$.
6: $\quad$ **while** $\sum\limits_{v_j \in P_i} d_j < \alpha\sqrt{m}\,\ell_{\max}$ **and** $\sum\limits_{v_j \in P_i} \tau_{j,j+1} < \dfrac{\beta T}{\sqrt{m}}$ **and** $q \leqslant n$ **do**
7: $\quad\quad P_i := P_i \oplus (v_q)$; $q := q + 1$.
8: Положить $\tau := \max_{j',j''} \tau_{j'j''}$.
9: **for** $j = 1, \ldots, k$ **do**
10: $\quad$ **for** $i = 1, \ldots, m$ **do**
11: $\quad\quad p'_{ji} := \sum\limits_{v_h \in P_j} p_{hi} + \sum\limits_{e_{hh'} \in P_j} \tau_{hh'}$.
12: **for** $i = 1, \ldots, m$ **do**
13: $\quad p'_{0i} := 0$.
14: Определить пример I задачи $O|r_{ij} = \tau|C_{\max}$ с $k + 1$ работой и m машинами. Длительность работы J'_j, $j = 0, \ldots, k$, на машине M_i, $i = 1, \ldots, m$, равна p'_{ji}.
15: Алгоритмом 5.2 найти расписание σ для примера I.
16: Преобразовать расписание σ в допустимое расписание σ' для исходного примера задачи $RO||\widetilde{C}_{\max}$. Пусть $v_{q_1}, \ldots, v_{q_r}$ — последовательность вершин в пути P_j в порядке их появления в R_{CS}.
17: **for** $h = 1, \ldots, r$ **do**
18: $\quad$ **for** $i = 1, \ldots, m$ **do**
19: $\quad\quad$ положить $s_{q_h i}(\sigma') := s_{ji}(\sigma) + \sum\limits_{l=1}^{h-1} p_{q_l i} + \tau(h-1)$.
20: Выдать расписание σ'.

трудоемкость алгоритма 5.3 определяется первым шагом и равна $O(n^3)$. Заметим, что время работы алгоритма 5.3 может быть снижено до $O(n^2)$, если на первом шаге строить обход широко известным алгоритмом двойного обхода минимального остовного дерева, который строит обход длины не больше $2T^*$.

5.1.2 Анализ точности алгоритма 5.3

Заметим, что для каждого пути P_i, $i \neq k$ выполнено либо неравенство $\sum\limits_{v_j \in P_i} d_j \geqslant \alpha\sqrt{m}\,\ell_{\max}$, либо неравенство $\sum\limits_{v_j \in P_i} \tau_{j,j+1} \geqslant \frac{\beta T}{\sqrt{m}}$. Так как общая загрузка всех машин не превышает величины $m\ell_{\max}$, то существует не более $\lfloor \frac{\sqrt{m}}{\alpha} \rfloor$ путей с $\sum\limits_{v_j \in P_i} d_j \geqslant \alpha\sqrt{m}\,\ell_{\max}$.

Заметим, что $\sum_{i=1}^{k}\sum_{v_j\in P_i}\tau_{j,j+1}=T$. Отсюда следует, что существует не более $\lfloor\frac{\sqrt{m}}{\beta}\rfloor$ путей с $\sum\limits_{v_j\in P_i}\tau_{j,j+1}\geqslant\dfrac{\beta T}{\sqrt{m}}$. Таким образом, количество всех путей можно оценить как $k\leqslant\lfloor\frac{\sqrt{m}}{\alpha}\rfloor+\lfloor\frac{\sqrt{m}}{\beta}\rfloor+1$.

Пусть $d'_{\max}$ — длина максимальной работы и $\ell'_{\max}$ — максимальная загрузка машины в примере I, построенном на шаге 14 алгоритма 5.3. Согласно шагу 6 имеем

$$d'_j=\sum_{i=1}^{m}p'_{ji}=\sum_{v_h\in P_j}d_h+m\sum_{e_{hh'}\in P_j}\tau_{hh'}\leqslant\alpha\sqrt{m}\,\ell_{\max}+d_{\max}+m\frac{\beta T}{\sqrt{m}}$$

и, следовательно,

$$d'_{\max}\leqslant\alpha\sqrt{m}\,\ell_{\max}+d_{\max}+\beta\sqrt{m}\,T\ .$$

По определению загрузка каждой машины M_i в примере I равна

$$\ell'_i=\sum_{j=1}^{k}p'_{ij}=\sum_{j=1}^{k}\left(\sum_{v_h\in P_j}p_{ih}+\sum_{e_{hh'}\in P_j}\tau_{hh'}\right)\leqslant\ell_i+T\ .$$

Отсюда следует, что $\ell'_{\max}\leqslant\ell_{\max}+T$ и $\gamma_{\max}=\ell'_{\max}+(k+1)\tau$. Лемма 34 влечет

$$F_{\max}(\sigma)\leqslant d'_{\max}+\ell'_{\max}+(k+1)\tau\leqslant\alpha\sqrt{m}\,\ell_{\max}+d_{\max}+\beta\sqrt{m}\,T + \ell_{\max}+T+\left(\left\lfloor\frac{\sqrt{m}}{\alpha}\right\rfloor+\left\lfloor\frac{\sqrt{m}}{\beta}\right\rfloor+2\right)\tau\ .$$

С учетом неравенств $T\leqslant\frac{3}{2}T^*$ и $\tau\leqslant\frac{1}{2}T^*$ имеем

$$F_{\max}(\sigma)\leqslant\alpha\sqrt{m}\,\ell_{\max}+\left(\frac{3\beta}{2}+\frac{1}{2\alpha}+\frac{1}{2\beta}\right)\sqrt{m}\,T^*+d_{\max}+\ell_{\max}+\frac{5}{2}T^*\ .$$

Полагая $\alpha=\frac{\sqrt{2}}{2}$ и $\beta=\frac{\sqrt{3}}{3}$, получим оценку

$$F_{\max}(\sigma)\leqslant\left((\sqrt{3}+\sqrt{2})\sqrt{m}+3.5\right)\bar{F}\ .$$

Теорема 27 *Алгоритм 5.3 является $O(\sqrt{m})$-приближенным алгоритмом для задачи $RO||\widetilde{C}_{\max}$.*

5.2 Произвольное число машин. $O(\log m)$-приближенный алгоритм

В предыдущем разделе представлен приближенный алгоритм для задачи $RO||\widetilde{C}_{\max}$ с гарантированной оценкой точности $O(\sqrt{m})$. Время работы алгоритма оценивается полиномом второй или третьей степени от числа работ в зависимости от трудоемкости алгоритма, которым находится приближенное решение задачи коммивояжера. На данный момент неизвестно другого строго полиномиального алгоритма, который

имеет лучшую оценку точности. Однако, в [168] был предложен $O(\log m(\log\log m)^{1+\varepsilon})$-приближенный алгоритм, основанный на сведении исходной задачи к классической цеховой задаче потокового типа. Оценка точности следует из результатов Шмойса, Штайна и Вайна [158], которые предложили $O(\log m(\log\log m)^{1+\varepsilon})$-приближенный рандомизированный алгоритм для цеховой задачи потокового типа $F||C_{\max}$.

В этом разделе представим $O(\log m)$-приближенный алгоритм, основанный на сведении задачи $RO||\widetilde{C}_{\max}$ к цеховой задаче рабочего типа с единичными длительностями операций. Для решения последней задачи также известны различные приближенные алгоритмы [134, 145], которые находят расписания, длина которых не более чем в ρ раз больше нижней оценки. К сожалению, как и алгоритм Шмойса, Штайна и Вайна, они имеют огромную, хотя и теоретически полиномиальную, трудоемкость и мало реализуемы на практике. Поэтому результаты этого раздела носят теоретический характер.

Сначала приведем неформальное описание алгоритма. Как и в алгоритме 5.3, используя алгоритм 5.1, найдем в графе G обход R_{CS}, длина которого $T \leqslant \frac{3}{2}T^*$. Затем заменим исходное множество работ $\mathcal{J}$ на $\min\{2m, n\}$ новых *агрегированных* работ (А-работ). Каждая агрегированная работа состоит из нескольких исходных работ, последовательно расположенных на сегменте тура R_{CS}, которые будем называть ее *компонентами*. Время выполнения операции новой агрегированной работы равно сумме времен выполнения операций ее компонент на этой машине. Операции агрегированной работы назовем А-операциями. Таким образом, исходный пример I задачи $RO||C_{\max}$ трансформируется в пример I' задачи $O||C_{\max}$ с тем же множеством машин, но с $n' \leq 2m$ работами, с $p'_{ij} \leq d'_j \leq \bar{C}$, где p'_{ij} обозначает длительность А-операций O'_{ij} и d'_j обозначает длину агрегированной работы J'_j. Заметим, что в построенном примере цеховой задачи открытого типа время передвижения машин не учитывается.

Пусть p'_{max} — длительность наибольшей А-операции в примере I'. Используя стандартную технику масштабирования [158], изменим длительности операций так, что $p'_{max} \leq 2m^2$ и все длительности операций — целые числа. Затем округлим вверх время выполнения каждой операции до ближайшей степени числа 2. В новом примере I'' будет не более $\lceil \log_2 p'_{max} \rceil$ различных длительностей операций. Разобьем пример I'' на не более $\lceil \log_2 p'_{max} \rceil$ примеров, в которых все А-операции имеют одинаковую длину.

Пусть I_l, $1 \leq l \leq \lceil \log_2 p'_{max} \rceil$ — один из таких примеров. Поменяем роль машин и работ и найдем порядок выполнения операций, которые принадлежат каждой новой работе. В итоге получим пример I'_l ациклической цеховой задачи рабочего типа с единичными длительностями операций. Применив для последней задачи алгоритм Лейтона-Маггс-Рича [134], получим расписание σ'_l длины не более $O(\bar{C})$.

На последнем этапе преобразуем расписание σ'_l в частичное допустимое расписание σ_l примера I_l, т.е. допустимое расписание примера I_l задачи $RO||C_{\max}$ на соответствующем подмножестве операций. Пусть А-операция занимает некоторый интервал в расписании σ'_l. Поместим операции соответствующих компонент в данный интервал в том же порядке, в котором они появлялись в обходе R. Повторим эту процедуру для всех А-операций в σ'_l. Затем сдвинем время начала каждой операции на дли-

ну пути в обходе R из вершины v_0 в вершину, в которой должна быть выполнена эта операция. Получим допустимое частичное расписание σ_l примера I_l такое, что длина расписания σ_l не превышает $\rho\bar{C} + |R|$, где ρ — некоторая константа. Пусть расписание $\bar{\sigma}$ получено объединением всех частичных расписаний. Из неравенства $\lceil \log_2 p'_{max} \rceil \leq 2m^2$ получим, что длина расписания $\bar{\sigma}$ не более чем в $O(\log m)$ раз хуже оптимальной.

Алгоритм 5.4

1: Используя алгоритм 5.1, найти в графе G обход R_{CS}, длина которого $T \leqslant \frac{3}{2}T^*$.
2: Разбить обход R на непересекающиеся пути $P_1, \ldots, P_k$, количество путей k будет определено в конце разбиения.
3: Положить $i := 0;\ q := 0$.
4: **while** $q \leqslant n$ **do**
5: Положить $i := i + 1;\ P_i := \emptyset$.
6: **while** $\sum_{v_j \in P_i} d_j \leq \ell_{\max}$ **and** $q \leqslant n$ **do**
7: Положить $P_i := P_i \oplus v_j;\ q := q + 1$
8: Определить пример I' задачи $O||C_{\max}$ с k работами на m машинах.
9: Для каждого пути $P_j,\ j = 1, \ldots, k$ определим A-работу J'_j. Время выполнения работы J'_j на машине M_i положить $p'_{ji} \doteq \sum_{v_h \in P_j} p_{hi}$.
10: Вычислить $p'_{max} = \max_{ij} p'_{ji}$.
11: Положить ω равным общему числу операций в примере I'.
12: Положить $p''_{ji} := \max\{k \in \mathbb{Z} | \frac{kp'_{max}}{\omega} \leq p'_{ji}\}$.
13: Округлить вверх каждый p''_{ji} до ближайшей степени числа 2. Пусть L — число различных длительностей операций после округления.
14: Для каждого $l = 1, \ldots, L$ определить пример I'_l цеховой задачи потокового типа с k машинами и m работами, т.е. операция O_{j+1i} может стартовать только после завершения операции O_{ji} для всех $i = 1, \ldots, m$ и $j = 1, \ldots, k-1$.
15: Длительность операции O_{ji} положить $p^{(l)}_{ji} = \lceil \frac{p''_{ji} p'_{max}}{\omega} \rceil$, если $p''_{ji} = 2^l$, и $p^{(l)}_{ji} = 0$ в противном случае.
16: Найти приближенное расписание σ'_l для каждого примера I'_l.
17: В каждом расписании σ'_l удалить все операции нулевой длины.
18: **for** каждой ненулевой операции O'_{ji} **do**
19: Пусть $J_{h+1} \ldots, J_{h+k}$ — компоненты работы J'_j.
20: Положить $q := 1, \tau := s_{ji}(\sigma'_l)$.
21: **while** $q \leq k$ **do**
22: Положить $s_{h+qi}(\sigma_l) := \tau;\ \tau := \tau + p_{h+qi};\ q := q + 1$.
23: **for** каждого расписания σ'_l и каждой операции O_{ji} **do**
24: Положить $s_{ji}(\sigma_l) := s_{ji}(\sigma'_l) + \lambda_j$
25: Выдать расписание $\bar{\sigma} = \sigma_1 \circ \sigma_2 \circ \cdots \circ \sigma_L$.

Время работы алгоритма 5.4 определяется выбором алгоритма на шаге 16. Как показано в [133], цеховая задача потокового типа, все ненулевые операции в которой имеют одинаковые длительности, является частным случаем задачи передачи

пакетов в сети с единичными пропускными способностями ребер и единичным временем передачи вдоль каждого ребра. Лейтон, Маггс и Рао [133] доказали существование протокола (расписания) передачи пакетов, длина которого линейно зависит от максимальной загрузки ребра и максимальной длины маршрута. В терминах цеховых задач это соответствует максимальной загрузке l_{max} и длительности максимальной работы d_{max}. В [134] Лейтон, Маггс и Рича для задачи передачи пакетов предложили алгоритм, который находит расписание длины $O(l_{max} + d_{max})$ за время $O(\omega'_l(\log\log\omega'_l)\log\omega'_l)$, где ω' — число операций в примере I'_l. Однако следует заметить, что константа перед выражением $l_{max} + d_{max}$ огромна. Недавно Пейс и Визе показали [145], что существует расписание длины не более $23.4(l_{max} + d_{max})$. Единственное неконструктивное место в доказательстве этого результата — это использование локальной леммы Ловаша. Мозер и Тардош [140] разработали общую концепцию алгоритмизации локальной леммы Ловаша и предложили рандомизированный алгоритм, основанный на этой концепции и позволяющий найти структуру, существование которой гарантируется в этой лемме. И наконец, Шандрасекаран, Гойал и Хоплер [69] на основе идей Мозера и Тардош разработали детерминированный полиномиальный алгоритм для использования локальной леммы Ловаша.

5.3 Две машины, произвольная сеть

В этом разделе будет представлен приближенный алгоритм для задачи $RO2||\widetilde{C}_{\max}$, в которой все работы должны быть выполнены на двух машинах. Далее в этой главе будем называть эти машины машиной A и машиной B. Напомним, что данная задача является NP-трудной в сильном смысле. В [41] для нее был предложен 7/4-приближенный алгоритм. Пусть задан порядок выполнения операций π_X на каждой машине $X \in \{A, B\}$. Предположим, что каждая машина обслуживает операции без задержек, т.е. момент начала выполнения очередной операции равен моменту завершения предыдущей операции на этой машине (или 0 для первой по порядку работы) плюс время, необходимое на перемещения машины из одной вершины в другую. Две перестановки π_A и π_B назовем *бесконфликтными*, если для всех работ интервалы выполнения их операций не пересекаются. Работы, у которых интервалы выполнения их операций пересекаются, назовем *конфликтными*. С учетом введенных терминов приведем ниже алгоритм из [41] (алгоритм 5.5).

Нетрудно заметить, что для любых исходных данных существует не более одной конфликтной работы относительно перестановок π_A и π_B. Пусть σ — расписание, построенное алгоритмом 5.5. В [41] показано, что если перестановки π_A и π_B, построенные алгоритмом 5.5, являются бесконфликтными, то

$$C_{\max}(\sigma) = |R| + \ell_{\max}. \tag{5.3}$$

В противном случае, длину построенного расписания можно оценить через длину конфликтной работы J_c

$$C_{\max}(\sigma) \leqslant |R| + \ell_{\max} + \frac{1}{2}d_c\ . \tag{5.4}$$

Алгоритм 5.5 Алгоритм Авербаха-Берман-Черных [41]

1: Найти в графе G обход R.
2: Упорядочить перестановку π_A работ на машине A в порядке появления этих работ в R_{CS}.
3: Упорядочить перестановку π_B работ на машине B в порядке, обратном появлению этих работ в R_{CS}.
4: **if** перестановки π_A и π_B бесконфликтные **then**
5: выдать раннее расписание σ, в котором машины выполняют работы согласно перестановкам π_A и π_B.
6: **else**
7: Найти конфликтную работу J_j.
8: Построить раннее расписание σ_1, в котором работа J_j выполняется сначала на машине A, и машины выполняют работы согласно перестановкам π_A и π_B.
9: Построить раннее расписание σ_2, в котором работа J_j выполняется сначала на машине B, и машины выполняют работы согласно перестановкам π_A и π_B.
10: Выдать кратчайшее из расписаний σ_1 и σ_2.

