ISSN 1816-0301

ИНФОРМАТИКА

4 (36)

ОКТЯБРЬ-ДЕКАБРЬ
2012

Адрес редакции:
220012, Минск,
ул. Сурганова, 6, к. 305
тел. (017) 284-26-22
e-mail: rio@newman.bas-net.by
http://uiip.bas-net.by

ОБЪЕДИНЕННЫЙ ИНСТИТУТ ПРОБЛЕМ ИНФОРМАТИКИ
НАЦИОНАЛЬНОЙ АКАДЕМИИ НАУК БЕЛАРУСИ

ИНФОРМАТИКА

ЕЖЕКВАРТАЛЬНЫЙ НАУЧНЫЙ ЖУРНАЛ

Издается с января 2004 г.

№ 4(36) • октябрь-декабрь 2012

СОДЕРЖАНИЕ

КОНФЕРЕНЦИИ

Редактор Г.Б. Гончаренко
Корректор А.А. Михайлова
Компьютерная верстка В.И. Голенкевич

Сдано в набор 12.11.2012. Подписано в печать 06.12.2012.
Формат 60×84 1/8. Бумага офсетная. Гарнитура Таймс.
Усл. печ. л. 14,9. Уч.-изд. л. 14,6. Тираж 100 экз. Заказ 19.

Государственное научное учреждение «Объединенный институт проблем информатики Национальной академии наук Беларуси».
ЛИ № 02330/0549421 от 08.04.2009.
Ул. Сурганова, 6, 220012, Минск.

Отпечатано с оригинала-макета на ризографе Объединенного института проблем информатики Национальной академии наук Беларуси.
Ул. Сурганова, 6, 220012, Минск.

THE UNITED INSTITUTE OF INFORMATICS PROBLEMS
OF THE NATIONAL ACADEMY OF SCIENCES OF BELARUS

INFORMATICS

PUBLISHED QUATERLY

Issued since 2004

№ 4(36) • October-December 2012

CONTENTS

CONFERENCES

Посвящается 100-летию со дня рождения выдающегося ученого, крупного организатора науки в Республике Беларусь

Георгия Константиновича Горанского (1912 – 1999)

Георгий Константинович Горанский – первый директор Института технической кибернетики АН БССР, основатель научной школы по автоматизации проектирования в машино- и приборостроении, доктор технических наук, член-корреспондент Академии наук Беларуси, профессор, академик Международной академии наук информации, информационных процессов и технологий (Москва), участник Великой Отечественной войны.

После окончания семилетки в 1928 г. работал разнорабочим на Минской болотной станции и в строительных организациях, после трехмесячных курсов в Центральном институте труда стал каменщиком четвертого разряда, позже овладел специальностью литейщика-формовщика. Одновременно учился на вечернем рабфаке, в 1932 г. перевелся на дневную форму обучения. В 1938 г. с отличием окончил механический факультет БПИ по специальности «Станки, инструменты, механическая обработка металлов».

Окончание БПИ было омрачено ужасной трагедией. Его отец, полковник К.Ф. Горанский, был необоснованно репрессирован и лишь в 1956 г. полностью реабилитирован «за отсутствием состава преступления».

Поработав некоторое время главным механиком Минской щеточной фабрики, Г.К. Горанский перешел на станкостроительный завод им. С.М. Кирова конструктором и по совместительству работал ассистентом в политехническом институте на кафедре «Технология машиностроения». В сентябре 1939 г. был призван в Красную Армию. Будучи командиром огневого взвода артбатареи, участвовал в освобождении Западной Беларуси, советско-финской войне. В октябре 1940 г. возвратился на прежнее место работы – на завод им. С.М. Кирова. После нападения гитлеровской Германии в июне 1941 г. Г.К. Горанский получил назначение комвзвода в прибывшую из Москвы 37-ю мотострелковую дивизию. Участвовал в битве за Москву в составе 16-й армии К.К. Рокоссовского, в конце 1941 г. был сильно контужен и ранен. После излечения попал на Калининский фронт в зенитно-артиллерийский дивизион, начальником штаба которого окончил войну в составе 2-го Прибалтийского фронта, участвовал в освобождении северной части Витебской области.

В августе 1945 г. капитан Г.К. Горанский, награжденный орденами Отечественной войны I ст., Красной Звезды и медалями, вернулся на завод им. С.М. Кирова главным технологом, одновременно преподавал в БПИ. С 1946 г. на постоянной работе в БПИ: декан механического (до 1951) и торфяного факультетов (1953–54), зав. кафедрой, проректор (1954–55). За большую работу по восстановлению и развитию БПИ награжден в 1951 г. грамотой и в 1954 г. Почетной грамотой Верховного Совета БССР. В 1956–57 гг. начальник отдела и главный инженер Специального конструкторского бюро № 8 по проектированию агрегатных станков и автоматических линий Министерства станкостроения СССР.

В 1957 г. Г.К. Горанский был назначен и. о. директора вновь созданного Института машиноведения и автоматизации АН БССР и возглавил лабораторию по автоматизации процессов инженерного и управленческого труда, в которой проводились исследования по алгоритмизации проектирования. Г.К. Горанский приложил много сил для организации и становления института, налаживания тесных контактов с машиностроительными предприятиями.

В период с 1953 по 1960 г. Г.К. Горанский заканчивает серию из шести книг-справочников общим объемом 1466 с., выпущенных Белгосиздатом: «Заточка и доводка инструментов для скоростного резания» (1953); «Высокопроизводительный инструмент: резцы» (1954); «Фасонное точение» (1955); «Резание металлов» (1955); «Высокопроизводительный инструмент: инструмент для обработки отверстий» (1959); «Рациональное использование металлорежущих станков» (1960).

Первые монографии Г.К. Горанского «К теории автоматизации инженерного труда» (1962) и «Расчет режимов резания с помощью ЭВМ» (1963) заложили основу научного направления «автоматизация проектирования в машиностроении на базе средств вычислительной техники».

Г.К. Горанский сформулировал основные понятия о моделях объектов проектирования (машиностроительных конструкциях и деталях, технологических процессах), включая структуру объектов; форму, взаимное положение и размерные связи элементов; количественные и качественные характеристики объектов. Предложил подходы к формализации процессов конструирования, разработал методику кодирования информации о машиностроительных объектах, а также методы использования аппарата теории множеств для формализации и алгоритмизации процессов машиностроительного проектирования.

В апреле 1963 г. Г.К. Горанский вместе с лабораторией переведен в Институт математики и вычислительной техники АН БССР и назначен зам. директора института по вычислительному центру. С 13 февраля 1965 г. назначен и. о. директора Института технической кибернетики АН БССР, позже избран директором. В последующие годы ярко проявился его талант руководителя научного учреждения. Исследования института того периода охватывали весь комплекс проблем, связанных с автоматизацией технической подготовки производства, включая проектирование деталей и узлов машин, станков, технологических процессов и приспособлений, инструментов, штампов и другой оснастки; автоматизацию проектирования схем управления станками и оборудованием. Разрабатывались и создавались различные технические средства ввода, хранения и обработки графической информации и программное обеспечение для них. Под его редакцией издавался сборник «Вычислительная техника в машиностроении» (1965–1970). Институт стал известен не только в СССР, но и за рубежом. Наладилось интенсивное международное сотрудничество с научными организациями и промышленными предприятиями ГДР, Венгрии, Бельгии, Англии, Канады. Г.К. Горанский возглавил делегацию ученых АН СССР на Международном конгрессе по кибернетике в Лондоне в 1969 г. и выступил с докладом. Входил в редакционный совет Белорусской советской энциклопедии.

В 1970 г. совместно с учениками изданы три монографии Г.К. Горанского: «Элементы теории автоматизации машиностроительного проектирования с помощью вычислительной техники» (Минск : Наука и техника. 267 с.), «Автоматизация технического нормирования работ на металлорежущих станках с помощью ЭВМ» (М. : Машиностроение. 224 с.), «Опыт построения эффективных алгоритмов машинного проектирования специальных станочных приспособлений» (Минск : Наука и техника. 120 с.).