В частности, если на первом шаге алгоритма 5.5 используется алгоритм 5.1, то для случая бесконфликтных перестановок получим

$$C_{\max}(\sigma) \leqslant \frac{3}{2}T^* + \ell_{\max} \leq \frac{3}{2}OPT, \tag{5.5}$$

а если конфликтная работа J_c существует, то

$$C_{\max}(\sigma) \leqslant \frac{3}{2}T^* + \ell_{\max} + \frac{1}{2}d_c\ . \tag{5.6}$$

Если $\ell_{\max} \geqslant d_c$, то из (5.6) и (5.1) получим $C_{\max}(\sigma) \leqslant \frac{3}{2}OPT$.
В противном случае можно доказать, что $C_{\max}(\sigma) \leqslant \frac{7}{4}OPT$.

5.3.1 Приближенный алгоритм

Для построения нового приближенного алгоритма для задачи $RO2||\widetilde{C}_{\max}$ будет использоваться модифицированный алгоритм Кристофидиса-Сердюкова. Пусть J_j — работа с $d_{\max} = d_j$. Наша цель — найти обход, который содержит ребро между вершинами v_0 и v_j. Для этого добавим к минимальному остовному дереву требуемое ребро и удалим одно из ребер дерева, чтобы разрушить цикл.

Из неравенства треугольника следует, что длина цикла R не превышает общего веса ребер в H плюс сумма весов ребер в M и вес ребра e_{0j}. Отсюда получаем, что $|R| \leq \frac{3}{2}T^* + \tau_{0j}$.

Алгоритм 5.6

1: Найти в графе G остовное дерево H минимального веса.
2: **if** $e_{0j} \notin H$ **then**
3: Добавить ребро e_{0j} к H и удалить любое отличное от e_{0j} ребро из образовавшегося цикла.
4: Пусть X – множество вершин, имеющих нечетную степень в H, и пусть G_X – индуцированный подграф в G. Найти в графе G_X совершенное паросочетание M минимального веса.
5: Найти эйлеров обход в мультиграфе $H \bigcup M$. Порядок вершин, в котором они появляются первый раз в эйлеровом обходе, определяет цикл R в графе G.
6: Выдать цикл R.

Пусть $J_j \in \mathcal{J}$ — работа, длина которой максимальна. Обозначим через R_1 цикл в графе G, найденный алгоритмом 5.1, и через R_2 цикл в графе G, найденный алгоритмом 5.6 и содержащий ребро e_{0j}. Алгоритм 5.7 выбирает лучшее из расписаний, построенных алгоритмом 5.5 с использованием циклов R_1 и R_2.

Алгоритм 5.7

1: Найти в графе G гамильтонов цикл R_1 алгоритмом 5.1.
2: Используя цикл R_1, построить расписание σ_{R_1} алгоритмом 5.5.
3: Найти работу J_j максимальной длины.
4: Найти в графе G гамильтонов цикл R_2, который содержит ребро e_{0j}, алгоритмом 5.6.
5: Используя цикл R_2, построить расписание σ_{R_2} алгоритмом 5.5.
6: Выдать кратчайшее из расписаний σ_{R_1} и σ_{R_2}.

В следующем параграфе оценим длину расписания, получаемого алгоритмом 5.7

5.3.2 Анализ точности алгоритма 5.7

Пусть $C_{\max}(\sigma) := \min\{C_{\max}(\sigma_{R_1}), C_{\max}(\sigma_{R_2})\}$ — длина расписания σ, построенного алгоритмом 5.7. Из ограничения (5.6) имеем

$$C_{\max}(\sigma_{R_1}) \leqslant \frac{3}{2}T^* + \ell_{\max} + \frac{1}{2}d_j. \tag{5.7}$$

Если $\ell_{\max} \geqslant d_{\max}$, то (5.7) влечет

$$C_{\max} \leqslant C_{\max}(\sigma_{R_1}) \leqslant \frac{3}{2}OPT\ .$$

Предположим, что $\ell_{\max} < d_{\max}$. Рассмотрим три следующих случая.

Случай 1. Если при построении расписания σ_{R_2} не было конфликтной работы, то

$$C_{\max}(\sigma_{R_2}) \leqslant \frac{3}{2}T^* + \tau_{0j} + \ell_{\max}. \tag{5.8}$$

Случай 2. Пусть J_j — конфликтная работа в расписании σ_{R_2}. В этом случае алгоритм 5.5 построит два расписания, назовем их $\sigma_{R_2'}$ и $\sigma_{R_2''}$, и выберет из них лучшее. Пусть в расписании $\sigma_{R_2''}$ машина B простаивает, ожидая, пока машина A закончит обрабатывать работу J_j, и в расписании $\sigma_{R_2'}$ простаивает машина A. В первом расписании машина B закончит работу в момент $d_j + 2\tau_{0j}$, а машина A, которая работает без простоев, закончит работу в момент $|R_1| + \tau_{0j} + \ell_1 \leqslant \frac{3}{2}T^* + \tau_{0j} + \ell_{\max}$. Следовательно, имеем

$$C_{\max}(\sigma_{R_2}) \leqslant \max\left\{d_j + 2\tau_{0j}, \frac{3}{2}T^* + \tau_{0j} + \ell_{\max}\right\} \tag{5.9}$$

независимо от того, какое расписание будет выбрано алгоритмом 5.5.

Заметим, что первое выражение в (5.9) является нижней оценкой на длину расписания. Предположим, что расписание σ_{R_2} не является оптимальным. Тогда получим оценку (5.8) на длину расписания.

Случай 3. Пусть J_i, $i \neq j$, была конфликтной работой в расписании σ_{R_2}. Тогда из (5.4) получим

$$C_{\max}(\sigma_{R_2}) \leqslant \frac{3}{2}T^* + \tau_{0j} + \ell_{\max} + \frac{d_i}{2}. \tag{5.10}$$

Сравнивая оценки (5.8), (5.9) и (5.10), получим

$$C_{\max}(\sigma_{R_2}) \leqslant \frac{3}{2}T^* + \tau_{0j} + \ell_{\max} + \frac{d_{i(\neq j)}}{2}, \tag{5.11}$$

где

$$d_{i(\neq j)} = \begin{cases} d_i, & i \neq j, \\ 0, & i = j. \end{cases}$$

Далее учитывая (5.7), (5.11), неравенство $d_j + d_{i(\neq j)} \leqslant 2\ell_{\max}$ и (5.1), получим

$$4C_{\max}(\sigma) \leqslant 3C_{\max}(\sigma_{R_1}) + C_{\max}(\sigma_{R_2}) \leqslant 3\left(\frac{3}{2}T^* + \ell_{\max} + \frac{1}{2}d_j\right)$$

$$+\frac{3}{2}T^* + \tau_{0j} + \ell_{\max} + \frac{d_{i(\neq j)}}{2} = 6T^* + 4\ell_{\max} + \tau_{0j} + \frac{3}{2}d_j + \frac{d_{i(\neq j)}}{2}$$

$$= 4(T^* + \ell_{\max}) + \frac{2\tau_{0j} + d_j}{2} + \left(T^* + \frac{d_j + d_{i(\neq j)}}{2}\right) + \left(T^* + \frac{d_j}{2}\right) \leqslant \frac{13}{2}OPT\ .$$

Тем самым окончательно имеем

$$C_{\max} = \min\{C_{\max}(\sigma_{R_1}), C_{\max}(\sigma_{R_2})\} \leqslant \frac{13}{8}OPT.$$

Заметим, что время работы алгоритмов 5.1 и 5.6 равно $O(n^3)$, а расписания σ_{R_1} и σ_{R_2} можно построить за линейное время. В итоге приходим к следующему результату.

Теорема 28 *Алгоритм 5.7 является* 1.625*-приближенным алгоритмом для задачи* $RO2||\widetilde{C}_{\max}$. *Время его работы равно* $O(n^3)$.

5.4 Две машины, легкий пример задачи коммивояжера

Хотя метрическая задача коммивояжера NP-трудна для общего случая, известно много примеров, когда она может быть решена точно за полиномиальное время. Например, полиномиально разрешимыми являются примеры, когда матрица расстояний является циркулянтной или удовлетворяет условиям Демиденко. Другой полиномиально разрешимый класс задач возникает на транспортных сетях, которые имеют древесную структуру. Более подробно о полиномиально разрешимых случаях задачи коммивояжера можно прочесть в [96].

В этом разделе предположим, что в рассматриваемых примерах задачи $RO2||\widetilde{C}_{\max}$ оптимальный гамильтонов цикл может быть найден за полиномиальное время. Обозначим полученную задачу $RO^*2||\widetilde{C}_{\max}$. При этом напомним, что задача с двумя машинами остается NP-трудной, даже если все работы и машины расположены на двухвершинной сети.

Пусть R^* обозначает оптимальный обход в графе G. Тогда неравенства (5.1) и (5.4) дают следующую оценку на длину расписания, получаемого алгоритмом 5.5,

$$C_{\max}(\sigma_R) \leqslant \frac{3}{2} OPT.$$

Отметим также, что построение оптимального решения на первом шаге в алгоритме 5.7 не улучшает оценку качества его работы в худшем случае.

В этом разделе представим 4/3-приближенный алгоритм для задачи $RO^*2||\widetilde{C}_{\max}$.

5.4.1 Приближенный алгоритм

Алгоритм выбирает лучшее из набора расписаний, каждое из которых определяется последовательностью выполнения работ на каждой из машин и фиксированным обходом каждой машиной вершин графа G.

Занумеруем работы согласно их появлению в оптимальном туре R^*, т.е. $R^* = (v_0, v_1, \ldots, v_n, v_0)$. Определим следующее семейство обходов, получаемых из цикла R^*:

$$R_0 = R^*,\ R_i = (v_0, v_i) \oplus R^*,\ R_{-i} = R^* \oplus (v_i, v_0) \quad (i = 1, \ldots, n),$$

$$R_{\bar{i}} = \bar{R}_i \quad (i = -n, \ldots, n),$$

где $\bar{R}_i$ — обход, обратный обходу R_i. Например, $R_{\bar{0}}$ — это обход, обратный обходу R^*.

Фактически каждый обход R однозначно определяет порядок $\pi(R)$ выполнения работ на машине, так как все вершины кроме, может быть, одной посещаются каждой машиной ровно один раз. Только вершина v_i в обходах с индексом i или $-i$ посещается дважды. В этом случае положим, что выполнение работы J_i происходит при посещении ее машиной вне обхода R (или $\bar{R}$).

Заметим, что $|R_0| = T^*$ и $|R_i| = |R_{-i}| = T^* + 2\tau_{0i}$. Предположим, что машина A посещает вершины графа G согласно обходу R_α и выполняет работы в порядке

$\pi(R_\alpha)$, а машина B посещает вершины графа G согласно обходу R_β и выполняет работы в порядке $\pi(R_\beta)$. Определим процедуру $SCHED(\alpha, \beta)$, которая по заданным перестановкам строит допустимое расписание и определяет конфликтную работу согласно следующему правилу.

Если перестановки $\pi(R_\alpha)$ и $\pi(R_\beta)$ бесконфликтные, то машины выполняют все операции без задержек. Если перестановки $\pi(R_\alpha)$ и $\pi(R_\beta)$ конфликтуют по одной работе, то, как и в алгоритме 5.5, рассмотрим два расписания, в которых конфликтная работа выполняется сначала либо на машине A, либо на машине B, и выберем среди них лучшее. Если перестановки $\pi(R_\alpha)$ и $\pi(R_\beta)$ конфликтуют более чем по одной работе, то будем считать расписание пустым.

Алгоритм 5.8 Процедура $SCHED(\alpha, \beta)$

1: Положить $\pi_A := \pi(R_\alpha)$
2: Положить $\pi_B := \pi(R_\beta)$
3: **if** перестановки π_A и π_B бесконфликтные **then**
4: Построить расписание σ, в котором машины выполняют работы согласно перестановкам π_A и π_B.
5: Положить $k := 0$.
6: **if** перестановки π_A и π_B конфликтуют по одной работе **then**
7: Найти конфликтную работу J_j.
8: Построить раннее расписание σ_1, в котором работа J_j выполняется сначала на машине A, и машины выполняют работы согласно перестановкам π_A и π_B.
9: Построить раннее расписание σ_2, в котором работа J_j выполняется сначала на машине B, и машины выполняют работы согласно перестановкам π_A и π_B.
10: Положить расписание σ равным кратчайшему из расписаний σ_1 и σ_2.
11: Положить $k := j$.
12: **if** перестановки π_A и π_B конфликтуют более чем по одной работе **then**
13: Положить $\sigma := \emptyset$ и $k := -1$.
14: Выдать $\langle \sigma, k \rangle$.

Заметим, что процедура $SCHED(\alpha, \beta)$ строит допустимое расписание только в случае, когда для заданных перестановок существует не более одной конфликтной работы. Это предположение, очевидно, выполняется, если $\alpha = \bar{\beta}$. Впоследствии это свойство будет установлено для всех случаев применения этой процедуры.

В частности, приближенный алгоритм 5.9 для задачи $RO^*2||\widetilde{C}_{\max}$ строит не более четырех расписаний с использованием процедуры $SCHED(\alpha, \beta)$, которые обозначим через σ_1, σ_2, σ_3, и σ_4. При этом в каждом из расписаний может быть не более одной конфликтной работы. Кроме того, в каждом из расписаний обход каждой машины содержит либо оптимальный тур R^*, либо обратный ему тур. Эту часть обхода назовем основным циклом. В расписании σ_i время прибытия машины $X \in \{A, B\}$ в вершину v_k в основном цикле обозначим через $r_i^X(k)$.

Алгоритм 5.9

1: Построить расписания σ_1 и σ_2:

$$\langle \sigma_1, \mu \rangle := SCHED(0, \bar{0}),\ \langle \sigma_2, \nu \rangle := SCHED(\bar{0}, 0).$$

2: **if** $\mu = 0$ или $\nu = 0$ **then**
3: Перейти на шаг 13:
4: **if** $\mu \neq \nu$ **then**
5: при необходимости переименуем машины и изменим направление оптимального обхода, так чтобы выполнялось неравенство $\mu < \nu$.
6: **else**
7: при необходимости либо переименуем машины, либо изменим направление в обходе R, либо изменим нумерацию двух расписаний, так чтобы $r_1^A(\mu) = \min\{r_1^A(\mu), r_2^A(\mu), r_1^B(\mu), r_2^B(\mu)\}$.
8: Построим расписание $\langle \sigma_3, \kappa \rangle := SCHED(\mu, \bar{\mu})$.
9: **if** $\mu = \nu$ **then**
10: построим расписание $\langle \sigma_4, \eta \rangle := SCHED(0, \bar{\mu})$
11: **if** $\mu \neq \nu$ **then**
12: $\langle \sigma_4, \eta \rangle := SCHED(\overline{-\nu}, -\nu)\big)$.
13: Выбрать лучшее из расписаний, построенных на предыдущих шагах.

5.4.2 Свойства и точность алгоритма 5.9

В этом параграфе оценим точность алгоритма 5.9.

Лемма 35 *Если в ходе работы алгоритма 5.9 $\mu = \nu \neq 0$, то*

(1) J_μ — единственная конфликтная работа (если есть) в расписании σ_4, т.е. $\eta \in \{\mu, 0\}$;

(2) $r_4^A(\mu) \leqslant \dfrac{T^* + l_{\max} - 0.5 d_\mu}{2}$.

Доказательство. Согласно работе алгоритма J_μ является конфликтной работой в σ_1 и $r_1^A(\mu) = \min\{r_1^A(\mu), r_1^B(\mu)\}$, $r_1^A(\mu) \leqslant r_1^B(\mu) \leqslant r_1^A(\mu) + a_\mu$, где a_μ — длительность операции работы J_μ на машине A. Порядок выполнения работ на машине A в расписании σ_4 такой же, как и в расписании σ_1, следовательно, $r_4^A(\mu) = r_1^A(\mu)$. Машина B в расписании σ_4 движется по тому же маршруту, как и в расписании σ_1, пока не достигнет вершины v_μ. Следовательно, $r_4^B(\mu) = r_1^B(\mu) \leqslant r_4^A(\mu) + a_\mu$. Отсюда получаем, что никакая работа J_j, $j > \mu$, не может быть конфликтной в σ_4, так как машина B завершит выполнение работы J_j до того как машина A прибудет в вершину v_j. Также никакая работа J_j, $j < \mu$, не может быть конфликтной в σ_4, так как $r_4^A(\mu) \leqslant r_4^B(\mu)$. Первое утверждение леммы доказано.

Заметим, что в расписаниях σ_1 и σ_2 обе машины обходят вершины в противоположных направлениях. Значит момент прибытия $r_1^A(\mu)$ машины A в вершину v_μ

в расписании σ_1 равен сумме длительностей работ $J_1, \ldots, J_{\mu-1}$ на машине A плюс время, потраченное на перемещение этой машины по маршруту $(v_0, v_n, v_{n-1}, \ldots, v_\mu)$. Аналогично, момент прибытия $r_2^A(\mu)$ равен сумме длительностей работ $J_{\mu+1}, \ldots, J_n$ на машине A плюс время, потраченное на перемещение этой машины по маршруту $(v_0, v_n, v_{n-1}, \ldots, v_\mu)$. В итоге получим $r_1^A(\mu) + r_2^A(\mu) = T^* + l_1 - a_\mu \leqslant T^* + l_{\max} - a_\mu$, что влечет

$$\min\{r_1^A(\mu), r_2^A(\mu)\} \leqslant \frac{T^* + l_{\max} - a_\mu}{2}. \tag{5.12}$$

Рассуждая аналогично, получим

$$\min\{r_1^B(\mu), r_2^B(\mu)\} \leqslant \frac{T^* + l_{\max} - b_\mu}{2}. \tag{5.13}$$

Комбинируя неравенства (5.12) и (5.13) с равенствами $a_\mu + b_\mu = d_\mu$ и $r_4^A(\mu) = r_1^A(\mu)$ и принимая во внимание соглашения, сделанные на шагах 5 и 7 алгоритма 5.9, получим утверждение леммы. □

Лемма 36 *Если в ходе работы алгоритма 5.9 $\mu \neq \nu$ и существуют конфликтные работы J_κ и J_η в расписаниях σ_3 и σ_4, то $\kappa \leq \mu \leq \nu \leq \eta$.*

Доказательство. Поскольку J_μ — конфликтная работа в σ_1, имеем

$$r_3^B(\mu) = r_1^B(\mu) \leqslant r_1^A(\mu) + a_\mu \leqslant r_1^A(\mu) + a_\mu + 2\tau_{0\mu} \leqslant r_3^A(\mu),$$

что влечет $\kappa \leq \mu$. Так как J_ν — конфликтная работа в σ_2, имеем

$$r_4^B(\nu) = r_2^B(\nu) \leqslant r_2^A(\nu) + a_\nu \leqslant r_2^A(\nu) + a_\nu + 2\tau_{0\nu} \leqslant r_4^A(\nu),$$

что влечет $\nu \geq \eta$. С учетом того, что согласно шагу 5 алгоритма 5.9 $\mu < \nu$, получим $\kappa \leq \mu \leq \nu \leq \eta$. □

Обозначим через $F_{\max}$ длину расписания, полученного алгоритмом 5.9. По лемме 35 каждое из расписаний, построенных алгоритмом, имеет не больше одной конфликтной работы. Если в σ_1 (σ_2) нет конфликтной работы, то σ_1 (σ_2) является оптимальным расписанием. Пусть J_μ и J_ν — конфликтные работы в σ_1 и σ_2 соответственно. Из (5.4) имеем

$$C_{\max}(\sigma_1) \leqslant T^* + \ell_{\max} + \frac{1}{2}d_\mu, \tag{5.14}$$

$$C_{\max}(\sigma_2) \leqslant T^* + \ell_{\max} + \frac{1}{2}d_\nu. \tag{5.15}$$

Предположим, что в одном из расписаний σ_3 или σ_4 нет конфликтных работ. В таком расписании обе машины работают без простоя. Для случая $\mu = \nu$ в расписании σ_4 получим

$$F_{\max} \leqslant T^* + \ell_{\max} + 2\tau_{0\mu}. \tag{5.16}$$

Складывая (5.16) и дважды (5.14), из (5.1) получим, что $3F \leqslant 4\bar{C}_R$, и $F \leqslant \frac{4}{3}OPT$. Аналогичные рассуждения влекут неравенство $F \leqslant \frac{4}{3}OPT$ и в случае $\mu \neq \nu$ в расписании σ_4.

Предположим, что во всех расписаниях, построенных алгоритмом 5.9, есть конфликтные работы. Рассмотрим два случая.

Случай 1: $\mu = \nu$. В этом случае в расписании σ_4 машина A движется по обходу R, а машина B — по обходу $\bar{R}_\mu$, при этом работа J_μ обслуживается последней, и по лемме 35 J_μ — конфликтная работа в σ_4. В одном из двух расписаний, построенном процедурой $SCHED$, машина A работает без простоев и завершает свою работу в момент $T^* + \ell_1 \leqslant \bar{F}$, а машина B по лемме 35 завершает свою работу в момент $\frac{T^*+\ell_{\max}-0.5d_\mu}{2} + d_\mu + \tau_{0\mu}$. Обозначим это расписание через σ'. Предположим без ограничения общности, что расписание σ_4 не оптимально, тогда $F_{\max}(\sigma') \leqslant \frac{T^*+\ell_{\max}-0.5d_\mu}{2} + d_\mu + \tau_{0\mu}$. Отсюда получим

$$F_{\max}(\sigma_4) \leqslant \frac{T^* + \ell_{\max}}{2} + \frac{3d_\mu}{4} + \tau_{0\mu}. \tag{5.17}$$

Пусть J_κ — конфликтная работа в расписании σ_3. Если $\kappa \neq \mu$, то из (5.4) получим

$$F_{\max}(\sigma_3) \leqslant T^* + \ell_{\max} + 2\tau_{0\mu} + \frac{1}{2}d_\kappa.$$

В случае $\kappa = \mu$ среди двух расписаний, построенных при выборе σ_3, рассмотрим расписание σ'', в котором машина A работает без простоев. Для этого расписания из (5.1) получим, что

$$F_{\max}(\sigma'') \leqslant \max\{T^* + \ell_{\max} + 2\tau_{0\mu}, d_\mu + 2\tau_{0\mu}\} \leqslant \max\{T^* + \ell_{\max} + 2\tau_{0\mu}, OPT\},$$

и, предполагая, что расписание σ_3 неоптимальное, имеем оценку

$$F_{\max}(\sigma_3) \leqslant T^* + \ell_{\max} + 2\tau_{0\mu} + \frac{d_{\kappa(\neq\mu)}}{2}. \tag{5.18}$$

Сложив (5.14), (5.18) и четырежды (5.17), получим

$$6F_{\max} \leqslant 4(T^* + \ell_{\max}) + 3(d_\mu + 2\tau_{0\mu}) + \frac{d_\mu + d_{\kappa(\neq\mu)}}{2}.$$

Используя (5.1) и учитывая, что последнее слагаемое не превосходит $\ell_{\max}$, получим $F_{\max} \leqslant \frac{4}{3}OPT$.

Случай 2: $\mu \neq \nu$. По лемме 36 имеем $\kappa \leq \mu < \nu \leq \eta$. Если σ_3 не оптимально, то выполнено неравенство (5.18) и неравенство

$$F_{\max}(\sigma_4) \leqslant T^* + \ell_{\max} + 2\tau_{0\nu} + \frac{d_{\eta(\neq\nu)}}{2}. \tag{5.19}$$

Складывая (5.18), (5.19), дважды (5.14) и дважды (5.15), и учитывая, что $\tau_{0j} \leqslant \frac{1}{2}T^*$, получим

$$6F_{\max} \leqslant 6(T^* + \ell_{\max}) + (d_\mu + 2\tau_{0\mu}) + (d_\nu + 2\tau_{0\nu}) + \frac{d_{\kappa(\neq\mu)} + d_{\eta(\neq\nu)}}{2} \leqslant$$
$$6(T^* + \ell_{\max}) + \left(T^* + \frac{d_\mu + d_\nu}{2}\right) + T^* + \frac{d_\mu + d_\nu + d_{\kappa(\neq\mu)} + d_{\eta(\neq\nu)}}{2}.$$

Так как последнее слагаемое не превосходит $\ell_{\max}$, неравенство (5.1) влечет $F_{\max} \leqslant \frac{4}{3}OPT$.

Заметим, что время работы процедуры $SCHED(\alpha, \beta)$ и алгоритма 5.9 линейно зависит от числа работ. Окончательно получаем следующий результат.

Теорема 29 *Предположим, что для любого примера метрической задачи коммивояжера на графе G известен оптимальный тур. Тогда алгоритм 5.9 строит $\frac{4}{3}$-приближенное решение примера задачи маршрутизации в цеховой задаче открытого типа на двух машинах за $O(n)$ элементарных операций, где n — число работ.*

5.5 Две машины, две вершины

В этом разделе рассмотрим задачу $RO2||V| = 2|\widetilde{C}_{\max}$, в которой транспортная сеть состоит всего из двух вершин v_0 и v_1. Множество работ $\mathcal{J}$ должно быть обслужено двумя машинами A и B. Множество работ $\mathcal{J}$ разбито на два подмножества N_0 и N_1. Все работы из N_0 лежат в вершине v_0 и все работы из N_1 лежат в вершине v_1. Для обслуживания каждой работы машина должна переместиться в вершину, где находится эта работа. Каждой машине требуется τ единиц времени для перемещения из одной вершины в другую. В начальный момент времени обе машины находятся в v_0 и должны вернуться туда после выполнения всех работ. Таким образом, каждая машина должна сделать четное число перемещений. Каждая работа имеет ровно две операции, одна из которых должна быть выполнена на машине A, а другая – на машине B. Для каждой работы $J_j \in \mathcal{J}$ обозначим длительности ее обслуживания машинами A и B через a_j и b_j соответственно. Никакие две операции, выполняемые на одной машине или принадлежащие одной работе, не могут выполняться одновременно. Прерывания во время выполнения операции запрещены. Требуется составить расписание выполнения работ и перемещений машин, в котором машины выполнят все работы и вернутся в исходную вершину v_0 за минимальное время.

5.5.1 Основные обозначения и предварительные результаты

Решение задачи $RO2||V| = 2|\widetilde{C}_{\max}$ будем представлять ориентированным графом $G = (V, E)$, в котором множество вершин V содержит по одной вершине для каждой операции и две специальные вершины — источник 0 и сток C_{max}. Каждой вершине приписан неотрицательный вес, равный длительности соответствующей операции. Веса стока и источника положим равными нулю. Для двух операций x и y дуга $(x, y) \in E$ указывает на отношение предшествования между этими операциями, т.е.

операция x должна завершиться до начала операции y. Так как никакие две операции, выполняемые на одной машине или принадлежащие одной работе, не могут выполняться одновременно, то в G должны существовать дуги между любыми двумя операциями, выполняемыми на одной машине, и любыми двумя операциями одной работы. Кроме того, заданы дуги из источника в каждую вершину и из каждой вершины в сток.

Поскольку в задаче $RO2||V| = 2|\widetilde{C}_{\max}$ машинам потребуется переезд из одной вершины в другую, будем дополнительно предполагать, что каждая дуга имеет неотрицательный вес δ. Запись $x \xrightarrow{\delta} y$ означает, что операция y начинает выполняться не раньше, чем через время δ после завершения операции x. Запись $x \to y$ означает, что вес дуги (x, y) равен 0. Длина пути в данном графе определяется как сумма весов вершин и дуг, принадлежащих этому пути.

Через $s(x, \sigma)$ и $C(x, \sigma)$ обозначим моменты начала и завершения обслуживания операции x в расписании σ. Построим по заданному графу G расписание σ по следующему правилу. Для каждой операции x положим $s(x, \sigma)$ равным длине максимального пути из 0 в x. Расписание, построенное по такому правилу, называется *активным*, и каждый такой граф определяет единственное активное расписание [23]. Длина активного расписания равна длине самого длинного в графе пути из источника в сток, который будем называть *критическим*. Дуга $(x, y) \in E$ называется *транзитивной*, если в G существует путь из x в y, проходящий через другую вершину. Легко показать, что в G всегда существует критический путь, не содержащий транзитивных дуг нулевого веса.

При построении расписаний будем использовать понятие "схемы расписания", которое упрощает представление расписания в виде ориентированного графа. В такой схеме каждая вершина x, кроме источника и стока, может быть либо операцией, либо подмножеством операций. При этом предполагается, что все операции в подмножестве, соответствующем вершине, выполняются без задержек и вес вершины равен сумме длин операций. Назовем такое подмножество операций *блоком*. Дополнительно потребуем, чтобы критический путь либо не проходил через вершину, соответствующую блоку, либо содержал его целиком.

Пусть $l_1 = \sum_{j \in N} a_j$ и $l_2 = \sum_{j \in N} b_j$ — загрузка первой и второй машины соответственно, и пусть $l_{max} = \max\{l_1, l_2\}$. Величина $d_j = a_j + b_j$ называется длиной работы j. Пусть $d^0_{max} = \max_{j \in N_0} d_j$ и $d^1_{max} = \max_{j \in N_1} d_j$ — длины максимальной работы в вершине v_0 и v_1 соответственно. Положим $d_{max} = \max\{d^0_{max}, d^1_{max}\}$.

Если все работы сосредоточены в одной вершине, то задача $RO2||V| = 2|\widetilde{C}_{\max}$ эквивалентна классической цеховой задаче открытого типа на двух машинах $O2||C_{max}$. Напомним, что для задачи $O2||C_{max}$ справедлива следующая оценка на длину оптимального расписания:

$$C^*_{max}(O2||C_{max}) \geq \lambda_O \doteq \max\{l_{max}, d_{max}\}.$$

Для задачи $RO2||V| = 2|\widetilde{C}_{\max}$ аналогичная оценка выписывается следующим образом [40]:

$$C^*_{max}(RO2||V| = 2|\widetilde{C}_{\max}) \geq \lambda_R \doteq \max\{l_{max} + 2\tau, d^0_{max}, d^1_{max} + 2\tau\}.$$

Для задачи $O2||C_{max}$ Гонзалез и Сани [97] предложили простой полиномиальный алгоритм, строящий расписание длины λ_O. Поскольку λ_O является нижней оценкой длины любого допустимого расписания, алгоритм Гонзалеза-Сани находит оптимальное расписание. Операции работы J_j на машинах A и B будем обозначать через A_j и B_j соответственно. Работу J_0, на которой достигается значение $\max\{\max\{a_i|J_i \in N_{\leq}\}, \max\{b_i|J_i \in N_{>}\}\}$, назовем *диагональной*. Опишем алгоритм Гонзалеза-Сани в следующем виде.

Алгоритм 5.10 Алгоритм Гонзалеза-Сани

1: Разбить множество работ $\mathcal{J}$ на два подмножества: $N_{\leq} = \{J_j \in \mathcal{J}|a_j \leq b_j\}$ и $N_{>} = \{J_j \in \mathcal{J}|a_j > b_j\}$.
2: Найти диагональную работу J_0.
3: **if** $J_0 \notin N_{\leq}$ **then**
4: переименовать машины и переопределить подмножества $N_{\leq}$ и $N_{>}$.
5: Положить $d_0 := a_0 + b_0$.
6: Положить $N'_{\leq} \doteq N_{\leq} \setminus \{J_0\}$.
7: **if** $d_0 = \lambda_O$ **then**
8: занумеровать оставшиеся работы на каждой машине в произвольном порядке,
9: **else**
10: занумеровать оставшиеся работы так, что первыми идут все работы из $N'_{\leq}$, а затем все работы из $N_{>}$.
11: Определить следующий порядок выполнения операций на машине A:
$$A_1 \to A_2 \to \cdots \to A_{n-1} \to A_0.$$
12: Определить следующий порядок выполнения операций на машине B:
$$B_0 \to B_1 \to B_2 \to \cdots \to B_{n-1}.$$
13: Определить следующий порядок выполнения работы J_0: диагональная работа сначала выполняется на машине B, а потом на машине A.
14: Определить следующий порядок выполнения остальных работ: все остальные работы сначала выполняются на машине A, а потом на машине B.
15: Построить активное расписание σ согласно порядку выполнения операций, определенному выше.
16: Выдать расписание σ.

Лемма 37 [97] *Алгоритм Гонзалеза-Сани строит расписание σ с $C_{max}(\sigma) = \lambda_O$ за $O(n)$ элементарных операций.*

Заметим, что на шаге 8 алгоритма Гонзалеза-Сани можно выбрать произвольный порядок работ из множества $\mathcal{J} \setminus \{J_0\}$ на каждой из машин, а на шаге 10 можно выбрать любой порядок работ из множества $N'_{\leq}$ и любой из множества $N_{>}$. В дальнейшем мы используем это свойство для построения оптимальных расписаний в задаче $RO2||\widetilde{C}_{\max}$, выбирая подходящий порядок работ в зависимости от их расположения в сети.

5.5.2 Достаточные условия для полиномиальной разрешимости задачи $RO2||V| = 2|\widetilde{C}_{\max}$

В этом разделе установим достаточные условия, когда оптимальное решение задачи $RO2||V| = 2|\widetilde{C}_{\max}$ может быть найдено за линейное от числа работ время. Как и в алгоритме Гонзалеза-Сани, обозначим диагональную работу через J_0. Для любого множества работ X через $\{X\}_A$ ($\{X\}_B$) будем обозначать множество его операций на машине A (B).

Случай 1. Пусть $J_0 \in N_0$. Рассмотрим следующее расписание σ_1. Пусть машина A сначала выполняет все работы из $N'_{\leq} \cap N_0$, затем — все работы из $N_{\leq} \cap N_1$, потом — все работы из $N_{>} \cap N_1$, после этого — все работы из $N_{>} \cap N_0$ и последней — работу J_0. Машина B первой выполняет работу J_0, затем — оставшиеся работы в том же порядке, как и машина A. Схема $G(\sigma_1)$ этого расписания представлена на рисунке 5.1.

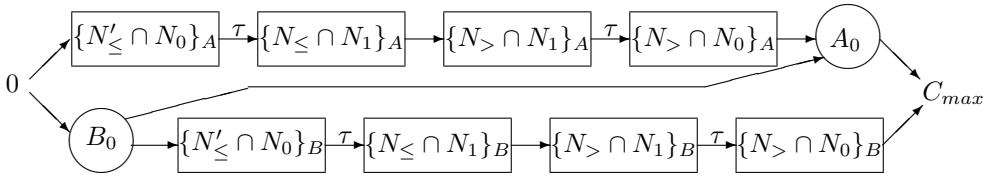

Рис. 5.1: Схема $G(\sigma_1)$, $J_0 \in N_0$.