В последующие годы были опубликованы монографии «Автоматизированные системы технологической подготовки производства в машиностроении» (М. : Машиностроение, 1976. 240 с.); «Технологическое проектирование в комплексных автоматизированных системах подготовки производства» (М. : Машиностроение. 1981. 456 с.); «Автоматизация проектирования технологических процессов и средств оснащения» (Минск : Ин-т техн. кибернетики НАН Беларуси, 1997. 276 с.). В них Г.К. Горанский последовательно развивал методы решения задач, поставленных в первых двух монографиях.

В 1971 г. Г.К. Горанский организовал и до 1982 г. возглавлял Минское отделение Центрального научно-исследовательского технологического института Миноборонпрома СССР, являясь главным конструктором по системам автоматизации проектирования (САПР) и автоматизированным системам технологической подготовки производства (АС ТПП).

Г.К. Горанским в соавторстве и под его редакцией подготовлено и издано учебное пособие «Основы разработки автоматизированных систем технологической подготовки производства в машиностроении» (Челябинск : ЧПИ, 1977. 373 с.).

В 1985–1993 гг. в НИИ научно-технической информации и технико-экономических исследований Госплана БССР на основе новых результатов опубликовал серию из 11 брошюр

о создании и использовании АС ТПП в машино- и приборостроении. В них собраны и представлены методические указания, руководящие и справочные материалы по разработке и эксплуатации АС ТПП.

С 1993 г. Г.К. Горанский, главный научный сотрудник Института технической кибернетики АН Беларуси, исследует проблемы создания новых информационных технологий для комплексной автоматизации процессов подготовки производства в машиностроении.

Г.К. Горанский – автор многочисленных научных работ, в том числе 8 монографий, 20 брошюр, нескольких изобретений и госстандартов, научно-популярных статей, один из авторов и член редколлегии трехтомника «Прогресс кибернетики» (Лондон : Гордон и Бридж, 1970).

Г.К. Горанский создал научную школу, подготовив 38 кандидатов наук для Беларуси, России, Узбекистана, Литвы, Эстонии и других стран, 10 из них стали докторами наук. В этих странах с помощью Г.К. Горанского созданы и начали плодотворную деятельность научные школы по автоматизации машиностроительного проектирования, а также внедрен ряд разработок в этой области. Он являлся членом технического совета Совнархоза БССР, председателем секции автоматизации инженерного труда научного совета по вычислительной технике ГКНТ СССР, членом комиссии по внедрению вычислительной техники при Госплане БССР, председателем Белорусской территориальной группы Международной федерации по автоматическому управлению, членом научного совета по комплексной проблеме «Кибернетика» при Президиуме АН СССР.

Г.К. Горанский – один из первых советских ученых, определивших основные направления автоматизации машиностроительного проектирования с использованием вычислительной техники и создавших теоретические основы и практические методы решения возникающих при этом проблем.

Сотрудникам института Георгий Константинович запомнился как яркая, колоритная личность: инженер, ученый, педагог, организатор трех научных коллективов, научный редактор, знаток литературы и культуры Беларуси.

А.В. Тузиков,
Е.В. Владимиров,
Н.П. Савик

ОБРАБОТКА СИГНАЛОВ, ИЗОБРАЖЕНИЙ И РЕЧИ

УДК 004.932.2

В.А. Ковалев, В.А. Левчук

ПОИСК СТРУКТУРНЫХ ОСОБЕННОСТЕЙ СТРОЕНИЯ ТОМОГРАФИЧЕСКИХ ИЗОБРАЖЕНИЙ НА ОСНОВЕ КОНЦЕПЦИИ АКТИВНЫХ АГЕНТОВ

Предлагается алгоритм поиска особенностей строения томографических изображений, основанный на концепции самоорганизующихся агентных систем. В реализованной мультиагентной системе поставленная задача решается коллективно за счет конкуренции автономных агентов двух типов. Работа мультиагентного алгоритма демонстрируется на примере задачи поиска закономерностей строения, связанных с туберкулезом легких. Эффективность предложенного алгоритма оценивается на достаточно большой базе данных трехмерных КТ-изображений, включающей томограммы грудной клетки 111 пациентов общим объемом около 10 000 слоев.

Введение

В связи с постоянным ростом количества получаемой цифровой информации возникает задача ее автоматической обработки, анализа и выделения особенностей. В частности, с появлением все новых медицинских баз данных и ростом объема уже существующих появляется потребность в инструментах и методиках, облегчающих работу с такими базами. К данным методикам можно отнести поиск медицинских изображений в базах данных по содержанию [1], задачи автоматической сегментации изображений, поиска и выделения наиболее информативных участков и т. д.

Задача автоматического поиска закономерностей строения биомедицинских изображений также актуальна при разработке новых способов диагностики заболеваний, обнаружения признаков тех или иных характеристик протекания болезни и при прогнозировании. Традиционно такие исследования проводятся при наличии ряда гипотез, которые предполагают существование взаимосвязей между определенными структурами на изображениях с некоторыми медицинскими данными [2, 3]. Будучи разработан и успешно применен к входным данным, подход, основанный на ряде определенных гипотез, позволяет выявить связи лишь с теми структурами на изображениях, которые были заранее выбраны (отсегментированы) для последующей обработки и анализа, при этом часть полезной информации может остаться за пределами исследования.

В данной работе предлагается методика поиска закономерностей, т. е. связей между структурами на изображениях с клиническими показателями в отсутствие какой-либо гипотезы или априорных данных об исследуемых структурах. Ограничимся лишь предположением, что искомые объекты некоторым образом локализованы на изображениях. Следует подчеркнуть, что исследование связей между морфологическими признаками изображений и медицинскими данными по пациентам, представленное в работе [4], проводилось при отсутствии гипотезы об этих связях. Однако в [4] постановка задачи значительно отличалась от рассматриваемой, поскольку в силу специфики исследуемых в ней гистологических изображений предположение о локальности искомых структур не имело места.

Алгоритм решения поставленной задачи был реализован на основе достаточно новой концепции активных агентов и может быть отнесен к семейству так называемых «биологически навеянных» алгоритмов (biologically-inspired algorithms) [5] или алгоритмов, основанных на концепции стайного интеллекта (swarm intelligence). Такой подход позволяет задействовать некоторые преимущества самоорганизующихся систем, включая самообучение, самоорганизацию, перераспределение вычислительных мощностей и т. д. Концепция самоорганизующихся

систем на данный момент является относительно молодой и недостаточно изученной областью информатики. Поэтому одной из целей настоящей работы является изучение потенциала самоорганизующихся систем применительно к задаче поиска особенностей строения изображений.

Отметим, что предлагаемый алгоритм не ориентирован на конкретный тип изображений и может быть применен для нахождения особенностей в данных различного рода. Отличительной чертой рассматриваемой задачи поиска является то, что предмет поиска изначально неизвестен и ищется не что-то похожее на образец (классическая задача [1, 6]), а, напротив, – нечто не похожее ни на что из противоположного класса.

1. Концепция активных агентов

Активные (автономные) агенты – довольно молодая область исследований в обработке изображений, возможности которой в настоящее время в достаточной мере не изучены. Однако идея применения самоорганизующихся систем к решению различного рода задач обработки изображений в последние годы приобретает все большую популярность.

Вычисление при помощи активных агентов можно отнести к области стайного интеллекта, в основе которого лежит идея отсутствия какого-либо центрального механизма управления всей системой. Система в этом случае состоит из большого числа самостоятельных единиц (подсистем, агентов), способных получать информацию об окружающей среде, перемещаться в данной среде и взаимодействовать между собой. При этом каждая подсистема способна управлять только своим поведением, однако выбор конкретного поведения зависит от состояния и поведения окружающих агентов. Во время работы система самоорганизуется благодаря заложенным в ней вероятностным механизмам и механизмам взаимодействия, которые прямо или косвенно определяют конечное равновесное состояние. Примером таких систем могут служить системы, реализующие так называемый муравьиный алгоритм (ant colony optimization), который применяется для решения некоторых задач теории графов [7]. При использовании этого алгоритма для получения сведений об изучаемой системе анализируется поведение не отдельной особи стаи, а стаи как целого, т. е. анализируются кумулятивные данные.

Прежде всего необходимо дать определение термину «агент». В работе [8] предлагается следующее определение: *агент* – это программный модуль, которому присущи следующие характеристики:

– *цикл агента* (исследование окружения агента – анализ – планирование дальнейших шагов – выполнение – исследование окружения...), определяющий его базовое функционирование;

– индивидуальный *запас знаний*, содержащий знания агента о других агентах и своем окружении в явном виде;

– исчерпывающие возможности *обмена информацией* (различные протоколы обмена данными, вещание, голосование), которые дают возможность агентам решать задачи кооперативно;

– способность агента интерпретировать и изменять свою *роль* и моделировать как свою роль, так и роль других агентов.

Таким образом, *мультиагентная система* – это сообщество агентов, коллективно решающих поставленную задачу. Однако такое определение агента не является жестким. Для решения конкретной задачи некоторые из перечисленных характеристик могут быть опущены, в то время как дополнительные особенности, например такие, как *жизненный цикл*, могут быть добавлены.

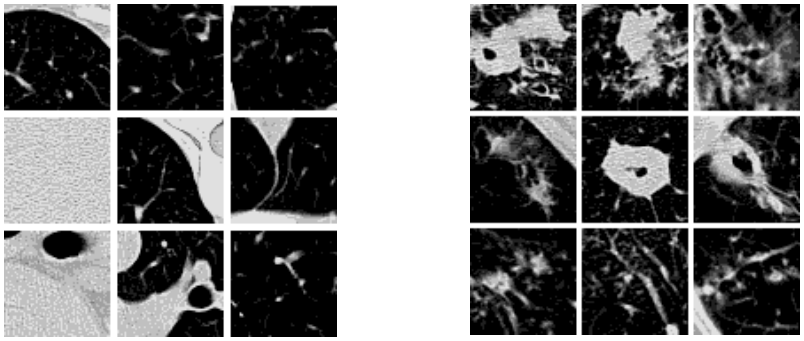
В ряде работ для решения разнообразных задач используется данное выше определение агента с некоторыми упрощениями или дополнительными особенностями. В работе [9] агенты использовались для детектирования лесных пожаров. Это были агенты одного типа, способные в случае нахождения «интересных» участков плодить себе подобных. В работе [10] для нахождения границ на слоях КТ-изображений использовались агенты нескольких типов. В [11] на основе агентного подхода решалась задача сегментации МРТ-изображений мозга. В работе [12] использовались агенты одного типа с полноценным жизненным циклом: агенты, нашедшие цель, успешно размножались (рождались новые), а не нашедшие цель в течение длительного времени исключались из системы (умирали). Применение такого жизненного цикла также по-

зволяет более разумно распределять машинные ресурсы, увеличивая приоритет более интересных участков на изображениях по сравнению с менее интересными.

2. Понятие структурных особенностей

Поскольку описываемый алгоритм ориентирован на поиск «особенных» структур на изображениях, данное понятие необходимо определить. Предположим, что имеются изображения одного класса (томографические изображения одной модальности каких-либо частей тела здоровых людей или гистологические изображения здоровых тканей определенного органа), для которых характерно присутствие одних и тех же структур на изображениях одного класса. Такие изображения можно назвать *базовыми*. Пусть разнообразие присутствующих на изображениях структур не слишком велико и изображения в рамках одного класса с первого взгляда кажутся вовсе неразличающимися. Тогда множество структур, присутствующих на изображениях рассматриваемого класса, можно назвать множеством *обыкновенных* структур (рис. 1, *а*), характерных для данного класса изображений.

Далее рассмотрим изображения, принадлежащие к смежным классам, например томографические изображения тех частей тела больных, у которых признаки той или иной болезни проявляются на изображениях, и гистологические изображения раковой ткани. Для таких «особенных» изображений характерно присутствие как обыкновенных структур, так и структур, не встречающихся на базовых изображениях (структур, являющихся признаками заболевания). Именно такие структуры, не похожие ни на что из заданного множества, будем называть *особенными* (рис. 1, *б*), и они будут являться предметом поиска в рамках поставленной задачи. Очевидно, что предмет поиска определяется набором обыкновенных структур, поэтому цель поиска в предлагаемом алгоритме будет определяться автоматически в самом процессе поиска.

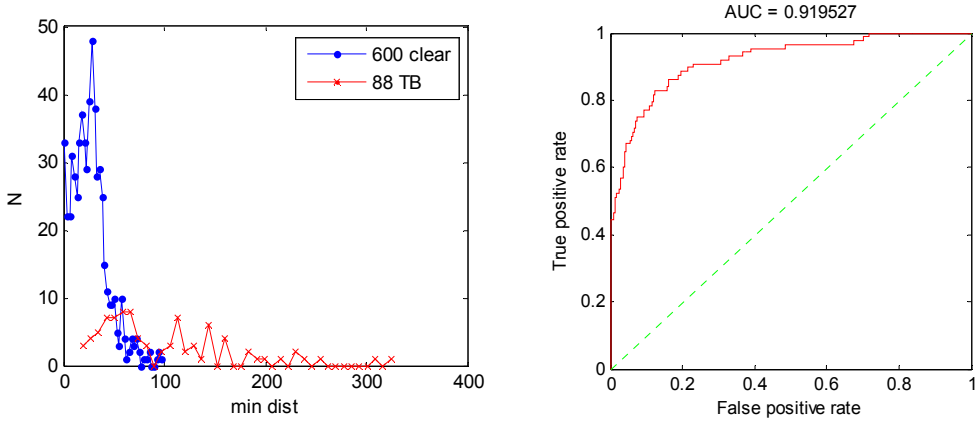


а) *б*)

Рис. 1. Примеры слоев тестовых участков изображений:
а) обыкновенные участки (здоровые); *б*) особенные участки (пораженные туберкулезом)

Приведем способ численного описания степени особенности той или иной структуры. Пусть в наличии имеется набор обыкновенных структур, содержащий значительное количество образцов, и набор потенциально особенных структур. Для каждой структуры может быть вычислен какой-либо вектор признаков (дескриптор), численно характеризующий рассматриваемую структуру. Согласно полученным векторам признаков может быть оценена степень несхожести (расстояние в пространстве признаков) между любой парой структур. Тогда для каждой структуры может быть вычислена следующая величина: *минимальное* расстояние от дескриптора исследуемой структуры до какого-либо из дескрипторов набора обыкновенных структур. Именно эту величину можно использовать в качестве численной характеристики степени особенности структуры, так как для обыкновенных структур она оказывается закономерно меньшей, чем для особенных (рис. 2, *а*). На рис. 2, *б* показана кривая ошибок бинарной классификации дескрипторов участков КТ-изображений кубической формы на обыкновенные и особенные. Основываясь на предположении, что описанные выше минимальные расстояния для обыкновенных структур приближенно подчиняются нормальному распределению, можно вве-

сти порог для минимального расстояния, выше которого исследуемая структура считается особенной, пользуясь правилом «трех сигм».



а) *б)*

Рис. 2. Разделение структур на обыкновенные и особенные: *а)* гистограммы распределения минимальных расстояний до обыкновенных дескрипторов для 600 здоровых и 88 туберкулезных участков; *б)* кривая ошибок классификации

Очевидно, что основные структуры, представленные на рассматриваемом классе изображений, обладают текстурными свойствами. Для описания текстурных свойств были выбраны многомерные матрицы совместной встречаемости типа IID, являющиеся обобщением матриц смежных вероятностей [13] и подробно описанные в [14]. Дескрипторы такого типа хорошо подходят для описания текстурных свойств как 2D-, так и 3D-изображений, так как обладают многими полезными свойствами (устойчивостью к шумам, инвариантностью к поворотам, отражениям и др.) и доказали свою эффективность в целом ряде работ (см., например, [5, 14, 15]).