Покажем, что один из следующих путей является критическим в $G(\sigma_1)$:

(1) $0 \to \{N'_{\leq} \cap N_0\}_A \xrightarrow{\tau} \{N_{\leq} \cap N_1\}_A \to \{N_{>} \cap N_1\}_A \xrightarrow{\tau} \{N_{>} \cap N_0\}_A \to A_0 \to C_{max}$,

(2) $0 \to B_0 \to \{N'_{\leq} \cap N_0\}_B \xrightarrow{\tau} \{N_{\leq} \cap N_1\}_B \to \{N_{>} \cap N_1\}_B \xrightarrow{\tau} \{N_{>} \cap N_0\}_B \to C_{max}$,

(3) $0 \to B_0 \to A_0 \to C_{max}$.

Заметим, что длина первого пути равна $l_1 + 2\tau$, длина второго пути — $l_2 + 2\tau$, и длина третьего пути — d^0_{max}. Кроме указанных путей только пути вида $0 \to A_1 \to \cdots \to A_i \to B_i \to \cdots \to B_{n-1} \to C_{max}$ не содержат транзитивных дуг нулевого веса. Покажем, что ни один из этих путей не может быть критическим.

Пусть $J_i \in N'_{\leq}$. Учитывая, что $a_j \leq b_j$ для всех $j \in N_{\leq}$ и $a_i \leq a_0 \leq b_0$, получаем

$$\sum_{j=1}^{i} a_j + \sum_{j=i}^{n-1} b_j + 2\tau \leq \sum_{j=1}^{n-1} b_j + a_i + 2\tau \leq \sum_{j=1}^{n-1} b_j + b_0 + 2\tau = l_2 + 2\tau.$$

Пусть $J_i \in N_{>}$. Учитывая, что $a_j \geq b_j$ для всех $j \in N_{>}$ и $b_i \leq a_0$, получаем

$$\sum_{j=1}^{i} a_j + \sum_{j=i}^{n-1} b_j + 2\tau \leq \sum_{j=1}^{n-1} a_j + b_i + 2\tau \leq \sum_{j=1}^{n-1} a_j + a_0 + 2\tau = l_1 + 2\tau.$$

Следовательно, $C_{max}(\sigma_1) = \lambda_R$ и σ_1 – оптимальное расписание.

Случай 2. Пусть $J_0 \in N_1$ и $d_0 \geq l_{max}$. Рассмотрим следующее расписание σ_2. Пусть машина A сначала выполняет все работы из N_0, затем — все работы из $N_1 \backslash \{J_0\}$ и последней — работу J_0, а машина B первой выполняет работу J_0, затем — все работы из $N_1 \backslash \{J_0\}$ и последними — работы из N_0. Схема $G(\sigma_2)$ этого расписания представлена на рисунке 5.2.

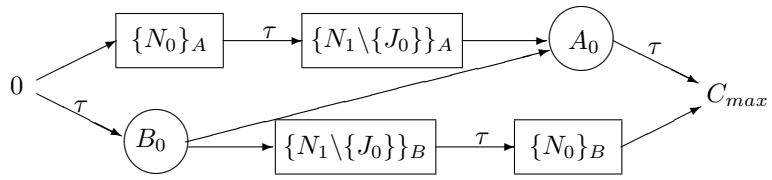

Рис. 5.2: $G(\sigma_2)$, $J_0 \in N_1$ и $d_0 \geq l_{max}$.

Заметим, что $d_0 \geq l_{max}$ влечет $\sum_{i=1}^{n-1} a_i \leq b_0$ и $\sum_{i=1}^{n-1} b_i \leq a_0$. Отсюда следует, что путь $0 \xrightarrow{\tau} B_0 \rightarrow A_0 \xrightarrow{\tau} C_{max}$ является критическим в $G(\sigma_2)$. Длина этого пути равна $d^1_{max} + 2\tau$. Следовательно, $C_{max}(\sigma) = \lambda_R$ и σ_2 — оптимальное расписание.

Так как расписания σ_1 и σ_2 могут быть построены за линейное от числа работ время, то получаем следующее утверждение.

Теорема 30 *Если в примере I задачи* $RO2||V| = 2|\widetilde{C}_{\max}$ *выполнено одно из двух следующих условий:*

a) $J_0 \in N_0$,

b) $J_0 \in N_1$ *и* $d_0 \geq l_{max}$,

то оптимальное расписание имеет длину λ_R *и может быть построено за время* $O(n)$.

5.5.3 Точный алгоритм и приближенная схема

Теорема 30 определяет конструктивные и легко проверяемые условия, которые гарантируют полиномиальную разрешимость задачи $RO2||V| = 2|\widetilde{C}_{\max}$. Таким образом, только в случае, когда диагональная работа лежит в удаленной вершине, и ее длина меньше l_{max}, нам не удается построить для ее решения точный полиномиальный алгоритм. Именно для этого случая в [41] доказана NP-трудность задачи $RO2||V| = 2|\widetilde{C}_{\max}$.

5.5.4 Конфигурации оптимальных расписаний в трудных примерах

В оставшейся части статьи будем предполагать, что $J_0 \in N_1$ и $d_0 < l_{max}$. Следующая теорема устанавливает верхнюю оценку на длину оптимального расписания.

Теорема 31 *Пусть* $J_0 \in N_1$ *и* $d_0 < l_{max}$. *Тогда* $C^*_{max}(RO2||V| = 2|\widetilde{C}_{\max}) \leq l_{max} + 4\tau$.

Доказательство. Пусть σ_3 – расписание, в котором машины выполняют работы в том же порядке, что и в расписании σ_1. Схема $G(\sigma_3)$ этого расписания представлена на рисунке 5.3.

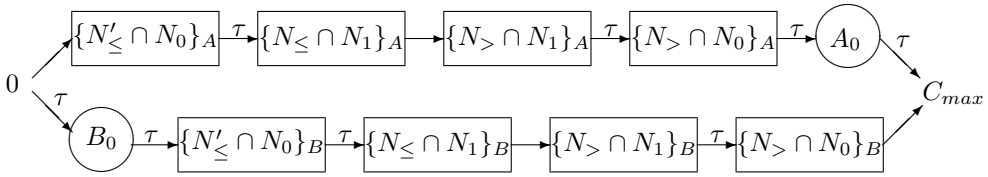

Рис. 5.3: Схема $G(\sigma_3)$, $J_0 \in N_1$ и $d_0 < l_{max}$.

Повторяя рассуждения аналогичные рассуждениям для случая 1 предыдущего параграфа, получаем что один из двух следующих путей

$$0 \to \{N'_{\leq} \cap N_0\}_A \xrightarrow{\tau} \{N_{\leq} \cap N_1\}_A \to \{N_{>} \cap N_1\}_A \xrightarrow{\tau} \{N_{>} \cap N_0\}_A \xrightarrow{\tau} A_0 \xrightarrow{\tau} C_{max},$$

$$0 \xrightarrow{\tau} B_0 \xrightarrow{\tau} \{N'_{\leq} \cap N_0\}_B \xrightarrow{\tau} \{N_{\leq} \cap N_1\}_B \to \{N_{>} \cap N_1\}_B \xrightarrow{\tau} \{N_{>} \cap N_0\}_B \to C_{max}$$

является критическим. Следовательно, $C^*_{max}(RO2||V| = 2|\widetilde{C}_{\max}) \leq l_{max} + 4\tau$. □

Обозначим через $\psi(M, \sigma)$ количество перемещений из одной вершины в другую машины M в расписании σ. Далее без ограничения общности будем считать, что $l_1 \geq l_2$. Тогда непосредственно из теоремы 31 вытекает следующий результат.

Следствие 5 *Если существует оптимальное расписание σ^* с $\psi(A, \sigma^*) \geq 4$, то σ_3 также является оптимальным расписанием.*

Пусть σ — некоторое расписание, в котором машина A совершает ровно одну поездку в вершину V_1, т.е. машина A выполняет какие-то работы из N_0, затем перемещается в вершину V_1, выполняет все работы из множества N_1, возвращается в V_0 и выполняет оставшиеся работы, тем самым $\psi(A, \sigma) = 2$.

Покажем, что для любого допустимого расписания σ с $\psi(A, \sigma) = 2$ множество работ N можно так разбить на конечный (не зависящий от n) набор подмножеств $\mathcal{N}$, что существует расписание σ^*, удовлетворяющее следующим условиям:

(C1) $C_{max}(\sigma^*) \leq C_{max}(\sigma)$;

(C2) $\psi(A, \sigma^*) = 2$;

(C3) для каждого подмножества X из $\mathcal{N}$ операции работ из X на каждой из машин образуют блок в σ^*.

Для доказательства этого утверждения построим цепочку преобразований расписания σ в σ^*. Будем говорить, что *операция сдвинута влево (вправо)* относительно ее позиции в расписании σ, если в новом расписании она начинает выполняться не

позднее (не раньше) момента начала ее выполнения в σ. Блоки операций одного подмножества работ будем называть *парными*.

Пусть $t_1 = \min\{s(A_i, \sigma) | J_i \in N_1\}$, $t_2 = \max\{C(A_i, \sigma) | J_i \in N_1\}$. Тогда все работы из множества N_1 выполняются на первой машине на отрезке $[t_1, t_2]$. Обозначим через N_0' множество работ из N_0, выполняющихся на машине A на отрезке $[0, t_1]$, и через N_0'' множество работ из N_0, выполняющихся на машине A на отрезке $[t_2, C_{max}]$. Для введенных обозначений порядок выполнения работ на машине A в σ может быть записан схемой $0 \to \{N_0'\}_A \xrightarrow{\tau} \{N_1\} \xrightarrow{\tau} \{N_0''\}_A \to C_{max}$.

Пусть L – множество работ с первой операцией на машине A, и R – множество работ с первой операцией на машине B в σ. Тогда следующие преобразования допустимы относительно операций работ этих множеств:

- сдвиг влево операции работы из L на машине A или сдвиг вправо операции работы из L на машине B;
- сдвиг вправо операции работы из R на машине A или сдвиг влево операции работы из R на машине B.

Заметим, что множества L и R не будут изменяться в последующих преобразованиях.

С учетом введенных обозначений можно переупорядочить работы на A. Рассмотрим расписание $\bar{\sigma}$, в котором на машине A операции выполняются в следующем порядке:

$$0 \to \{N_0' \cap L\}_A \to \{N_0' \cap R\}_A \xrightarrow{\tau} \{N_1 \cap L\}_A \to$$
$$\{N_1 \cap R\}_A \xrightarrow{\tau} \{N_0'' \cap L\}_A \to \{N_0'' \cap R\}_A \to C_{max},$$

и при этом внутри каждого из блоков операции идут в том же порядке, что и в σ. Нетрудно видеть, что переупорядочение работ эквивалентно сдвигу влево работ из L и вправо — работ из R, и в $\bar{\sigma}$ каждое из множеств операций $\{N_0'\}_A$, $\{N_1\}_A$, $\{N_0''\}_A$ может быть выполнено блоком внутри того же интервала времени, в котором оно выполняется в σ.

Дополнительно разобьем каждое из множеств $N_0' \cap L$ и $N_0'' \cap R$ на два подмножества. Пусть $t_3 = 0$, если $N_1 \cap R = \emptyset$, и $t_3 = \min\{s(B_i, \sigma) | J_i \in N_1 \cap R\}$ в противном случае; $t_4 = C_{max}$, если $N_1 \cap L = \emptyset$, и $t_4 = \max\{C(B_i, \sigma) | J_i \in N_1 \cap L\}$ в противном случае.

Положим $\bar{N}_{0L}' = \{J_i \in N_0' \cap L | s(B_i, \bar{\sigma}) \geq t_4\}$ и $N_{0L}' = N_0' \cap L \setminus \bar{N}_{0L}'$. Без ограничения общности можно считать, что все работы из N_{0L}' предшествуют работам из $\bar{N}_{0L}'$ на обеих машинах. Действительно, по определению все работы из N_{0L}' заканчиваются на машине B до момента t_4. Так как первая операция этих работ выполняется на машине A, то и на машине A они заканчиваются до момента t_4. Пусть некоторая работа $J_i \in \bar{N}_{0L}'$ выполняется на машине A перед некоторой работой $J_k \in N_{0L}'$. Поставим ее на машине A после последней работы из N_{0L}' и сдвинем все работы с J_k по J_i влево к моменту $s(A_i, \bar{\sigma})$. Полученное расписание является допустимым, так как операция A_i, которая сдвигается вправо, завершится до момента t_4, а все остальные работы сдвигались влево, что не создает конфликтов со вторыми операциями тех же работ.

i	p_i	D_i
$i : J_i \in R \setminus \bar{N}''_{0R}$	$p_i = b_i$	$D_i = s(A_i, \sigma_2)$
$i : J_i \in \bar{N}''_{0R}$	$p_i = b_i$	$D_i = t_3$
$k+1$	$p_{k+1} = \tau$	$D_{k+1} = \min\{s(A_i, \sigma_2) \mid J_i \in N_{1R}\}$
$k+2$	$p_{k+2} = \tau$	$D_{k+2} = C_{max}$, если $N''_{0R} = \emptyset$ или $D_{k+2} = \min\{s(A_i, \sigma_2) \mid J_i \in N''_{0R}\}$

Таблица 5.1: Исходные данные примера I

Положим $\bar{N}''_{0R} = \{J_i \in N''_0 \cap R | C(B_i, \bar{\sigma}) \leq t_3\}$ и $N''_{0R} = N''_0 \cap R \setminus \bar{N}''_0$. Аналогично предыдущему случаю можно считать, что работы из N''_{0R} выполняются после всех работ из $\bar{N}''_{0R}$. Пусть $N'_{0R} = N'_0 \cap R$, $N_{1L} = N_1 \cap L$, $N_{1R} = N_1 \cap R$, $N''_{0L} = N''_0 \cap L$. Тогда в расписании $\bar{\sigma}$ на A операции выполняются в следующем порядке π_0 :

$$0 \to \{N'_{0L}\}_A \to \{\bar{N}'_{0L}\}_A \to \{N'_{0R}\}_A \xrightarrow{\tau} \{N_{1L}\}_A \to$$

$$\{N_{1R}\}_A \xrightarrow{\tau} \{N''_{0L}\}_A \to \{\bar{N}''_{0R}\}_A \to \{N''_{0R}\}_A \to C_{max}.$$

Положим $\mathcal{N}_1 = \{N'_{0R}, \bar{N}''_{0R}, N_{1R}, N''_{0R}, N'_{0L}, N_{1L}, \bar{N}'_{0L}, N''_{0L}\}$. В расписании $\bar{\sigma}$ для каждого подмножества $X \in \mathcal{N}_1$ операции работ из X образуют блок на машине A и $C_{max}(\bar{\sigma}) \leq C_{max}(\sigma)$. Заметим, что некоторые из подмножеств в наборе $\mathcal{N}_1$ могут быть пустыми.

Далее покажем, что операции на машине B можно переупорядочить так, что в полученном расписании будут выполнены условия (C1) и (C3). Переупорядочение работ на машине B зависит от количества поездок машины B и от распределения по подмножествам N_{1R} и N_{1L} работ из N_1 в расписании $\bar{\sigma}$. Поэтому далее мы рассмотрим три случая.

Случай 1: $\psi(B, \bar{\sigma}) \geq 4$, $N_{1L} \neq \emptyset$ и $N_{1R} \neq \emptyset$.
Преобразуем расписание $\bar{\sigma}$ в расписание σ^* такое, что для него выполнены условия (C1) – (C3) и

(C4) $\psi(B, \sigma^*) \leq 4$.

Рассмотрим следующую вспомогательную задачу I построения допустимого расписания работ на одной машине с заданными директивными сроками. Пусть $|R| = k$. Пример I содержит $k+2$ работы, занумерованные от 1 до $k+2$. Длительности работ p_i и директивные сроки D_i, $i = 1, \ldots k$, заданы в таблице 5.1.

Первые k работ соответствуют k операциям из $\{R\}_B$. Две последние работы соответствуют интервалам переезда машины B из одной вершины в другую. Покажем, что для данного примера существует допустимое расписание σ_R.

Пусть первые k работ в σ_R выполняются в те же интервалы времени, что и соответствующие им операции на машине B в расписании $\bar{\sigma}$. Если $J_i \in \bar{N}''_{0R}$, то по определению множества $\bar{N}''_{0R}$ работа J_i заканчивает выполнение до момента t_3 и $C(i, \sigma_R) \leq t_3 = D_i$. Если $J_i \in R \setminus \bar{N}''_{0R}$, то она заканчивает выполнение до того как она начнет выполняться на A. Имеем $C(i, \sigma_R) \leq s(A_i, \bar{\sigma}) = D_i$.

Пусть в σ_R работа $k+1$ выполняется в интервале времени длины τ, соответствующем первому переезду машины B из V_0 в V_1. Обозначим время прибытия B в V_1 через t. Покажем, что $t \le D_{k+1}$.

Действительно, так как $N_{1R} \neq \emptyset$, то до момента $\min\{s(A_i,\bar\sigma)|J_i \in N_{1R}\}$ на B должна быть закончена по крайней мере одна операция работы из N_{1R}. Следовательно, $t \le \min\{s(A_i,\bar\sigma)|J_i \in N_{1R}\} = D_{k+1}$.