3. Алгоритм поиска

Поскольку в поставленной задаче понятие особенных структур как предмета поиска напрямую основывается на понятии обыкновенных структур, в рамках данной работы были использованы два типа агентов, ответственных за обыкновенные и особенные структуры. Агенты одного типа – назовем их *ищущими* (seeking) – помещаются на исследуемые изображения, потенциально содержащие текстурные аномалии, и занимаются поиском особенных структур. В процессе поиска агенты этого типа используют информацию об обыкновенных структурах, которую добывают агенты другого типа – *проверяющие* (checking), помещенные на соответствующие базовые изображения, заведомо не содержащие аномалий.

Задача ищущих агентов состоит в нахождении участков на изображениях, у которых минимальное расстояние от их дескриптора до дескриптора любого участка на базовых изображениях достаточно велико. Задача проверяющих агентов состоит в возможном нахождении на своих изображениях участков, похожих на найденные ищущими. Если же для некого ищущего агента найти такой похожий участок на базовых изображениях не удается, то можно считать, что ищущий агент нашел некую особенность, не характерную для базовых изображений. Таким образом, поставленная задача решается при помощи мультиагентной системы, работающей на основе конкуренции двух типов агентов (рис. 3).

Во время работы алгоритма агенты передвигаются по изображению, исследуя его. Результат исследования для каждого агента характеризуется соответствующей целевой функцией (cost function), определяющейся по-разному в зависимости от типа агента. Для ищущих агентов целевая функция есть минимальное расстояние от его дескриптора до дескрипторов проверяющих агентов, присутствующих на других изображениях. При этом ищущий агент стремится максимизировать значение своей целевой функции. Для проверяющих агентов это минимальное расстояние до дескриптора какого-либо из ищущих агентов, которое агент стремится минимизировать. Постоянное движение агентов вдоль направления градиента целевой функции может привести к

их застреванию в множестве локальных минимумов (максимумов). Поэтому для целевой функции вводится некий порог, выше которого движение агента происходит случайным образом, а ниже – вдоль градиента целевой функции. Таким образом идет поиск наиболее выразительных локальных минимумов (максимумов) целевой функции.

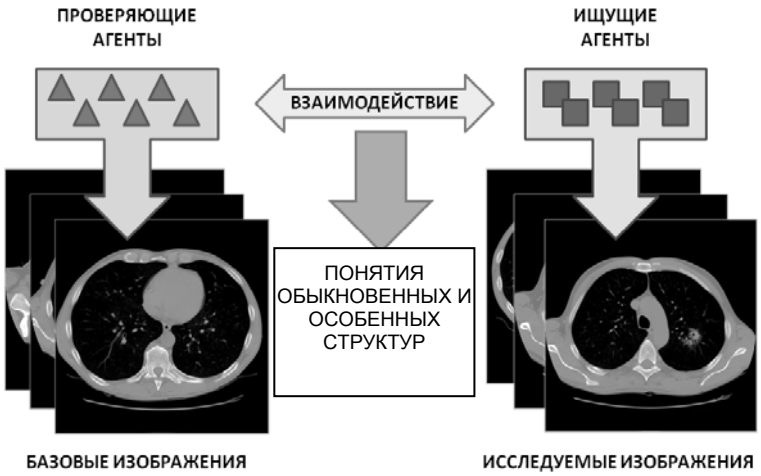


Рис. 3. Схематическое изображение мультиагентной системы

Для более экономного использования временных и вычислительных ресурсов предусматривается контроль над жизнью агентов. Поскольку при помощи проверяющих агентов определяется понятие обыкновенных структур, их количество на протяжении всей работы алгоритма должно быть значительным. В то же время для ищущих агентов ставятся следующие условия. Если на протяжении определенного числа итераций целевая функция агента не достигла порогового значения, агенты исключаются из системы (умирают). Если же на протяжении нескольких итераций целевая функция ищущего агента все время превышала заданный порог (агент нашел особенность и находка не опровергнута), агент производит себе подобного (размножается). Таким образом, в «неинтересных» местах агенты постепенно погибают, а в более «интересных» их количество увеличивается до тех пор, пока все особенные структуры не будут заняты агентами.

Описываемый алгоритм можно схематически представить следующим образом:

1. Загрузка изображений
2. Помещение агентов на изображения, вычисление дескрипторов агентов
3. Обнуление счетчиков
4. **Для** каждой итерации повторять:
5. Вычисление стандартных отклонений элементов дескрипторов
6. Вычисление пороговых значений целевой функции
7. **Для** каждого агента выполнить:
8. Вычисление значений целевой функции в соседних положениях
9. **Если** хотя бы одно из вычисленных значений лучше порогового
10. Смещение в направлении улучшения целевой функции
11. Инкрементация счетчика целевых шагов
12. **Иначе**
13. Смещение в случайном направлении
14. Инкрементация счетчиков случайных шагов
15. **Конец Если**
16. **Если** тип агента ищущий
17. **Если** значение счетчиков случайных шагов больше порога
18. Смерть агента
19. **Если** значение счетчиков целевых шагов больше порога
20. Размножение агента
21. **Конец Если**

22. **Конец Для**
23. **Конец Для**
24. Анализ конечных положений агентов

Данный алгоритм несколько схож с алгоритмами, приведенными в [9–12]. Одной из его отличительных черт является то, что работа алгоритма основана не на кооперации, а на конкуренции, так как агенты каждого типа стремятся улучшить значение своей целевой функции, тем самым ухудшая соответствующие значения агентов противоположного типа.

4. Анализ изображений

В данной работе использовалась база, содержащая КТ-изображения 111 пациентов, больных туберкулезом легких, средний возраст которых составлял 45 лет. Изображения были получены на четырех различных видах аппаратов: GE MEDICAL SYSTEMS LightSpeed Pro 16, GE MEDICAL SYSTEMS LightSpeed Pro 32, SEMENS Emotion и SIEMENS Emotion 6, что несколько затрудняло работу с полученными изображениями. Данные о состоянии пациентов включали в себя множество характеристик, таких как пол, возраст, рост, вес, диагноз, сопутствующие заболевания. Для 97 пациентов были известны данные о лекарственной устойчивости (ЛУ) туберкулеза – важной характеристике заболевания, поиск признаков которой на изображениях в настоящее время является одной из приоритетных задач медицины.

4.1. Демонстрация работы алгоритма на примере задачи поиска различий двух изображений

Для начала рассмотрим следующий демонстрационный вычислительный эксперимент. Возьмем изображение пациента, у которого лишь одно легкое содержит явные признаки заболевания. На вход алгоритма подадим два изображения: легкое без признаков болезни и больное легкое. В данном случае каждое из изображений является само по себе исследуемым и в то же время эталонным для другого изображения. Тогда задачей алгоритма будет являться нахождение различий в строении этих двух изображений.

Для демонстрационного эксперимента использовались следующие параметры: размер агента (области, по которой вычисляется дескриптор) A = 10 мм, длина шага агента $Step$ = = 11 мм, начальное расстояние между агентами $Dist$ = 30 мм, пороговые значения для счетчиков случайных и целевых шагов $maxRand = toBirth$ = 30.

Рассмотрим КТ-срезы исследуемых изображений и положения агентов на них на разных стадиях работы алгоритма (рис. 4). Квадратами синего и голубого цвета обозначены размеры и позиции проверяющих агентов, красного и желтого цвета – ищущих. Тонкими квадратами изображены агенты, движущиеся в процессе поиска оптимальных условий случайным образом; толстыми квадратами – агенты, движущиеся вдоль направления улучшения целевой функции. Тот факт, что задача решается в объеме, делает затруднительным задачу наглядной и информативной визуализации состояния процесса. Поэтому в данном случае ограничимся изображениями наиболее информативных плоских слоев изображений.