Пусть в σ_R работа $k+2$ выполняется в интервале времени длины τ, соответствующем первому переезду машины B из V_1 в V_0. Обозначим время прибытия машины B в V_2 через t'. Очевидно, что $t' \le C_{max}$. Пусть $N''_{0R} \neq \emptyset$. Тогда до момента $\min\{s(A_i,\bar\sigma)|J_i \in N''_{0R}\}$ на B должна быть закончена по крайней мере одна операция работы из N''_{0R}. Следовательно, $t' \le \min\{s(A_i,\bar\sigma)|J_i \in N''_{0R}\} = D_{k+2}$, и расписание σ_R является допустимым относительно заданных директивных сроков.

Упорядочим в примере I все работы по неубыванию директивных сроков. Такая перестановка работ также определяет расписание, допустимое относительно директивных сроков. В исходной задаче такому расписанию соответствует следующий порядок выполнения операций на машине B:

$$0 \to \{N'_{0R}\}_B \to \{\bar N''_{0R}\}_B \xrightarrow{\tau} \{N_{1R}\}_B \xrightarrow{\tau} \{N''_{0R}\}_B.$$

Согласно этому порядку множество операций $\{R\}_B$ может быть выполнено без простоев на машине B в интервале $[0, \sum_{J_i\in R} b_i + 2\tau]$.

Рассмотрев течение времени в $\bar\sigma$ в обратном направлении от C_{max} к 0 и проведя аналогичные рассуждения для множества L, получим, что операции множества $\{L\}_B$ можно выполнить на машине B в интервале $[C_{max} - \sum_{J_i\in L} b_i - 2\tau, C_{max}]$ без конфликта с операциями этих работ на A в следующем порядке:

$$\{N'_{0L}\}_B \xrightarrow{\tau} \{N_{1L}\}_B \xrightarrow{\tau} \{N'_{0L}\}_B \to \{N''_{0L}\}_B \to C_{max}.$$

Так как $C_{max}(\bar\sigma) \ge l_2 + 4\tau \ge \sum_{J_i\in N} b_i + 4\tau$, совместная длина интервалов $[0, \sum_{J_i\in R} b_i + 2\tau]$ и $[C_{max} - \sum_{J_i\in L} b_i - 2\tau, C_{max}]$ не превосходит C_{max}.

Суммируя вышесказанное, получаем расписание σ_1^*, в котором на машине A операции выполняются в порядке π_0, а на машине B — в порядке π_1 :

$$0 \to \{N'_{0R}\}_B \to \{\bar N''_{0R}\}_B \xrightarrow{\tau} \{N_{1R}\}_B \xrightarrow{\tau} \{N''_{0R}\}_B \to$$
$$\{N'_{0L}\}_B \xrightarrow{\tau} \{N_{1L}\}_B \xrightarrow{\tau} \{\bar N'_{0L}\}_B \to \{N''_{0L}\}_B \to C_{max}.$$

Лемма 38 *Пусть σ – допустимое расписание, для которого выполнены следующие условия $\psi(A,\sigma) = 2$, $\psi(B,\sigma) \ge 4$, $N_{1L} \neq \emptyset$, и $N_{1R} \neq \emptyset$. Тогда существует расписание σ^*, в котором A обслуживает работы в порядке π_0, B обслуживает работы в порядке π_1 и выполнены условия (C1) – (C4).*

Отметим, что разбиение работ на множества L и R задает порядок выполнения операций для каждой работы, а перестановки π_0 и π_1 задают порядок между блоками операций одной машины. Будем говорить, что набор подмножеств $\mathcal{N}_1$ и перестановки π_0 и π_1 определяют конфигурацию K_1, допустимое расписание σ лежит в K_1 и писать $\sigma \in K_1$, если σ удовлетворяет следующим условиям:

1. множество работ $\mathcal{J}$ может быть разбито на набор $\mathcal{N}_1$ попарно непересекающихся подмножеств, покрывающих все множество;

2. все операции работ одного подмножества образуют блок на каждой машине в σ;

3. на машине A блоки выполняются в порядке π_0, и на машине B блоки выполняются в порядке π_1.

Как и расписания, конфигурации удобно представлять в виде ориентированного графа, где каждая вершина является блоком. Два парных блока связаны дугой, указывающей, операции какого блока должны быть выполнены первыми. Вес дуги — переменная, значение которой зависит от конкретного расписания.

Имеем $\sigma_1^* \in K_1$. Схема конфигурации K_1 представлена на рисунке 5.4.

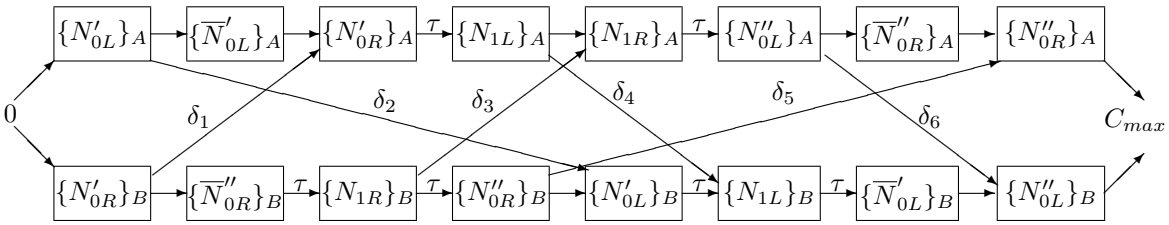

Рис. 5.4: Конфигурация K_1

Отметим, что в конфигурации K_1 представлено восемь парных блоков. Величины δ_i, $i = 1, \ldots, 6$, указывают на минимальное время задержки между завершением выполнения первого блока и началом выполнения второго блока. Дуги между парными блоками подмножеств $\bar{N}'_{0L}$ и $\bar{N}''_{0R}$ опущены, так как интервалы, в которых они выполняются, не пересекаются. Нетрудно понять, что $\delta_i \leq 0$ и их конкретное значение зависит от перестановки операций в парных блоках. Вопрос определения оптимальной перестановки операций в парных блоках будет рассмотрен в следующем разделе.

Случай 2: $\psi(B, \bar{\sigma}) \geq 2$, $N_{1L} \neq \emptyset$, и $N_{1R} = \emptyset$.
Последнее равенство влечет $t_3 = 0$ и $\bar{N}''_{0R} = \emptyset$. Следовательно, перестановку π_0 выполнения работ на машине A можно переписать как π_2 :

$$0 \to \{N'_{0L}\}_A \to \{\bar{N}'_{0L}\}_A \to \{N'_{0R}\}_A \xrightarrow{\tau} \{N_{1L}\}_A \xrightarrow{\tau} \{N''_{0L}\}_A \to \{N''_{0R}\}_A \to C_{max}.$$

Пусть $|R| = k$. Рассмотрим вспомогательную задачу I построения допустимого расписания k работ на одной машине с длительностями $p_i = b_i$ и заданными директивными сроками $D_i = s(A_i, \bar{\sigma})$.

Упорядочим в примере I все работы по неубыванию директивных сроков. Такая перестановка работ определяет расписание, допустимое относительно директивных сроков. В исходной задаче этому расписанию соответствует следующий порядок выполнения блоков операций из $\{R\}_B$ на машине B :

$$0 \to \{N'_{0R}\}_B \to \{N''_{0R}\}_B.$$

Согласно этому порядку множество операций $\{R\}_B$ может быть выполнено без простоев на машине B в интервале $[0, \sum_{J_i \in R} b_i]$.

Из анализа для случая 1 мы знаем, что операции множества $\{L\}_B$ можно выполнить на машине B в интервале $[C_{max} - \sum_{J_i \in L} b_i - 2\tau, C_{max}]$ без конфликта с операциями этих работ на A в следующем порядке:

$$\{N'_{0L}\}_B \xrightarrow{\tau} \{N_{1L}\}_B \xrightarrow{\tau} \{\bar{N}'_{0L}\}_B \to \{N''_{0L}\}_B \to C_{max}.$$

В итоге получаем расписание σ_2^*, в котором на машине A операции выполняются в порядке π_2 и на машине B — в порядке π_3 :

$$0 \to \{N'_{0R}\}_B \to \{N''_{0R}\}_B \to \{N'_{0L}\}_B \xrightarrow{\tau} \{N_{1L}\}_B \xrightarrow{\tau} \{\bar{N}'_{0L}\}_B \to \{N''_{0L}\}_B \to C_{max}.$$

Лемма 39 *Пусть σ – допустимое расписание , для которого выполнены следующие условия: $\psi(A, \sigma) = 2$, $\psi(B, \sigma) \geq 2$, $N_{1L} \neq \emptyset$, и $N_{1R} = \emptyset$. Тогда существует расписание σ^*, в котором A обслуживает работы в порядке π_2, B обслуживает работы в порядке π_3 и выполнены условия (C1)–(C3) и*

(C5) $\psi(B, \sigma^*) \leq 2$.

Набор подмножеств $\mathcal{N}_2 = \{N'_{0R}, N''_{0R}, N'_{0L}, N_{1L}, \bar{N}'_{0L}, N''_{0L}\}$ и перестановки π_2 и π_3 определяют конфигурацию K_2. Конфигурация K_2, содержащая расписание σ_2^*, представлена на рисунке 5.5.

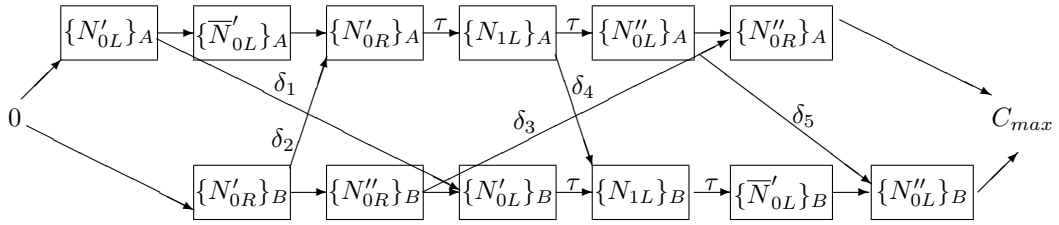

Рис. 5.5: Конфигурация K_2

Случай, когда $N_{1L} = \emptyset$ и $N_{1R} \neq \emptyset$, симметричен рассмотренному и может быть сведен к нему заменой порядка выполнения операций на каждой из машин на обратный в исходном расписании σ.

Случай 3: $\psi(A, \bar{\sigma}) = 2$, $\psi(B, \bar{\sigma}) = 2$, $N_{1L} \neq \emptyset$, и $N_{1R} \neq \emptyset$.
Пусть $t_5 = \min\{s(B_i, \bar{\sigma}) | J_i \in N_1\}$ и $t_6 = \max\{C(B_i, \bar{\sigma}) | J_i \in N_1\}$. Тогда $N_{1L} \neq \emptyset$ влечет $t_1 \leq t_6$ и из $N_{1R} \neq \emptyset$ следует, что $t_5 \leq t_2$. Напомним, что на машине A в интервале $[0, t_1 - \tau]$ выполняются операции из $\{N'_{0L}\}_A \cup \{\bar{N}'_{0L}\}_A \cup \{N'_{0R}\}_A$, в интервале $[t_1, t_2]$ выполняются операции из $\{N_{1L}\}_A \cup \{N_{1R}\}_A$ и в интервале $[t_1 + \tau, C_{max}]$ выполняются операции из $\{N''_{0L}\}_A \cup \{\bar{N}''_{0R}\}_A \cup \{N''_{0R}\}_A$. В свою очередь на машине B операции из множеств $\{N'_{0L}\}_B$, $\{N'_{0R}\}_B$ и $\{\bar{N}''_{0R}\}_B$ выполняются в интервале $[0, t_5 - \tau]$, операции из множеств $\{N_{1L}\}_B$ и $\{N_{1R}\}_B$ — в интервале $[t_5, t_6]$ и остальные операции — в интервале $[t_6 + \tau, C_{max}]$. Поскольку $[0, t_1 - \tau] \cap [t_6 + \tau, C_{max}] = \emptyset$ и $[0, t_5 - \tau] \cap [t_2 + \tau, C_{max}] = \emptyset$,

используя перестановочный прием, получим порядок π_4 выполнения операций на машине B :

$$0 \to \{N'_{0R}\}_B \to \{\bar{N}''_{0R}\}_B \to \{N'_{0L}\}_B \xrightarrow{\tau} \{N_{1R}\}_B \to$$
$$\{N_{1L}\}_B \xrightarrow{\tau} \{N''_{0R}\}_B \to \{\bar{N}'_{0L}\}_B \to \{N''_{0L}\}_B \to C_{max}.$$

Лемма 40 *Пусть σ — допустимое расписание, для которого выполнены следующие условия: $\psi(A,\sigma) = 2$, $\psi(B,\sigma) \geq 2$, $N_{1L} \neq \emptyset$, и $N_{1R} \neq \emptyset$. Тогда существует расписание σ^*, в котором машина A обслуживает работы в порядке π_0, машина B обслуживает работы в порядке π_4 и выполнены условия (C1)–(C3) и (C5).*

Набор подмножеств $\mathcal{N}_1$ и перестановки π_0 и π_4 определяют конфигурацию K_3, соответствующую расписанию σ_3^*, которая представлена на рисунке 5.6.

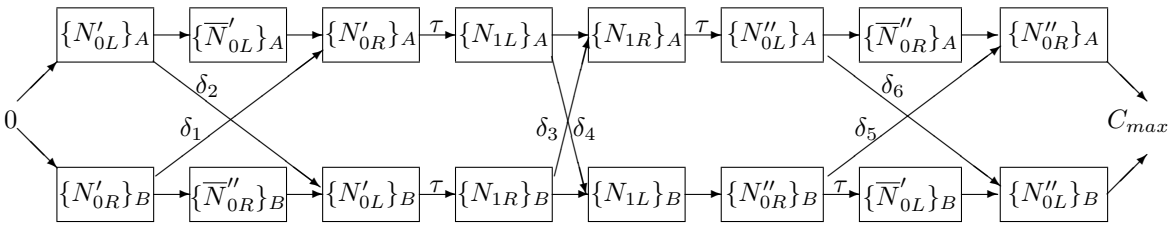

Рис. 5.6: Конфигурация K_3

Рассмотрим произвольное оптимальное расписание. Согласно леммам 38-40 оно может быть преобразовано в расписание такой же длины, принадлежащее одной из трех описанных конфигураций. Как следствие, получим следующий результат.

Теорема 32 *Пусть $J_0 \in N_1$ и $d_0 < l_{max}$. Тогда существует оптимальное расписание σ^* такое, что $\sigma^* \in K_1 \cup K_2 \cup K_3$.*

5.5.5 Построение оптимальной перестановки внутри идентичных блоков

В предыдущем разделе было установлено, что при нахождении оптимального расписания достаточно ограничиться теми, которые принадлежат одной из трех описанных конфигураций. Таким образом, достаточно найти оптимальное расписание в каждой из конфигураций и выбрать среди них наилучшее. Для нахождения оптимального расписания в данной конфигурации требуется распределить множество работ по заданным подмножествам и внутри каждого из них определить порядок выполнения операций на каждой машине. В этом разделе предположим, что распределение работ по подмножествам задано и требуется определить оптимальную перестановку операций внутри каждого блока.

Пусть p_{ij} — время выполнения работы j на машине i. Перестановкой Джонсона π_{JR} называют последовательность, в которой сначала идут работы множества $N_{\leq}$ в

порядке неубывания значений a_j, а затем — работы множества $N_>$ в порядке невозрастания значений b_j. Через $\bar{\pi}_{JR}$ обозначим последовательность, в которой работы идут в порядке, обратном к перестановке Джонсона.

Поскольку первые операции работ в каждом блоке могут выполняться одна за другой без задержек, время работы первой машины, требуемое для выполнения всех операций из одного блока, равно сумме длительностей операций и не зависит от выбранной перестановки. Следовательно, требуется найти перестановку, которая минимизирует время завершения работы второй машины. Эта задача эквивалентна задаче построения минимального по длине расписания в системе поточного типа на двух машинах ($F2||C_{max}$) [115]. Согласно введенным выше обозначениям, перестановка π_{JR} является оптимальной для работ из множеств N'_{0L}, $\bar{N}'_{0L}$, N_{1L} и N''_{0L}, а перестановка $\bar{\pi}_{JR}$ — оптимальной для работ из множеств N'_{0R}, N_{1R}, N''_{0R} и $\bar{N}''_{0R}$.

Пусть X — одно из перечисленных выше множеств и $h(X)$ — длина расписания в соответствующей задаче $F2||C_{max}$ для этого множества работ. Тогда вес дуги δ между парными блоками в конфигурациях вычисляется по формуле $\delta = h(X) - \sum_{J_j \in X}(a_j + b_j)$, что соответствует времени простоя второй машины в оптимальном расписании для задачи $F2||C_{max}$. Более того, если дуга между парными блоками X_A и X_B входит в критический путь, то общее время выполнения работ из X равно $h(X)$. Отсюда следует, что по назначению работ по множествам N'_{0L}, $\bar{N}'_{0L}$, N_{1L}, N''_{0L}, N'_{0R}, N_{1R}, N''_{0R} и $\bar{N}''_{0R}$ можно определить длину оптимального расписания в каждой конфигурации.