На рис. 4, *а – г* показаны состояния мультиагентной системы на начальных итерациях. Для этой стадии характерно присутствие на изображениях большого числа агентов обоих типов. Большинство ищущих агентов находится в состоянии случайного поиска, и лишь небольшая их часть нашла участки, претендующие называться особенными. Проверяющие же агенты стремятся найти на своих изображениях структуры, похожие на структуры, найденные достаточно большим множеством ищущих агентов на противоположном изображении.

На рис. 4, *д – з* изображены состояния мультиагентной системы после большего количества итераций, чем максимально допустимое количество случайных шагов, т. е. все ищущие агенты, не нашедшие потенциально особенных участков, которых большинство, исключаются из системы. В это же время агенты, которые в течение длительного времени пребывали в состоянии градиентного поиска, начинают плодить себе подобных. Вместе со значительным уменьшением количества ищущих агентов уменьшается разнообразие предметов поиска для проверяющих агентов.

На рис. 4, *и – м* изображены финальные состояния системы после завершения 100 итераций (показаны только ищущие агенты). Конечное расположение агентов говорит о найденных особенных структурах. Видим, что алгоритму удалось определить яркое узловое образование в легком, характерное для туберкулеза, как особенную структуру. Этот объект на изображении выделен четырьмя близко расположенными агентами. Помимо искомого туберкулезного образования алгоритм выделил в качестве особенности и другие участки. Среди них есть как артефакты, обусловленные биением сердца во время сканирования пациента (рис. 4, *л*), так и обычные структуры (рис. 4, *к*), оказавшиеся доступными лишь на одной из частей изображения из-за небольшого смещения плоскости симметрии легких относительно центра изображения. Такого рода ложные особенности могут впоследствии быть отсеяны посредством различного рода процедур постобработки.

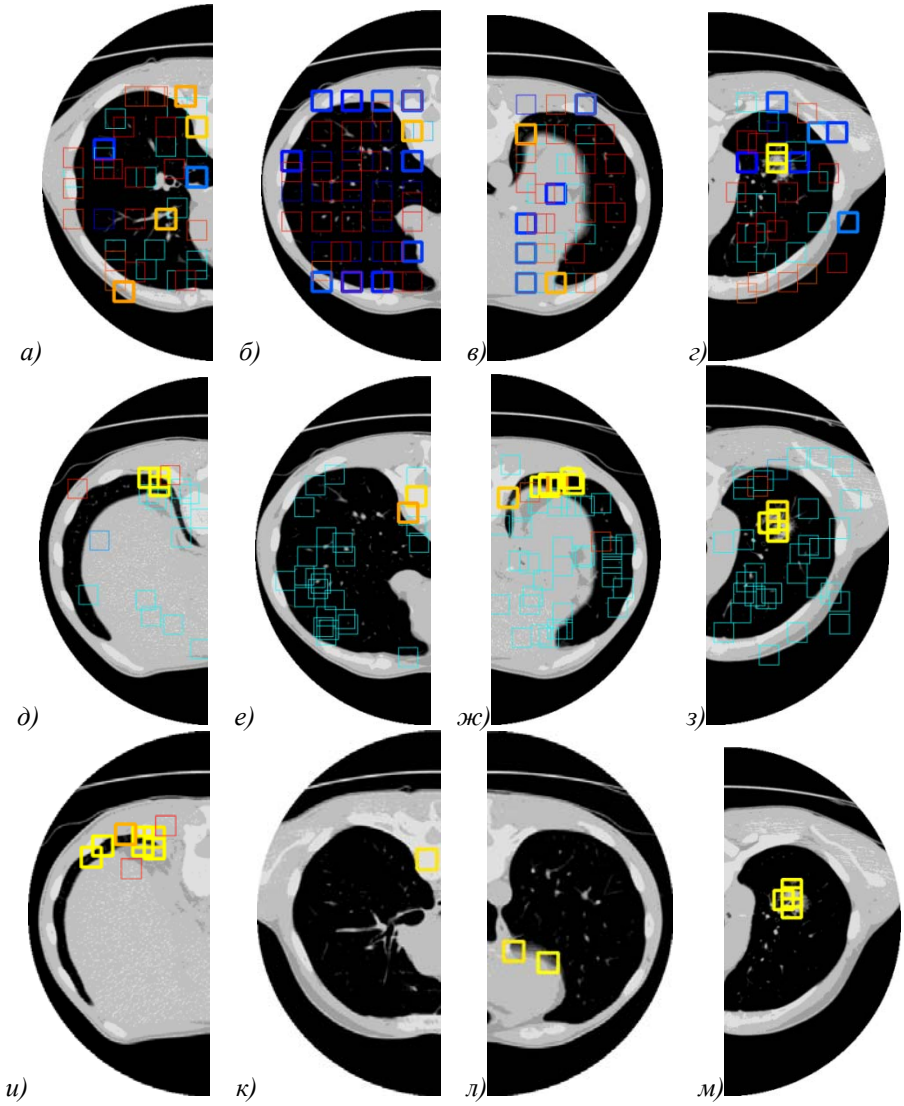


Рис. 4. Срезы исследуемых КТ-изображений с представленными на них агентами на разных стадиях работы алгоритма (две первые колонки – изображения без признаков болезни, две последние колонки – изображения больного легкого; оттенками красного показаны ищущие агенты, оттенками синего – проверяющие): *а) – г)* 4 итерации; *д) – з)* 37 итераций; *и) – м)* 100 итераций

4.2. Поиск особенностей строения, связанных с туберкулезом

Рассмотрим работу алгоритма применительно к достаточно большой базе данных КТ-изображений пациентов, больных туберкулезом. В этом случае можно поставить задачу автоматиче-

ского поиска особенностей строения изображений базы данных, причем роль особенностей будут играть структуры на изображениях, являющиеся признаками заболевания (в том числе туберкулеза). Задача автоматического выделения очагов туберкулеза не представляет прямого интереса с научной точки зрения и скорее является тестовой задачей для исследуемого алгоритма. Научный интерес в данном случае состоит в проверке возможностей самоорганизующихся систем применительно к задачам такого рода. Однако поставленная задача имеет интерес с практической точки зрения, а именно с точки зрения систем, работающих с базами данных значительных объемов.

Для описываемого вычислительного эксперимента в качестве эталонных изображений были использованы предварительно отсегментированные изображения частей легких 13 пациентов из базы данных, в которых не содержалось признаков заболеваний. Применялись следующие параметры: размер агента A = 13 мм, длина шага *Step* = 14 мм, начальное расстояние *Dist* = 40 мм, *maxRand* = *toBirth* = 40. Максимальное число итераций составляло 150. Исследованию подвергались все 111 изображений из базы данных. Из имеющихся изображений шесть оказались непригодными для анализа (другой яркостной диапазон, плохое качество изображений). В результате работы алгоритма на 85 из 105 пригодных для анализа изображений (81 %) очаги туберкулеза были выделены в качестве особенностей. При этом на семи изображениях присутствовали только слабо выраженные признаки туберкулеза (см. рис. 4, *б*), которые в результате не были выделены. Также из 105 изображений на 71 (68 %) в качестве особенностей были выделены другие структуры: бронхи, артефакты реконструкции, артефакты, связанные с дыханием пациентов и сердцебиением во время снимка (см. рис. 4, *и*, *л*).