5.5.6 Алгоритм динамического программирования

В этом подразделе опишем общую схему алгоритма динамического программирования, строящего оптимальное решение для каждой из трех конфигураций, рассмотренных выше. Занумеруем работы из N согласно их порядку в π_{JR}. Для каждого $X \in \mathcal{N}_1$ обозначим через $[A, B, C]_X$ целочисленный вектор, определяемый по назначению работ по множествам из $\mathcal{N}_1$, где $A = \sum_{i \in X} a_i$, $B = \sum_{i \in X} b_i$ и $C = h(X)$. Тогда каждое расписание $\sigma \in K_i$, $i = 1, 2, 3$, может быть закодировано упорядоченным набором W из не более чем восьми таких векторов. Пусть $\Delta = \sum_{j=1}^{n} d_j$. Очевидно, что значения координат векторов $[A, B, C]_X$ не превосходят Δ.

Пусть $\mathcal{W}_k$ обозначает множество наборов, соответствующих допустимым расписаниям работ $J_1, \ldots, J_k$. Тогда $\mathcal{W}_0$ содержит единственный набор с нулевыми векторами.

Пусть $W \in \mathcal{W}_j$, $j = 0, ..., n-1$. Если работа J_{j+1} назначается в X, где X является одним из множеств N'_{0L}, $\bar{N}'_{0L}$, N_{1L}, N''_{0L}, то вектор $[A, B, C]_X$ преобразуется в вектор $[A+a_{j+1}, B+b_{j+1}, \max\{A+a_{j+1}, C\}]_X$, а все остальные вектора в наборе не меняются. Если работа J_{j+1} назначается в X, где X является одним из множеств N'_{0R}, N_{1R}, N''_{0R}, $\bar{N}''_{0R}$, то вектор $[A, B, C]_X$ заменяется на вектор $[A+a_{j+1}, B+b_{j+1}, \max\{B+b_{j+1}, C\}]_X$, а все остальные вектора в наборе не меняются. В конце алгоритма среди наборов из $\mathcal{W}_n$ выбирается тот, у которого длина полученного расписания наименьшая.

Ниже дадим формальное описание алгоритма. Для удобства изложения положим

$X_1 \doteq N_{1L}, X_2 \doteq N'_{0L}, X_3 \doteq N''_{0L}, X_4 \doteq \bar{N}'_{0L}, X_5 \doteq N_{1R}, X_6 \doteq N'_{0R}, X_7 \doteq N''_{0R}$ и $X_8 \doteq \bar{N}''_{0R}$.

Алгоритм 5.11 Алгоритм DP

Положить $\mathcal{W}_0 = \{[0,0,0]_{X_k}\}$, $k = 1, \ldots, 8$.
for $j = 1 \ldots, n$ **do**
  if $J_j \in N_1$ **then**
    for каждого $W \in \mathcal{W}_{j-1}$ **do**
      положив $[A,B,C]_{X_1} := [A + a_j, B + b_j, \max\{A + a_j, C\}]_{X_1}$ и $[A,B,C]_Y = [A,B,C]_Y$ для $Y \in \mathcal{N}_A \setminus X_1$, получить новый набор W' и добавить его в $\mathcal{W}_j$;
      положив $[A,B,C]_{X_5} := [A + a_j, B + b_j, \max\{B + b_j, C\}]_X$ и $[A,B,C]_Y = [A,B,C]_Y$ для $Y \in \mathcal{N}_A \setminus X_5$, получить новый набор W' и добавить его в $\mathcal{W}_j$;
  else
    for каждого $W \in \mathcal{W}_{j-1}$ **do**
      for $k = 2, \ldots, 4$ **do**
        положив $[A,B,C]_{X_k} := [A + a_j, B + b_j, \max\{A + a_j, C\}]_{X_1}$ и $[A,B,C]_Y = [A,B,C]_Y$ для $Y \in \mathcal{N}_A \setminus X_k$, получить новый набор W' и добавить его в $\mathcal{W}_j$.
      for $k = 6, \ldots, 8$ **do**
        положив $[A,B,C]_{X_k} := [A + a_j, B + b_j, \max\{B + b_j, C\}]_X$ и $[A,B,C]_Y = [A,B,C]_Y$ для $Y \in \mathcal{N}_A \setminus X_k$, получить новый набор W' и добавить его в $\mathcal{W}_j$.
for каждого $W \in \mathcal{W}_n$ и $i = 1, \ldots, 3$ **do**
  вычислить длину расписания $C_{max}(W, K_i)$ согласно конфигурации K_i. Для конфигурации K_2 рассматриваются только решения, в которых $[A,B,C]_{X_4} = [0,0,0]$ и $[A,B,C]_{X_8} = [0,0,0]$.
Выбрать W^* и i^* такие, что $C_{max}(W^*, K_{i^*}) = \min_{W \in \mathcal{W}_n, i \in \{1,2,3\}} C_{max}(W, K_i)$.

Так как алгоритм 5.11 перебирает все значения векторов, соответствующих допустимым расписаниям в искомых конфигурациях, он найдет и наилучшие решения, принадлежащие этим конфигурациям. Вычислительная сложность алгоритма определяется общим количеством вычисляемых векторов и оценивается через $O(n\Delta^{24})$ элементарных операций.

Теорема 33 *Алгоритм 5.11 строит оптимальное расписание для трудного примера задачи* $RO2||V| = 2|\widetilde{C}_{\max}$ *не более чем за* $O(n\Delta^{24})$ *элементарных операций.*

5.5.7 Вполне приближенная полиномиальная схема

Напомним, что семейство алгоритмов H_ε называется вполне приближенной полиномиальной схемой, если для любого $\varepsilon > 0$ любой алгоритм из H_ε является $(1 + \varepsilon)$-

приближенным алгоритмом и время его работы ограничено полиномом от размера входа и величины $\frac{1}{\varepsilon}$.

Вычислительная сложность алгоритма 5.11 полиномиально зависит от Δ, и, следовательно, он является псевдополиномиальным алгоритмом. Существует несколько различных способов преобразования алгоритма динамического программирования во вполне приближенную полиномиальную схему. Мы воспользуемся техникой сокращения числа рассматриваемых векторов [109]. Ее основная идея — итеративно уменьшать число рассматриваемых векторов в ходе выполнения алгоритма динамического программирования, заменяя группу "близких" друг к другу векторов одним из представителей этой группы. Для полноты изложения опишем, как применить эту технику к решению задачи $RO2||V| = 2|\widetilde{C}_{\max}$.

Пусть

$$\psi(x) = \begin{cases} 0, & x = 0, \\ \lfloor \log_{(1+\varepsilon/2n)} x \rfloor, & x > 0. \end{cases}$$

Для набора векторов $W = \{[A, B, C]_{X_k} | k = 1, \ldots, 8\}$ определим отображение $\Psi(W) = \{[\psi(A), \psi(B), \psi(C)]_{X_k} | k = 1, \ldots, 8\}$. Будем говорить, что наборы W_1 и W_2 близки друг к другу, если $\Psi(W_1) = \Psi(W_2)$. Используя введенные обозначения, для фиксированного $\varepsilon > 0$ опишем $(1 + \varepsilon)$-приближенный алгоритм (алгоритм 5.12). На каждой итерации алгоритма 5.12 кроме "сокращенного" множества набора векторов $\mathcal{W}'_i$, соответствующих допустимым расписаниям, будем хранить "полное" множество их отображений $\mathcal{V}_j$, $j = 1, \ldots, n$.

Вычислительная сложность алгоритма 5.12 определяется общим количеством вычисляемых векторов. В начале итерации j число векторов ограничено размером множества $\mathcal{V}_{j-1}$ и оценивается через $O(\log_{(1+\varepsilon/2n)} \Delta) \leq O((1+2n/\varepsilon) \ln \Delta)$. Отсюда следует, что вычислительная сложность алгоритма оценивается через $O(n((1+2n/\varepsilon)) \ln \Delta)^{24})$ и ограничена полиномом от размера входа и величины $\frac{1}{\varepsilon}$.

Осталось оценить качество полученного решения. Заметим, что мы отбрасываем набор W на итерации j, если существует другой набор W' такой, что $\Psi(W) = \Psi(W')$. Следовательно, для векторов $[A, B, C]_{X_i} \in W$ и $[A', B', C']_{X_i} \in W'$ имеют место неравенства $A' \leq A(1 + \varepsilon/2n)$, $B' \leq B(1 + \varepsilon/2n)$, $C' \leq C(1 + \varepsilon/2n)$. Из этих неравенств вытекает следующее утверждение.

Утверждение 13 *Для каждого набора $W \in \mathcal{W}_n$, полученного алгоритмом 5.11, существует набор $W' \in \mathcal{W}'_n$ такой, что для векторов $[A, B, C]_{X_i} \in W$ и $[A', B', C']_{X_i} \in W'$ имеют место неравенства $A' \leq A(1 + \varepsilon/2n)^n$, $B' \leq B(1 + \varepsilon/2n)^n$, $C' \leq C(1 + \varepsilon/2n)^n$.*

Утверждение 13 легко доказать по индукции, и доказательство может быть найдено в [154]. На предпоследнем шаге оба алгоритма 5.11 и 5.12 вычисляют длину расписания для каждого полученного набора векторов и соответствующей конфигурации. Легко проверить, что длина любого пути из 0 в C_{max} в расписании, соответ-

Алгоритм 5.12 Алгоритм FPTAS(ε)

1: Положить $\mathcal{W}'_0 = \{[0,0,0]_{X_k},\}\ k = 1,\ldots,8$.
2: **for** $j = 1\ldots,n$ **do**
3: **if** $J_j \in N_1$ **then**
4: **for** каждого $W \in \mathcal{W}_{j-1}$ **do**
5: положив $[A,B,C]_{X_1} := [A + a_j, B + b_j, \max\{A + a_j, C\}]_{X_1}$ и $[A,B,C]_Y = [A,B,C]_Y$ для $Y \in \mathcal{N}_A \setminus X_1$, получить новый набор W'
6: **if** $\Psi(W') \notin \mathcal{V}_j$, **then**
7: добавить W' в $\mathcal{W}'_j$,
8: добавить $\Psi(W')$ в $\mathcal{V}_j$.
9: положив $[A,B,C]_{X_5} := [A + a_j, B + b_j, \max\{B + b_j, C\}]_X$ и $[A,B,C]_Y = [A,B,C]_Y$ для $Y \in \mathcal{N}_A \setminus X_5$, получить новый набор W'
10: **if** $\Psi(W') \notin \mathcal{V}_j$ **then**
11: добавить W' в $\mathcal{W}'_j$,
12: добавить $\Psi(W')$ в $\mathcal{V}_j$;
13: **else**
14: **for** каждого $W \in \mathcal{W}_{j-1}$ **do**
15: **for** $k = 2,\ldots,4$ **do**
16: положив $[A,B,C]_{X_k} := [A + a_j, B + b_j, \max\{A + a_j, C\}]_{X_1}$ и $[A,B,C]_Y = [A,B,C]_Y$ для $Y \in \mathcal{N}_A \setminus X_k$, получить новый набор W'.
17: **if** $\Psi(W') \notin \mathcal{V}_j$ **then**
18: добавить W' в $\mathcal{W}'_j$,
19: добавить $\Psi(W')$ в $\mathcal{V}_j$;
20: **for** $k = 6,\ldots,8$ **do**
21: положив $[A,B,C]_{X_k} := [A + a_j, B + b_j, \max\{B + b_j, C\}]_X$ и $[A,B,C]_Y = [A,B,C]_Y$ для $Y \in \mathcal{N}_A \setminus X_k$, получить новый набор W';
22: **if** $\Psi(W') \notin \mathcal{V}_j$ **then**
23: добавить W' в $\mathcal{W}'_j$,
24: добавить $\Psi(W')$ в $\mathcal{V}_j$;
25: **for** каждого $W \in \mathcal{W}'_n$ и $i = 1,\ldots,3$ **do**
26: вычислить длину расписания $C_{max}(W,K_i)$ согласно конфигурации K_i. Для конфигурации K_2 рассматриваются только решения, в которых $[A,B,C]_{X_4} = [0,0,0]$ и $[A,B,C]_{X_8} = [0,0,0]$.
27: Выбрать $\overline{W}$ и $\bar{i}$ такие, что $C_{max}(\overline{W},K_{\bar{i}}) = \min_{W\in\mathcal{W}_n, i\in\{1,2,3\}} C_{max}(W,K_i)$.

ствующем одной из трех конфигураций, является линейной функцией от значений координат векторов $[A,B,C]_{X_i} \in W$, $i = 1,\ldots,8$.

Пусть W^* и i^* — набор векторов и номер конфигурации, соответствующие оптимальному решению, полученному алгоритмом 5.11. Тогда из утверждения 13 следует, что существует набор $W' \in \mathcal{W}'_n$, найденный алгоритмом 5.12 такой, что $C_{max}(W',i^*) \le (1+\varepsilon/2n)^n C_{max}(W^*,i^*) \le (1+\varepsilon)C_{max}(W^*,i^*)$. Согласно последнему шагу алгоритма 5.12 получаем $C_{max}(\overline{W},K_{\bar{i}}) \le C_{max}(W',i^*) \le (1+\varepsilon)C_{max}(W^*,i^*)$, что влечет следую-

щий результат.

Теорема 34 *Семейство алгоритмов FPTAS(ε) является вполне полиномиальной приближенной схемой для задачи* $RO2||V| = 2|\widetilde{C}_{\max}$.

Заключение

В диссертации исследуются задачи теории расписаний, появившиеся в 80–90-х годах прошлого века и интенсивно изучающиеся на протяжении последних трех десятилетий. Они имеют обширные приложения в организации производства, в бизнесе, в оптимизации компьютерных вычислений и в индустрии сервиса.

В первой главе вводятся основные термины и понятия, принятые в календарном планировании, теории расписаний и в целом в дискретной оптимизации. Обсуждаются подходы к изучению свойств рассматриваемых задач и методы их решения.

Вторая глава диссертации посвящена задачам теории расписаний с воспроизводимым ресурсом. Они возникают в управлении базами данных и в планировании финансовых инвестиций. Понятие воспроизводимого ресурса обобщает классические понятия возобновимого и невозобновимого ресурсов. В диссертации установлена NP-трудность различных задач теории расписаний с воспроизводимым ресурсом и предложены приближенные алгоритмы с гарантированной оценкой точности для NP-трудных вариантов задачи.

В третьей главе рассмотрены задачи построения энергетически эффективных расписаний. Появление этих задач обусловлено развитием технологий, которые позволяют ускорять работу современных вычислительных процессоров за счет дополнительных энергозатрат. Отметим, что задачи построения энергетически эффективных расписаний характеризуются нелинейной целевой функцией и требуют новых нестандартных подходов к их решению. В диссертации впервые рассмотрены неоднородные задачи, в которых объемы работ и интервалы их выполнения зависят от выбора машин, на которых они обслуживаются. Для неоднородных задач построения энергетически эффективных расписаний и для задач построения энергетически эффективных расписаний работ без прерываний разработаны приближенные алгоритмы с гарантированной оценкой точности.

Четвертая глава посвящена задачам построения расписаний с малыми задержками передачи данных, которые моделируют вычисление на параллельных компьютерах. Для рассматриваемых задач предложен алгоритм нахождения (α, β)-расписаний при одновременной минимизации длины расписания и взвешенной суммы моментов окончания работ. В диссертации также построены приближенные алгоритмы с гарантированной оценкой точности для задач с малыми задержками в иерархической коммуникационной системе.

В пятой главе рассматриваются цеховые задачи открытого типа с маршрутизацией. Данные задачи являются обобщением двух классических NP-трудных за-

дач дискретной оптимизации: цеховой задачи открытого типа и метрической задачи коммивояжера. В диссертации получены результаты по комбинаторной сложности и аппроксимируемости цеховых задачи открытого типа с маршрутизацией.

Результаты диссертации имеют теоретический характер. Они позволяют лучше понять комбинаторную структуру новых задач теории расписаний, их сложность, свойства их оптимальных решений и границы возможностей полиномиальных алгоритмов. Для всех рассмотренных NP-трудных задач найдены новые приближенные алгоритмы с лучшими известными оценками точности. При этом необходимо подчеркнуть, что все построенные алгоритмы, за исключением алгоритмов в параграфах 3.2 и 5.2, имеют низкую трудоемкость и могут быть использованы для эффективного построения хороших приближенных решений.

В диссертации получены следующие основные результаты.

1. Решен открытый вопрос о вычислительной сложности задачи построения кратчайшего расписания единичных работ на двух параллельных идентичных машинах при наличии воспроизводимого ресурса, поставленный Амиром и Капланом в 1988 году [31]. Доказано, что данная задача является NP-трудной в сильном смысле. Установлена также NP-трудность в сильном смысле следующих задач с воспроизводимым ресурсом:

- задача построения кратчайшего расписания единичных работ на двух параллельных машинах с фиксированным распределением работ по машинам,
- задача минимизации суммы моментов завершения работ на одной машине,
- задача минимизации взвешенной суммы моментов завершения единичных работ на одной машине.

2. Получены первые приближенные алгоритмы с гарантированными оценками точности для задач построения энергетически эффективных расписаний, в которых запрещены прерывания работ.

3. Разработан общий подход к построению приближенных алгоритмов для задач построения энергетически эффективных расписаний, основанный на решении задач линейного программирования, в которых число переменных не ограничено полиномом от размера входа задачи, с последующим вероятностным округлением полученных решений. Как следствие, впервые получены приближенные алгоритмы с гарантированной оценкой точности для неоднородных задач на минимизацию расхода энергии.