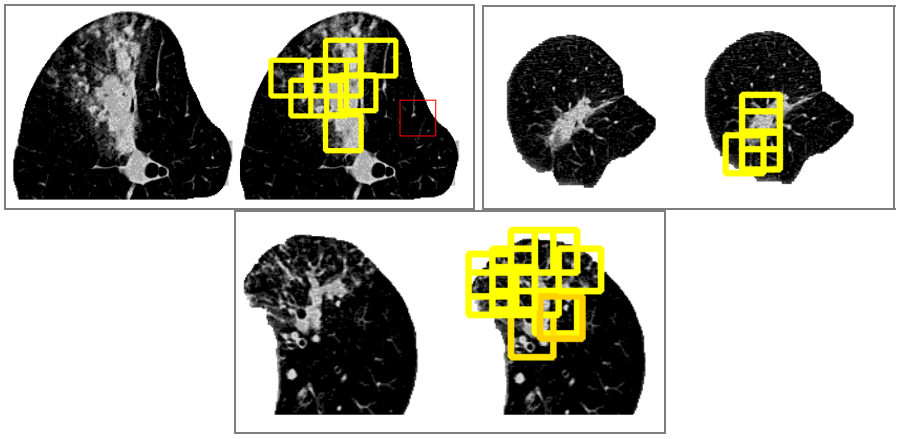


Рис. 5. Примеры автоматического выделения областей легкого, пораженных туберкулезом, при помощи агентов

Полученные результаты можно сравнить с результатами автоматического выделения очагов туберкулеза на тех же изображениях, произведенного при помощи алгоритма, работающего прямым перебором, в котором решение о наличии признака туберкулеза в структуре принималось на основе вычисления того же рода дескрипторов, применения метода главных компонент и обучения на выборке данных (таблица). В результате работы такого «прямого» алгоритма очаги туберкулеза были выделены на 73 изображениях из 105 (70 %), другие структуры – на 70 изображениях из 105 (67 %). Видно, что алгоритм, основанный на мультиагентной системе с применением дескриптора типа IID, дал не худшие результаты по сравнению с алгоритмом, основанным на обучении. Отметим, что условия работы этих алгоритмов были принципиально разными: для обучающегося алгоритма нужно было найти нечто, похожее на заданные образцы (предмет поиска явно задан, присутствует обучение), для мультиагентного подхода – нечто, не встречающееся на заданных изображениях (самообучение).

Несмотря на наличие значительного количества ошибок в выходных данных, результат работы алгоритма можно назвать удовлетворительным, так как предмет поиска ввиду специфики задачи не был предварительно определен. Ошибочные результаты в выходных данных могут быть отсеяны с применением несложных процедур постобработки (более детального анализа и классификации выходных данных).

Результаты работы двух методик

Метод	Выделен туберкулез, кол-во случаев, %	Выделены другие структуры, кол-во случаев, %
На основе мульти-агентной системы	85 (81 %)	71 (68 %)
На основе обучения	73 (70 %)	70 (67 %)

Отметим также, что быстрое решение поставленной задачи без применения мультиагентной системы при помощи метода прямого перебора не представляется осуществимым либо технически затруднено. Использованные для решения вышеописанной задачи мультиагентные системы включали в себя порядка 4000 агентов, которые для эффективной работы алгоритма вычислялись и хранились в памяти матрицы расстояний размером 4000×4000. При решении той же задачи методом перебора количество участков изображений, подлежащих рассмотрению, будет порядка 100 000, что значительно превышает количество агентов. В этом случае возникает необходимость вычислять и хранить в памяти матрицу расстояний размером $100\,000 \times 100\,000$, содержащую 10^{10} элементов, что делает решение задачи методом перебора практически неосуществимым на обычных ПК.

4.3. Поиск особенностей строения, связанных со степенью лекарственной устойчивости

В настоящее время научный интерес представляет задача поиска на изображениях признаков ЛУ у больных туберкулезом. ЛУ определяет степень восприимчивости заболевания к определенным группам лекарств и возможность дальнейшего лечения. Анализ на определение степени ЛУ производится при помощи биомедицинских тестов, результаты которых готовятся в течение месяца. В то же время сейчас осуществляется поиск быстрых методов определения степени ЛУ, в том числе исследуется возможность выявления степени ЛУ по КТ-изображениям легких.

Мультиагентная система использовалась для решения задачи поиска различий между двумя классами изображений: изображений пациентов с низкой степенью ЛУ (0) и пациентов с высокой степенью ЛУ (3). Для этой задачи были сформированы две выборки изображений пациентов с соответствующими степенями ЛУ. Каждая выборка содержала по 20 изображений, полученных на одинаковых медицинских аппаратах при одинаковых параметрах. В экспериментах по нахождению различий между двумя наборами изображений поочередно один из наборов считался базовым, а другой – исследуемым. Для рассматриваемого эксперимента использовались следующие параметры: размер агента A = 12 мм, длина шага *Step* = 13 мм, начальное расстояние *Dist* = 39 мм, *maxRand* = *toBirth* = 40. Максимальное число итераций составляло 120. Двухсторонний поиск закономерностей не дал статистически значимых результатов. В результате работы алгоритма на изображениях обоих классов были выделены различного рода артефакты; области легкого, пораженные туберкулезом; бронхи. В единичных случаях на изображениях пациентов с высокой степенью ЛУ были выделены крупные узловые образования.

Отметим, что исследование связи количества узловых и других образований в легких процентов с ЛУ проводилось в работе [16]. В ходе исследования были обнаружены статистически значимые различия в количестве узловых образований в легких пациентов с множественной ЛУ и пациентов с лекарственно чувствительным туберкулезом. Таким образом, результаты работы описываемого алгоритма дают повод задуматься о возможности обнаружения признаков ЛУ среди характеристик узловых образований, имеющихся в легких.

Заключение

В работе сделана попытка изучить потенциал самоорганизующихся систем – достаточно молодой и не в полной мере развитой концепции информатики – применительно к задачам выявления скрытых закономерностей. Был разработан алгоритм поиска особенностей строения

изображений, в котором предмет поиска заранее не специфицирован, а определяется в процессе работы алгоритма.

Разработанный алгоритм показал свою эффективность при обработке имеющейся базы данных изображений и может быть применен к обработке объемных баз данных. Результаты обработки исследуемым алгоритмом оказались не хуже, чем в случае алгоритма, основанного на обучении. При этом условия работы этих двух алгоритмов принципиально разные.

При решении задачи поиска закономерностей строения изображений легких, связанных со степенью ЛУ, алгоритм не дал статистически значимых результатов.

Работа выполнена в рамках проекта BOBI-31055-MK-11, финансируемого фондом CRDF.

Список литературы

1. Kovalev, V.A. Retrieving 3D MRI brain images / V.A. Kovalev, F. Kruggel // Information Retrieval and Exploration in Large Medical Image Collections (VI-SIM-01/MICCAI-01). – Utrecht. The Netherlands, 2001. – P. 53–56.

2. Computer-aided image processing of angiogenic histological samples in ovarian cancer / M. Sprindzuk [et al.] // Journal of Clinical Medicine Research. – 2009. – Vol. 1, No. 5. – P. 249–261.

3. Ковалев, В.А. Картирование характеристик сверхбольших гистологических изображений раковой ткани / В.А. Ковалев, В.А. Левчук // Информатика. – 2012. – № 1(33). – С. 12–17.

4. A Method for Identification and Visualization of Histological Image Structures Relevant to the Cancer Patient Conditions / V. Kovalev [et al.] // Computer Analysis of Images and Patterns (CAIP-2011), LNCS. – Springer Verlag, 2011. – Vol. 6854(1). – P. 460–468.

5. A Biologically Inspired Algorithm for Microcalcification Cluster Detection / M.G. Linguraru [et al.] // Medical Image Analysis. – 2006. – Vol. 10, No 6. – P. 850–862.

6. Chang, P. Object recognition with color cooccurrence histograms / P. Chang, J. Krumm // CVPR'99. – Fort Collins. USA, 1999. – P. 498–504.

7. Colorni, A. Distributed Optimization by Ant Colonies / A. Colorni, M. Dorigo, V. Maniezzo // Actes de la première conférence européenne sur la vie artificielle. – Paris, France : Elsevier Publishing, 1911. – P. 134–142.

8. Luckenhaus, M. A multi-agent based system for parallel image processing / M. Luckenhaus, W. Eckstein // SPIE Optical Sciences, Instrumentation Parallel and Distributed Methods for Image Processing I. – San Diego, CA, USA, 1997. – Vol. 3166. – P. 21–30.

9. Movaghati, S. An agent-based algorithm for forest fire detection / S. Movaghati, F. Samadzadegan, A. Azizi // ISPRS Congress Beijing. – 2008. – Vol. XXXVII, Part. B7. – P. 631–634.

10. Mahdjoub, J. A Multi-agent Approach for the Edge Detection in Image Processings / J. Mahdjoub [et al.] // Fourth European Workshop on Multi-Agent Systems (EUMAS'06). – Lisbon, Portugal, 2006.

11. Richard, N. Multi-agent Approach for Image Processing: A Case Study for MRI Human Brain Scans Interpretation / N. Richard, M. Dojat, C. Garbay // AIME. – Protaras. Cyprus, 2003. – P. 91–100.

12. A swarm-based system for object recognition / T. Mirzayans [et al.] // Neural Network World. – 2005. – Vol. 15. – P. 243–255.

13. Харалик, Р.М. Статистический и текстурный подход к описанию текстур / Р.М. Харалик // ТИИЭР. – 1979. – № 5. – С. 98–120.

14. Ковалев, В.А. Анализ текстуры трехмерных медицинских изображений / В.А. Ковалев. – Минск : Белорусская наука, 2008. – 263 с.

15. Ковалев, В.А. Влияние мер близости в пространстве признаков на качество поиска медицинских изображений по содержанию / В.А. Ковалев, А.А. Дмитрук // Информатика. – 2011. – № 2. – С. 5–11.

16. Radiological Findings of Extensively Drug-Resistant Pulmonary Tuberculosis in Non-AIDS Adults: Comparisons with Findings of Multidrug-Resistant and Drug-Sensitive Tuberculosis / J. Cha [et al.] // Korean Journal of Radiology. – 2009. – Vol. 10(3). – P. 207–216.

Поступила 12.07.2012

Объединенный институт проблем информатики НАН Беларуси, Минск, Сурганова, 6
e-mail: vassili.kovalev@gmail.com
vitali.liauchuk@gmail.com

V.A. Kovalev, V.A. Liauchuk

SEARCHING FOR STRUCTURAL DIFFERENCES IN TOMOGRAFHY IMAGE STRUCTURE BASED ON THE ACTIVE AGENTS APPROACH

An algorithm is proposed for searching structural differences in tomography image structure which is based on the active agents approach. The searching task is implemented using a technique of competition of two agent types, the searching agents and the seeking ones. Functioning of the multi-agent algorithm is illustrated on the problem of searching morphological regularities in CT images structure of lungs of patients suffering from tuberculosis. The efficiency is assessed on a large CT image database which includes results of CT examination of 111 patients with a total of about 10 000 axial image slices of 512x512 pixels in size.

УДК 621.391

Б.М. Лобанов, В.А. Житко

ПРИМЕНЕНИЕ ОБЛАЧНЫХ ТЕХНОЛОГИЙ ПРИ РАСПОЗНАВАНИИ РЕЧИ

Рассматриваются особенности применения облачных интернет-технологий в задачах автоматического распознавания речи. Кратко описываются положительные особенности и основные модели построения современных облачных интернет-технологий. Рассматриваются существующие методы, технологии и архитектура систем автоматического распознавания речи, а также особенности облачной технологии распознавания речи, предоставляемой компанией Google. Описывается разработанная на основе этой технологии экспериментальная прикладная программа STENOGRAPH, а также приводятся предварительные результаты ее тестирования.

Введение

В связи с бурным ростом рынка интернет-поиска с использованием мобильных устройств разработкам в сфере упрощения ввода текстового запроса информации уделяется большое внимание. Наиболее перспективными в этой ситуации являются разработки, связанные с распознаванием речи, – современная альтернатива ручному вводу с клавиатуры. Однако в этом направлении существует масса различных проблем:

– большая вычислительная сложность и огромный объем баз данных, необходимых для корректного решения задачи распознавания речи;

– большое разнообразие используемых мобильных устройств и интернет-платформ.

Одним из возможных путей преодоления указанных трудностей является перенос на удаленный сервер системы распознавания речи. В связи с этим наиболее перспективным выглядит применение активно развивающихся в настоящее время облачных интернет-технологий, позволяющих легко наращивать по мере необходимости производительность и объем используемых баз данных.

1. Современные облачные интернет-технологии

Облачные технологии – это парадигма, предполагающая распределенную и удаленную обработку и хранение данных. Применение облачных технологий позволяет организовать повсеместный и удобный удаленный доступ к общему пулу вычислительных ресурсов, таких как устройства хранения данных, сети передачи данных, а также различные сервисы и приложения [1]. Одним из преимуществ данного подхода является снижение требований к вычислительным мощностям на стороне клиента при решении сложных вычислительных задач. Это позволяет интегрировать различные ресурсоемкие сервисы, например распознавание устной речи, в устройства с ограниченными ресурсами (мобильные телефоны, планшеты и др.). Облачные технологии имеют ряд положительных особенностей [2]:

1. Самообслуживание по требованию (on-demand self-service) – обеспечивает пользователю возможность самому определять вычислительные мощности конкретного облака.

2. Универсальный сетевой доступ (broad network access) – помогает организовывать однообразный доступ к облаку вне зависимости от используемого терминала.

3. Объединение ресурсов в пулы (resource pooling) – позволяет динамически изменять вычислительные мощности облака в зависимости от нагрузки.

4. Эластичность (rapid elasticity) позволяет предоставлять, расширять, сужать и убирать услуги в любой момент времени без дополнительных издержек взаимодействия с поставщиком.

5. Непрерывный учет потребления (measured service) – дает возможность исчислять объем услуг, предоставленный пользователю, в фактически потребленных ресурсах (объеме памяти, пропускной способности, количестве транзакций и пр.).

Перечисленные особенности делают облачные технологии особенно удобными для использования в развивающихся проектах, когда требования к вычислительным мощностям и базам данных постоянно изменяются.

Облачные сервисы могут предоставляться пользователю по трем основным моделям [2]:

программное обеспечение как услуга (software as a service) – готовая платформа с уже установленным программным обеспечением. В этом случае пользователю нет необходимости покупать программное обеспечение самому, а достаточно лишь использовать то, которое предоставляет облачный сервис;

платформа как услуга (platform as a service) – облачная платформа, а также возможность разворачивать на ней любые приложения;

инфраструктура как услуга (infrastructure as a service) – некоторая облачная инфраструктура (виртуальные серверы), а также возможность конфигурировать ее под нужды пользователя.

Данные модели сервисов позволяют использовать облачные технологии в широком спектре проектов, начиная от небольших интернет-порталов до корпоративных сетей с множеством сервисов и клиентов с широкой географией.

Один из наиболее развитых сервисов технологии облачных вычислений предоставляется компанией Google [3]. Компания Google развивала технологию облачных вычислений более 10 лет. Многопользовательская инфраструктура Google позволяет сократить время развертывания приложений и их обновлений для всех пользователей. Быстрое внедрение программных решений с использованием облачных технологий Google обладает преимуществом по сравнению с другими системами. Веб-приложения на базе облачных технологий Google предоставляют пользователям доступ к облачным приложениям и хранимой информации с различных устройств, что повышает удобство и мобильность пользователя. Хранение данных в облаке, а не на компьютерах пользователей позволяет нескольким пользователям обращаться к информации и работать с нею одновременно, не беспокоясь о совместимости операционных систем, программного обеспечения и браузеров. Синхронная репликация позволяет синхронизировать данные и действия пользователя в режиме реального времени между несколькими центрами обработки данных. Если один из них по какой-либо причине становится недоступен, то система мгновенно обращается к резервному центру, при этом пользователь не испытывает никаких перебоев в обслуживании.

2. Облачная технология автоматического распознавания речи

С развитием интернет-технологий, а также коммуникационных возможностей мобильных устройств все более актуальной становится разработка разнообразных интернет-сервисов. Одним из многообещающих сервисов являются системы обработки и распознавания человеческой речи. Современные системы автоматического распознавания речи (АРР) представлены двумя основными технологиями, реализующими распознавание голосовых команд (Voice Command Recognition, VCR) и распознавание слитной речи (Speech-to-Text, STT).