4. Для задачи составления расписания работ на параллельных машинах в иерархической коммуникационной сети с малыми коммуникационными задержками предложен первый алгоритм, который находит приближенное решение с гарантированной оценкой точности меньше тривиальной оценки, равной двум, а, именно, $12(\Phi +$

$1)/(12\Phi+1) < 2$, где $\Phi \geq 1$ — отношение длительности минимальной работы к длине максимальной задержки.

5. Предложены новые приближенные алгоритмы с гарантированными оценками точности для различных NP-трудных вариантов цеховой задачи открытого типа с маршрутизацией машин, которые имеют лучшие оценки точности среди известных алгоритмов одинаковой с ними трудоемкости.

6. Построен точный псевдополиномиальный алгоритм для двухмашинной цеховой задачи открытого типа с маршрутизацией для NP-трудного случая, когда работы и машины расположены на двухвершинной сети. Как следствие, решен открытый вопрос о вычислительной сложности этой задачи.

Литература

[1] Баптист Ф., Карлье Ж., Керан М., Кононов А.В., Севастьянов С.В., Свириденко М. Структурные свойства оптимальных расписаний с прерываниями операций // Дискретный анализ и исследование операций. — 2009. — Т 16, N 1. — С. 3–36.

[2] Гимади Э.Х., Глебов Н.И. Экстремальные задачи принятия решений. — Новосибирск: Изд-во Новосиб. ун-т, 1982.

[3] Гимади Э.Х., Залюбовский В.В., Севастьянов С.В. Полиномиальная разрешимость задач календарного планирования со складируемыми ресурсами и директивными сроками // Дискретный анализ и исследование операций. Серия 2. — 2000. — Т. 7. N 1. — С. 9–34.

[4] Гэри М., Джонсон Д. Вычислительные машины и труднорешаемые задачи. — М.: Мир, 1982. — 416 с.

[5] Зорич В. А. Математический анализ. Часть I. — 6-е изд. — М.: МЦНМО, 2012. — С. 289–290.

[6] Казаковцева Е. А., Сервах В. В. Кредитование и анализ надежности расписаний в задаче календарного планирования проектов // Автоматика и телемеханика. — 2014. — Т. 7. — С. 87–98.

[7] Каширских К. Н., Кононов А. В., Севастьянов С. В., Черных И. Д. Полиномиально разрешимый случай двухстадийной задачи open shop с тремя машинами // Дискретный анализ и исследование операций. Серия 1. — 2001. — Т. 8. N 1. — С. 24–40.

[8] Кононов А.В. О расписаниях работ на одной машине с длительностями, нелинейно зависящими от времени. // Дискретный анализ и исследование операций. — 1995. — Т 2, N 1. — С. 21–35.

[9] Кононов А.В. Комбинаторная сложность составления расписаний для работ с простым линейным ростом длительностей // Дискретный анализ и исследование операций. — 1996. — Т 3, N 2. — С. 15–32.

[10] Кононов А.В. Задачи теории расписаний на одной машине с длительностями работ, пропорциональными произвольной функции // Дискретный анализ и исследование операций. — 1998. — Т 5, N 3. — С. 17–37.

[11] Кононов А.В. О цеховой задаче открытого типа на двух машинах с маршрутизацией в двухвершинной сети // Дискретный анализ и исследование операций. — 2012. — Т 19, N 2. — С. 54–74.

[12] Кононов А. В., Севастьянов С. В. О сложности нахождения связной предписанной раскраски вершин графа // Дискретный анализ и исследование операций. Серия 1. — 2000. — Т. 7. N 2. — С. 21–46.

[13] Козлов М.К., Шафранский В.В. Календарное планирование выполнения комплексов работ при заданной динамике поступления складируемых ресурсов // Изв. АН СССР. Техническая кибернетика. — 1977. — Т. 4. — С. 75–81.

[14] Мартынова Е. А., Сервах В. В. О задаче календарного планирования проектов с использованием кредитов // Автоматика и телемеханика. — 2012. — Т. 3. — С. 107–116.

[15] Пападимитриу Х., Стайглиц К. Комбинаторная оптимизация. Алгоритмы и сложность. — М.: Мир, 1985. — 512 с.

[16] Севастьянов С.В. Приближенные алгоритмы в задачах Д жонсона и суммирования векторов // Управляемые системы. — 1980. — Т. 20 — С. 64–73.

[17] Севастьянов С.В. Геометрия в теории расписаний, модели и методы оптимизации // Труды Института математики. — 1988. — Т. 10. — С. 226–261.

[18] Севастьянов С.В. Нестрогое суммирование векторов в задачах теории расписаний // Сибирский журнал исследования операций. — 1994. — Т. 1(2). — С. 67–99.

[19] Сервах В. В., Сухих С. Л. Гибридный алгоритм для задачи календарного планирования с учетом реинвестирования прибыли // Автоматика и телемеханика. — 2004. — Т. 3. — С. 100–107.

[20] Сердюков А. И. О некоторых экстремальных обходах в графах // Управляемые системы: Сб. науч. тр. — 1984. — N 17. — С. 76–79.

[21] Танаев В.С., Гордон В.С., Шафранский Я. М. Теория расписаний. Одностадийные системы. — М.: Наука. Гл. ред. физ.-мат. лит., 1984. — 384 с.

[22] Танаев В.С., Ковалев М.Я., Шафранский Я. М. Теория расписаний. Групповые технологии. — Минск: Институт технической кибернетики НАН Беларуси, 1998. — 290 с.

[23] Танаев В.С., Сотсков Ю.Н., Струсевич В. А. Теория расписаний. Многостадийные системы. — М.: Наука. Гл. ред. физ.-мат. лит., 1989. — 329 с.

[24] Танаев В.С., Шкурба В.В. Введение в теорию расписаний. — М.: Наука, 1975. — 256 с.

[25] Хачиян Л.Г. Полиномиальный алгоритм в линейном пр ограммировании // ДАН СССР — 1979. — Т. 244, N 5. — С. 1093–1096.

[26] Шкурба В.В., Подчасова Т.П., Пшичук А.Н., Тур Л.П. Задачи календарного планирования и методы их решения. — Киев: Наукова думка, 1966. — 155 с.

[27] Ageev A., Fishkin A., Kononov A., Sevastianov S. Open Block Scheduling in Optical Communication Networks. // Theoretical Computer Science. — 2006. — V. 361. — P. 257–274.

[28] Albers S. Energy-efficient algorithms // Communications of the ACM — 2010. — Vol. 53(5) — P. 86–96.

[29] Albers S., Antoniadis A., and Greiner G. On multi-processor speed scaling with migration: extended abstract // Proceedings of 23rd ACM Symposium on Parallelism in Algorithms and Architectures (SPAA 2011) — 2011. — P. 279–288.

[30] Albers S., Müller F., and Schmelzer S. Speed scaling on parallel processors // Proceedings of 19th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA 2007) — 2007. — P. 289–298.

[31] Amir A., Kaplan E.H. Relocation problems are hard // International Journal of Computer Mathematics — 1988. — Vol. 25 — P. 101–110.

[32] Angel E., Bampis E., and Chau V. Throughput maximization in the speed-scaling setting // *CoRR*, abs/1309.1732, 2013.

[33] Angel E., Bampis E., Kacem F., and Letsios D. Speed scaling on parallel processors with migration // Proceedings of 18th International European Conference on Parallel and Distributed Computing (Euro-Par 2012), Lecture Notes in Computer Science, Berlin: Springer — 2012. — Vol. 7484 — P. 128–140.

[34] Angel E., Bampis E., Kononov A. On the approximate tradeoff for bicriteria batching and parallel machine scheduling problems. // Theoretical Computer Science. — 2003. — V. 306, N 1-3 — P. 319–338.

[35] Anderson T.E., Culler D.E., Patterson D.A. and the NOW Team. A Case for NOW (Network of Workstations) // IEEE Micro — 1995. — Vol. 15 — P. 54–64.

[36] Antoniadis A., Huang C.-C. Non-preemptive speed scaling // Proceedings of 13th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT 2012), Lecture Notes in Computer Science, Berlin: Springer — 2012. — Vol. 7357 — P. 249–260.

[37] Aslam J., Rasala A., Stein C., Young N. Improved bicriteria existence theorems for scheduling // Proceedings of ACM-SIAM Symposium on Discrete Algorithms — 1999. — P. 846–847.

[38] Averbakh I., Berman O. Routing Two-Machine Flowshop Problems on Networks with Special Structure // Transportation Science — 1996. — Vol 30(4) — P. 303–314.

[39] Averbakh I., Berman O. A Simple Heuristic for m-machine Flow-Shop and its Applications in Routing-Scheduling Problems // Operations Research — 1999. — Vol. 47(1) — 165–170.

[40] Averbakh I., Berman O., Chernykh I. A $\frac{6}{5}$-approximation algorithm for the two-machine routing open shop problem on a 2-node network // European Journal of Operational Research — 2005. — Vol. 166(1) — P. 3–24.

[41] Averbakh I., Berman O., Chernykh I. The Routing Open-Shop Problem on a Network: Complexity and Approximation // European Journal of Operational Research — 2006. — Vol. 173(2) — P. 521–539.

[42] Bampis E., Giroudeau R., König J.-C. A Heuristic for the Precedence Constrained Multiprocessor Scheduling Problem with Hierarhical Communications // In H. Reichel and S. Tison (eds.), Proceedings of STACS, Lecture Notes in Computer Science, Berlin: Springer — 2000. — Vol. 1770 — P. 443–454.

[43] Bampis E., Giroudeau R., König J.-C. On the Hardness of Approximating the Precedence Constrained Multiprocessor Scheduling Problem with Hierarchical Communications // RAIRO-RO — 2002. — Vol. 36 — P. 21–36.

[44] Bampis E., Giroudeau R., A. Kononov. Scheduling Tasks with Small Communication Delays for Clusters of Processors // Annals of Operations Research — 2004. — Vol. 129 — P. 47–63.

[45] Bampis E., Kononov A. Bicriteria Approximation Algorithms for Scheduling Problems with Communication Delays. // Journal of Scheduling. — 2005. — V.8, N 4. — P. 281–294.

[46] E. Bampis, A. Kononov, D. Letsios, G. Lucarelli, I. Nemparis, From preemptive to non-preemptive Speed-Scaling Scheduling. // COCOON 2013, Lecture Notes in Computer Science — 2013. — V. 7936 — p. 134–146.

[47] Bampis E., Kononov A., Letsios D., Lucarelli G., Sviridenko M. Energy efficient scheduling and routing via randomized rounding // In 33rd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2013), LIPIcs. Schloss Dagstuhl — Leibniz-Zentrum fuer Informatik — 2013.

[48] Bampis E., Letsios D., Milis I., Zois G. Speed scaling for maximum lateness. // In 18th Annual International Computing and Combinatorics Conference (COCOON 2012), Lecture Notes in Computer Science, Berlin: Springer — 2012. — Vol. 7434 P. 25–36.

[49] Bansal N., Kimbrel T., Pruhs K. Speed Scaling to Manage Energy and Temperature // Journal of the ACM — 2007. — Vol. 54, No. 1.

[50] Bansal N., Mahdian M., Sviridenko M. Minimizing makespan in no-wait job shops // Mathematics of Operations Research — 2006. — Vol. 30 — P. 817–831.

[51] Baptiste Ph., Carlier J., Kononov A., Queyranne M., Sevastyanov S., Sviridenko M. Properties of optimal schedules in preemptive shop scheduling // Discrete Applied Mathematics — 2011. — Vol. 159 — P. 272 – 280.

[52] Baptiste Ph., Carlier J., Kononov A., Queyranne M., Sevastianov S., Sviridenko M. Integer preemptive scheduling on parallel machines. // Operations Research Letters. — 2012. — V. 40, N 6. — P. 440–444.

[53] Bárány I., Fiala T., Többgépes ütemezési problémák közel optimális megoldása // Szigma-Mat.-Közgazdasági Folyóriat — 1982. — Vol. 15 — P. 177–191.

[54] Berend D., Tassa T. Improved bounds on Bell numbers and on moments of sums of random variables // Probability and Mathematical Statistics — 2010. — Vol. 30 — P. 185–205.

[55] Bhatt S.N., Chung F.R.K., Leighton F.T., and Rosenberg A.L. On optimal Strategies for Cycle-Stealing in Networks of Workstations // IEEE Transactions on Computers — 1997. — Vol. 46 — P. 545–557.

[56] Bingham B. D., Greenstreet M. R. Energy optimal scheduling on multiprocessors with migration. // In International Symposium on Parallel and Distributed Processing with Applications (ISPA 2008), IEEE — 2008. — P. 153–161.

[57] Blażewicz J., Barcelo J., Kubiak W., Röck H. Scheduling tasks on two processors with deadlines and additionsl resources // European Journal of Operational Research — 1986. — Vol. 26 (3) — P. 364 – 370.

[58] Blażewicz J., Brauner N., Finke G. Scheduling with Discrete Resource Constraints // Handbook of Scheduling. Algorithms, Models, and Performance Analysis, Ch. 23. — Boca Raton, London, New York, Washington, D.C.: Chapman & Hall, — P. 23-1 – 23-18.

[59] Blażewicz J. and Ecker K. A linear time algorithm for restricted bin packing and scheduling problems // Operations Research Letters — 1983. — Vol. 2 (2) — P. 80–83.

[60] Blażewicz J., Lenstra J.K., Rinnoy Kan A.H.G. Scheduling subject to resource constraints: classification and complexity // Discrete Applied Mathematics — 1983. — Vol. 5 (1) — P. 11–24.

[61] Blumafe R., Park D.S. Scheduling on Networks of Workstations // In 3rd Internat. Sympos. of High Performance Distr. Computing — 1997. — P. 96–105.

[62] Bo Chen, Potts C.N., Woeginger G. J. A Review of Machine Scheduling: Complexity, Algorithms and Approximability // In Handbook of Combinatorial Optimization, D.-Z. Du and P. M. Pardalos(Eds), Kluwer Academic Publisher, Amsterdam (1998), Vol. 3 — P. 21–169.

[63] Böttcher J., Drexl A., Salewski F. Project scheduling under partially renewable resource constraints // Management Sci. — 1999. — Vol. 112, N 1 — P. 3–41.

[64] Brucker, P. Scheduling algorithms — Springer-Verlag, Berlin, Heidelberg, Germany — 1998.

[65] Brucker, P., & Knust, S. Operations research: Complexity results of scheduling problems, // http://www. mathematik.uni-osnabrueck.de/research/OR/class/.

[66] Brucker P., Drexl A., Möhring R., Neumann K., Pesch E. Resource-constrained project scheduling: Notation, classification, models, and methods // Europian J. Oper. Res. — 1999. — Vol 112, N 1 — P. 3–41.

[67] Cappelo F., Fraignaud P., Mans B., Rosenberg A.L. HiHCoHP-Towards a Realistic Communication Model for Hierarhical HyperCluster of Heterogenous Processors // Proceedings of 15th International Parallel and Distributed Processing Symposium (IPDPS'01) — 2000. — Vol 1. — P. 10042a.

[68] Carlier, J., & Pinson E., An algorithm for solving the job-shop problem // Management Science — 1989. — Vol. 35, N 2 — P. 164–176.

[69] Chandrasekaran K., Goyal N., Haeupler B. Deterministic Algorithms for the Lovasz Local Lemma // Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms — 2010. — pp. 992–1004.

[70] Cheng T.C.E., Lin B.M.T. Johnson's rule, composite jobs and the relocation problem // European Journal of Operational Research — 2009. — Vol 192, N 3 — P. 1008–1013.

[71] Chernykh I., Dryuck N., Kononov A., Sevastyanov S. The Routing Open Shop Problem: New Approximation Algorithms // Approximation and Online algorithms: 7th International Workshop, WAOA 2009, (Copenhagen, Denmark, September 10-11, 2009). Revised Papers, Eds: Evripidis Bampis and Klaus Jansen – Heidelberg, Springer-Verl., 2010. — P. 75–85. (Lect. Notes Comp. Sci.; Vol. 5893.)

[72] Chernykh I., Kononov A., Sevastyanov S. Efficient approximation algorithms for the routing open shop problem. // Computers & Operations Research. — 2013. — V. 40, N 3. — P. 841–847.

[73] Chou S.-Y., Lin S.-W., Museum visitor routing problem with the ballancing of concurrent visitors // Complex Systems Concurrent Engineering – 2007. — Vol. 6 — P. 345–353.

[74] Christofides N. Worst-Case Analysis of a New Heuristic for the Travelling Salesman Problem. — Report 388, Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, PA, 1976.

[75] Chrétienne P., Coffman Jr. E.J., Lenstra J.K., and Liu. Z. Scheduling Theory and Its Applications. — New-York: Wiley, 1995.

[76] Chrétienne P. and Colin J.Y. Scheduling with Small Interprocessor Communication Delays // Operations Research — 1991. — Vol. 39, N 3 — P. 680–684.

[77] Coffman Jr. E.G., Garey M.R., Johnson D.S. An application of bin-packing to multiprocessor scheduling // SIAM Journal on Computing — 1978. — Vol. 7 — P. 1–17.

[78] Cole R., Ost K., Schirra S. Edge-coloring bipartite multigraphs in $O(E \log D)$ time. // Combinatorica — 2001. — Vol. 21 N. 1 — P. 5–12.

[79] Conway R.W., Maxwell W.L., Miller L.W. Theory of Scheduling. — Addison-Wesley, Reading, MA, 1967.

[80] Cook S. A. The complexity of theorem proving procedures // Proceedings of the 3rd Annual ACM Symposium on the theory of Computing — 1971. — P. 151–158.

[81] Dantzig G.B. Discrete variable extremum problems // Operations Research. — 1957. — Vol. 5. — P. 266–277.

[82] Darte A., Robert Y., Vivien F. Scheduling and Automated Parallelization. — Birkhäuser, Boston, 2000.

[83] Desrosiers J., Dumas Y., Solomon M., and Soumis F. Time Constrained Routing and Scheduling // Handbooks in Operations Research and Management Science, M.O.Ball, T.L.Magnanti, C.L.Monma and G.L.Nemhauser (Eds.), Vol. 8 — Network Routing. North-Holland — 1995. — P. 35–139.

[84] Drozdowski M. Scheduling for parallel processing.— London: Springer-Verlag, 2009.

[85] Edmonds J. Paths, trees, and flowers // Canadian Journal of Mathematics. — 1965. — Vol. 7 — P. 449–467.

[86] Fermandez de la Vega W., Lueker G.S. Bin packing can be solved within $1 + \varepsilon$ in linear time // Combinatorica, — 1981. — Vol. 1 — P. 349–355.

[87] Fischetti M., Laporte G., and Martello S. The Delivery Man Problem and Cumulative Matroids // Operations Research — 1993. — Vol. 41(6) — P. 1055–1064.

[88] Gabow H.N. Data structures for weighted matching and nearest common ancestor with linking // In: Proc. of the 1st Annual ACM-SIAM Symp. on Discrete Algorithms — 1990.

[89] Gabow H., Kariv O. Algorithms for edge coloring bipartite graphs and multigraphs. SIAM Journal of Computing — 1982. — Vol. 11 — P. 117–129.

[90] Gawiejnowicz S., Kononov A. NP-hard Cases in Scheduling Deteriorating Jobs on Dedicated Machines. // Journal of the Operational Research Society. — 2001. — V.52. — P. 708–717.

[91] Gawiejnowicz S., Kononov A. Complexity and approximability of scheduling resumable proportionally deteriorating jobs. // European Journal of Operational Research. — 2010. — V. 200, N 1. — P. 305–308.

[92] Gawiejnowicz S., Kononov A. Isomorphic scheduling problems. // Annals of Operations Research. — 2014. — V. 213. — P. 131-145.

[93] Garey M. and Johnson D. Computers and Intractability: A Guide to the theory of NP-completeness. — San Francisco, CA: W.H. Freemann and Company, 1979.

[94] Gens G.V., Levner E.V. Approximation algorithms for certain universal problems in scheduling theory // Engineering Cybernetics — 1978. — Vol. 16 — P. 31–36.

[95] Gens G.V., Levner E.V. Fast approximation algorithm for job sequencing with deadlines // Discrete Applied Mathematics — 1981. — Vol. 3 — P. 313–318.

[96] Gilmore P.C., Lawler E.L., and Shmoys D.B. Well-solved special cases, In: E.L. Lawler, J.K. Lenstra, A.H.G. Rinnoy Kan, D.B. Shmoys, eds., The Travelling Salesman Problem, 1985, John Wiley & Sons Ltd.

[97] Gonzalez T., Sahni S. Open Shop Scheduling to Minimize Finish Time // Journal of the Association for Computing Machinery — 1976. — Vol. 23 — P. 665–679.

[98] Graham R.L. Bounds for Certain Multiprocessing Anomalies // Bell System Technical Journal — 1966. — Vol. 45 — P. 1563–1581.

[99] Graham R.L., Lawler E.L., Lenstra J.K., Rinnooy Kan A.H.G. Optimization and approximation in determenistic scheduling: a servey // Annals of Discrete Mathematics — 1979. — Vol. 5 — P. 287–326.

[100] Greiner G., Nonner T., and Souza A. The bell is ringing in speed-scaled multiprocessor scheduling // In 21st ACM Symposium on Parallelism in Algorithms and Architectures (SPAA 2009), ACM — 2009. — P. 11–18.

[101] Grigoriev A., Kononov A., Sviridenko M. Logarithmic approximations for the relocation problem // 1st International Symposium & 10th Balkan Conference on Operational Research, September 22-24, 2011, Thessaloniki, Greece, University of Macedonia, Economic and Social Sciences, Proceedings Volume, — 2011. — p. 263–269.

[102] Grötschel M., Lovas L., Schrijver A. Geometric Algorithms and Combinatorial Optimization. — Springer-Verlag, 1988.

[103] Hall L.A. Approximability of flow shop scheduling // Mathematical Programming — 1998. — Vol. 82 — P. 175–190.

[104] Herroelen W., Demeulemeester E., De Reyk B. A classification scheme for project scheduling // Project Scheduling. Recent Models, Algorithms and Applications. Ch. 1. — Boston, London, Dordrecht: Kluwer Academic Publisher, 1999. — P. 1–26.

[105] Handbook of Scheduling: Algorithms, Models, and Performance Analysis, edited by Y-T.Leung, Chapman & Hall/CRC Computer and Information Science Series, CRC Press Company, 2004.

[106] Hochbaum D.S. and Shmoys D.B. Using dual approximation algorithms for scheduling problems: practical and theoretical results // Journal of the Association for Computing Machinery — 1987. — Vol. 34(1) — P. 144–162.

[107] Hoeffding W. On the distribution of the number of successes in independent trials // Annals of Mathematical Statistics — 1956. — Vol. 27 — P. 713–721.

[108] Hoogeveen J.A., Lenstra J.K. and Veltman B. Three, four, five, six, or the complexity of scheduling with communication delays // Operations Research Letters — 1994. — Vol. 16 — P. 129–137.

[109] Ibarra O.H., Kim C.E. Fast approximation algorithm for the Knapsack and sum of subset problems // Journal of the Association for Computing Machinery — 1975. — Vol. 22 — P. 463–468.

[110] Ibarra O.H., Kim C.E. Approximation algorithms for certain scheduling problems // Mathematics of Operations Research — 1978. — Vol. 3 — P. 280–289.

[111] Irani S., Pruhs K. Algorithmic problems in power management // SIGACT News — 2005. — Vol. 36(2) — P. 63–76.

[112] Jackson J.R. Scheduling a production line to minimize maximum tardiness. — Research Report 43, Management Science Research Project, University of California, Los Angeles, USA, 1955.

[113] Jackson J.R. An extension of Johnson's results on job lot scheduling // Naval Research Logistics Quaterly. — 1956. — Vol. 3. — P. 201–203.

[114] Jansen K., Solis-Oba R., Sviridenko M. Makespan Minimization in Job Shops: a Linear Time Approximation Scheme // SIAM Journal of Discrete Mathematics — 2003. — Vol. 16 — P. 288–300.

[115] Johnson S.M. Optimal two- and three stage production schedules with set-up times included // Naval Research Logistic Quaterly — 1954. — Vol. 1. — P. 61–68.

[116] Johnson T.J.R. An algorithm for the resource-constrained project scheduling problem // Ph.D. Dissertation — MIT, Boston, USA, 1967.

[117] Kaplan E.H. Relocation models for public housing redevelopment programs // Planning and Design — 1986. — Vol.13. — P. 5–19.

[118] Kaplan E.H. and Amir A. A fast feasibility test for relocation problems // European Journal of Operational Research — 1988. — Vol. 38. — P. 201–205.

[119] Kaplan E.H. and Berman O. Orient heights housing projects // Interfaces. — 1988. — Vol. 18. — P. 14–22.

[120] Karp R. M. Reducibility among Combinatorial Problems // in Complexity of Computer Computations, ed. R.E. Miller and J.W. Thatcher. New York: Plenum Press — 1972. — P. 85–103.

[121] Karuno Y., Nagamochi H., and Ibaraki. T. Vehicle Scheduling on a Tree with Release and Handling Times // Annals of Operations Research — 1997. — Vol. 69 — P. 193–207.

[122] Kelley J.E. Computers and Operations Research in Road Building // Operations Research, Computing and Management Decisions, Symposium Proceedings, Case Institute of Technology (Jan. 31, Feb. 1, 2, 1957) — 1957. — P. 58–68.

[123] Kelley J.E., Walker M.R. Critical Path Planning and Scheduling: An Introduction. — Mauchly Associates, Ambler PA, 1959.

[124] Kononov A., Hong J-S., Kononova P., Lin F-C. Quantity-based buffer-constrained two machine flowshop problem: active and passive prefetch models for multimedia applications. // Journal of Scheduling. — 2012. — V. 15. — P. 487-497.

[125] Kononov A., Lin B.M.-T. Relocation Problems with Multiple Working Crews // Discrete Optimization — 2006. — Vol. 3. — P. 366–381.

[126] Kononov A., Lin B.M.-T. Minimizing the total weighted completion time in the relocation problem // Journal of Scheduling. — 2010. — Vol. 13. N. 2. — P. 123 – 129.

[127] Kononov A., Sevastyanov S. Graph Structure Analysis and Computational Tractability of Scheduling Problems // In: Analysis of Complex Networks: From Biology to Linguistics. Edited by M. Dehmer and F. Emmert-Streib, 2009, Wiley-VCH Verlag Gmbh & Co. KGaA, Weinheim ISBN 978-3-527-32345-6, P. 295–322.

[128] Kononov A., Sevastianov S., Tchernykh I. When the difference in machine loads leads to efficient scheduling in open shops. // Annals of Operations Research. — 1999. — V. 92. — P. 211–239.

[129] Kononov A., Sviridenko M. Linear Time Combinatorial Approximation Scheme for Open Shop Problem with Release Dates // Operations Research Letters — 2002. — Vol. 30 — P. 276–280.

[130] Korte B., Vygen Y. Combinatorial Optimization. Theory and Algorithms. — Berlin, Heidelberg: Springer, 2000.

[131] Lawler E.L., Lenstra J.K., Rinnooy Kan A.H.G., and Shmoys D.B. Sequencing and Scheduling: Algorithms and Complexity // in Handbooks in Operations Research and Management Science, Volume 4, Logistics of Production and Inventory, ed. S. Graves, A.H.G. Rinnooy Kan, and P. Zipkin. North Holland, Amsterdam. — 1993. — P. 445–522.

[132] Lenstra J.K., Rinnooy Kan A.H.G., and Brucker P. Complexity of machine scheduling problems // Annals of Operations Research. — 1977. — Vol. 1. — P. 343–362.

[133] Leighton F. T., Maggs B. M., and Rao S. B. Packet routing and job-scheduling in O(Congestion + Dilation) steps. // Combinatorica. — 1994. — Vol. 14 — P. 167–186.

[134] Leighton F. T., Maggs B. M., and Richa A. W. Fast algorithms for finding O(Congestion + Dilation) packet routing schedules // Combinatorica. — 1999. — Vol. 19 (3) — P. 375–401.

[135] Li M., Yao A.C., Yao F.F. Discrete and continuous min-energy schedules for variable voltage processors. // Proceedings of the National Academy of Sciences USA. — 2006. — Vol. 103. — P. 3983–3987.

[136] Lin B.M.-T., Kononov A. Customer Order Scheduling to Minimize the Number of Late Jobs. // European Journal of Operational Research. — 2007. — V. 183, N 2. — P. 944-948.

[137] Lin B.M.T., Liu S.T. Maximizing the reward in the relocation problem with generalized due dates // International Journal of Production Economics. — 2008. — Vol. 115 (1) — P. 55–63.

[138] Malcolm D.J., Roseboom J.H., Clark C.E., Fazar W. Application of a Technique for Research and Development Program Evaluation // Operations Research. — 1959. — Vol. 9. N 5. — P. 646–669.

[139] Möhring R.H., Schäffter M.W. and Schulz A.S. Scheduling jobs with communication delays: Using infeasible solutions for approximation // in Algorithms-ESA '96, Lecture Notes in Comput. Sci. 1136, J. Diaz and M. Serna, eds., Springer, Berlin — 1996. — P. 76–90.

[140] Moser R. A. and Tardos G. A constructive proof of the general Lovasz Local Lemma // Journal of the ACM. — 2010. — Vol. 57 (2) — P. 11:1–11:15.

[141] Munier A. and Hanen C. An approximation algorithm for Scheduling Dependent Tasks on m Processors with Small Communication Delays // Discrete Applaid Mathematics. — 2001. — Vol. 108, N 3 — P. 239–257.

[142] Munier A. and König J.-C. A heuristic for a scheduling problem with communication delays // Operations research. — 1997. — Vol. 45 — P. 145–147.

[143] Papadimitriou C., Yannakakis M. Optimization, approximation and complexity classes // Journal of Computer and System Science. — 1991. — Vol 43. — P. 425–440.

[144] Pfister G.F. In search of clusters. — New York: Prentice Hall, 1995.

[145] Peis B. and Wiese A. Universal Packet Routing with Arbitrary Bandwidth and Transit Times // in: O. Günlük, G. Woeginger (Eds.), the Proceedings of IPCO 2011: Lecture Notes in Comp. Sci. — 2011. — Vol. 6655 — P. 362–375.

[146] Potts C.N. Analysis of a heuristic for one machine sequencing with release dates and delivery times // Operations Research. — 1980. — Vol. 28. — P. 1436–1441.

[147] Rayward-Smith V.J. UET scheduling with unit interprocessor communication delays // Discrete Applayd Mathematics — 1987. — Vol. 18 — P. 55–71.

[148] Rasala A. Existence theorems for scheduling to meet two objectives // Technical report PCS-TR 99-347, Dartmouth College, Hanover, NH, 1999.

[149] Sahni S. Algorithms for scheduling independent tasks // Journal of the Association for Computing Machinery — 1976. — Vol. 23 — P. 116–127.

[150] Schmidt J.P., Siegel A., Srinivasan A. Chernoff-Hoeffding bounds for applications with limited independence // Proceedings of the Forth Annual ACM-SIAM Symposium on Discrete Algorithms, philadelphia, PA, ACM/SIAM. — 1993. — P. 331–340.

[151] Schrijver, A. Combinatorial optimization. Polyhedra and efficiency // Algorithms and Combinatorics — Berlin: Springer-Verlag, 2003.

[152] Schulz A.S. Polytopes and Scheduling — PhD thesis, Technical University of Berlin, Berlin, Germany, 1995.

[153] Schulz A.S. Scheduling to minimize total weighted completion time: Performance guarantees of LP-based heuristics and lower bounds // Lect. Notes Comp. Sci. — 1996. — Vol. 1084. — P. 301–315.

[154] Schuurman P., Woeginger G.: Approximation schemes – a tutorial. In Lecture in Scheduling: R.H. Moehring, C.N. Potts, A.S. Schulz, G.J. Woeginger, L.A. Wolsey eds., 2002.

[155] Sevast'janov S.V. On Some Geometric Methods in Scheduling Theory: a survey // Discrete Appl. Math. — 1994 — Vol. 55. — P. 59–82.

[156] Sevastyanov S. V., Lin, B. M. T., Huang, H. L. Tight complexity analysis of the relocation problem with arbitrary release dates // Theoretical Computer Science — 2011. — Vol. 412. — P. 4536–4544.

[157] Sevastyanov S. V., Woeginger G.J. Makespan minimization in open shops: A polynomial time approximation schemes // Mathematical Programming — 1998. — Vol. 82 — P. 191–198.

[158] Shmoys D., Stein C., and Wein J. Improved approximation algorithms for shop scheduling problems // SIAM Journal on Computing. — 1994. — Vol. 23, N 3 — P. 617–632.

[159] Sinen O. Task Scheduling for Parallel Systems. — New Jersey: Wiley, Hoboken, 2007.

[160] Skutella M. List scheduling in order of α-points on a single machine // Efficient Approximation and Online Algorithms: Recent Progress on Classical Combinatorial Optimization Problems and New Applications in: Lecture Notes in Comp. Sci. — 2006. — Vol. 3484. — P. 250 – 291.

[161] Smith W.E. Various optimizers for single stage production // Naval Research Logistics Quaterly. — 1956. — Vol. 3. — P. 59–66.

[162] Stein C., Wein J. On the existence of schedules that are near-optimal for both makespan and total weighted completion time // Operations Research Letters — 1997. — Vol. 21, N 3 — P.115–122.

[163] Wiest, J. D. Some properties of schedules for large projects with limited resources // Oper. Res. — 1964. — Vol. 12. — P. 395–418.

[164] Williamson D., Hall L., Hoogeveen J., Hurkens C., Lenstra J.K., Sevastianov S., Shmoys D. Short shop schedules // Operations Research. — 1997. — Vol. 45, N. 2. — P. 288–294.

[165] Xie J.-X. Polynomial algorithms for single machine scheduling problems with financial constraints. // Operations Research Letters. — 1997. — Vol. 21. N. 1 — P. 30–42.

[166] Yao F., Demers A., Shenker S. A scheduling model for reduced CPU energy. // In 36th Annual Symposium on Foundation of Computer Science (FOCS 1995) — 1995. — P. 374–382.

[167] Yu V. F., Lin S.-W., Chou S.-Y. The museum visitor routing problem // Applied Mathematics and Computation — 2010. — Vol. 216(3) — P. 719–729.

[168] Yu W. and Zhang G. Improved approximation algorithms for routing shop scheduling // The Proceedings of of ISAAC 2011, in: Lecture Notes in Comp. Sci. — 2011. — Vol. 7074. — P. 30–39.