Исторически VCR-технология возникла раньше STT-технологии, и уже в 1990-х гг. VCR была успешно использована в ряде коммерческих приложений (см., например, [4, 5]). В основе VCR-технологии положен принцип сопоставления распознаваемой речевой команды с набором эталонов методами динамического программирования. При объеме словаря до 1 000 слов VCR-технология позволяет достичь высокой достоверности распознавания и к настоящему времени достаточно широко используется в мобильных устройствах при голосовом наборе и навигации по сайтам.

STT-технология применяет теорию скрытых Марковских моделей, с помощью которых реализуется метод статистического сравнения распознаваемой фразы с эталонами. При этом используются акустические модели слов, а также грамматические модели фраз и предложений. Задача высококачественного распознавания слитной речи в рамках STT-технологии еще не решена нигде в мире, хотя достоверность распознавания уже достаточно высока для ее использования в ряде практических приложений. Дальнейшее возрастание точности распознавания зависит от качества используемых акустических и языковых моделей. Для обучения этих моделей необходим большой объем исходного речевого материала. Так, для создания акустических

моделей требуются сотни часов записей речи для нескольких тысяч дикторов. При этом для повышения устойчивости распознавания к помехам и искажениям необходимо использовать записи, созданные в различных каналах и условиях. Еще более жесткие требования предъявляются к разработке языковых моделей распознавания. Для обучения языковых моделей необходимо использовать корпусы текстов объемом от сотен миллионов словоформ до нескольких миллиардов. Подготовка и обработка такого объема обучающего материала – это сложная, кропотливая и весьма дорогостоящая работа [6].

Архитектура системы APP определяется местом обработки речевого сигнала: на клиентском компьютере или на сервере. Первая – наиболее привычная и распространенная архитектура – называется встроенной. При такой реализации вся обработка речевого сигнала и распознавание выполняются на компьютере клиента или мобильном устройстве. Данное решение обладает целым рядом недостатков. Прежде всего, это проблемы, связанные с большим разнообразием существующих архитектур компьютеров и мобильных устройств. Для мобильных устройств это еще и дополнительные ограничения, накладываемые на вычислительные ресурсы и память.

Второй вариант построения систем APP – более мощная и более гибкая альтернатива встроенной системе. Эти системы построены на основе клиент-серверной архитектуры с использованием технологий облачных вычислений. Клиентский компьютер (мобильный телефон, смартфон, планшет или нетбук) в такой архитектуре осуществляет только ввод и передачу речевого сигнала по цифровому каналу связи на удаленный сервер, а сервер выполняет основную работу – распознавание полученной последовательности данных. Кроме того, у этой архитектуры отсутствуют ограничения на вычислительные ресурсы клиентского компьютера, что дает возможность использовать современные и более сложные алгоритмы распознавания, а также централизованно поддерживать и обновлять серверную программу системы APP [7].

На сегодняшний день компания Google является лидером по предоставлению облачных технологий распознавания речи [8]. В течение последних пяти лет активно развивалась облачная технология распознавания речи Google Voice, и к настоящему времени существуют технологии распознавания речи для большинства европейских языков, включая русский, а также японский и китайский. Одним из немаловажных компонентов системы распознавания речи Google Voice является обучающая выборка звукозаписей человеческого голоса. Для системы Google Voice источником таких записей являются различные сервисы, предоставляемые Google и использующие речевые технологии. К ним относятся система распознавания речи и команд в системе Android, сервис диктовки писем Google Mail, телефонная справочная система Goog411 и др. [9]. Таким образом, обучающая выборка постоянно пополняется новыми образцами голосов как с особенностями произношения, так и эффектами, от которых зависят технические характеристики записи и передачи голоса на различных устройствах. К примеру, в 2011 г. обучающая выборка для английского языка составляла примерно 230 млрд записей слов, извлеченных из реальных запросов [10]. Для обработки таких объемов информации требуется около 70 лет процессорного времени, однако при использовании облачной технологии Google время сокращается до одного дня [9].

Рассматриваемая реализация распознавания речи включает в себя три модели: акустическую, лексическую и языковую [8]. Акустическая модель ответственна за распознавание фонем, она учитывает все возможные варианты произношения, а также другие особенности, такие как тип используемого микрофона (качество записи), фоновые шумы, возраст и пол говорящего и многое другое. Наиболее важным в данном случае является объем обучающей выборки: чем он больше, тем лучше будет результат распознавания. В лексической модели фонемы объединяются в слова на основе словарей, в которых указаны различные варианты произношения слов. Языковая модель объединяет слова, используя статистический подход. На основе анализа поисковых запросов, текстов Интернета выделяются вероятности взаимного расположения слов в предложении.

3. Экспериментальная прикладная программа STENOGRAPH

Для оценки современного состояния разработки облачных интернет-технологий распознавания речи, а также возможности построения на их основе готовых речевых приложений был проведен эксперимент по использованию русскоязычной версии системы распознавания Google Voice для решения задачи стенографирования устной речи. Подобный программный продукт мог бы найти достаточно широкое применение во многих сферах деятельности (бизнес, медицина, веб-приложения) в качестве средства, заменяющего (дополняющего) клавиатурный ввод.

В качестве входных требований к проекту были взяты следующие положения:

– программное средство должно реализовываться в виде приложения под ОС Windows;

– работа с программным средством должна вестись не только в ручном стартстопном режиме (как это предусмотрено системой Google Voice), но и в автоматическом режиме путем дополнительного включения в его состав детектора речи, определяющего начало и конец голосового сигнала;

– программное средство должно обладать простым и понятным графическим интерфейсом;

– к программному средству должны предъявляться минимальные требования по установке и настройке.

В связи с этим в качестве средств реализации были выбраны язык программирования C++, который обеспечивает максимальную производительность при захвате и обработке речевого сигнала, и библиотека построения пользовательских интерфейсов Qt Quick, которая позволяет в кратчайшие сроки разрабатывать кроссплатформенные пользовательские интерфейсы.

В экспериментальной системе автоматического стенографирования выделены следующие программные компоненты:

– детектор речи;

– компонент сжатия и обработки речевого сигнала;

– компонент для работы с распознаванием речи Google;

– графический пользовательский интерфейс.

Детектор речи является необходимым дополнительным компонентом, обусловленным используемой архитектурой удаленного распознавания речи. В этом случае канал передачи данных (в нашем случае интернет-соединение) является «бутылочным горлышком», ограничивающим максимально возможный объем передачи данных в заданный промежуток времени. При передаче по каналу слишком длинного отрезка речевого сигнала происходит недопустимо длительная задержка ответной реакции системы распознавания. Кроме того, при этом возникает значительный риск получения ошибочных результатов распознавания. Ввод коротких отрезков речи возможен при использовании ручного стартстопного режима, предоставляемого системой Google Voice. Однако для решения поставленной здесь задачи стенографирования устной речи этот режим оказывается крайне неудобным. Обычно диктовка осуществляется короткими фразами, заканчивающимися паузами. После произнесения каждой из них пользователь, как правило, желает убедиться в правильности распознавания и при необходимости повторить ее более четко.

Задача детектора речи заключается в автоматической локализации полезного сигнала, т. е. в определении начала и конца произнесенной фразы. Это обеспечивает автоматический пофразовый ввод речи. В экспериментальной программной системе использован относительно простой, но достаточно эффективный алгоритм определения полезного потока речи, основанный на расчете усредненной текущей энергии звуковой волны входного сигнала. Перед началом работы в течение одной секунды производится замер фонового шума для определения порога срабатывания детектора речи. Порог вычисляется как произведение средней энергии фонового шума на коэффициент, заданный в настройках программы (по умолчанию равный 2,0).

Далее ведется запись в кольцевой буфер задержки сигнала (по умолчанию задержка равна 1,0 с) и одновременно вычисляется текущая средняя энергия звуковой волны входного сигнала (по умолчанию время усреднения равно 0,4 с). Если порог будет превышен, задержанный входной сигнал передается на вход компонента сжатия и обработки сигнала, а передача сигнала будет вестись до тех пор, пока значение средней энергии не станет меньше значения порога.

Принцип работы детектора речи поясняется на примере произнесенной фразы «Раз два три» (рис. 1).

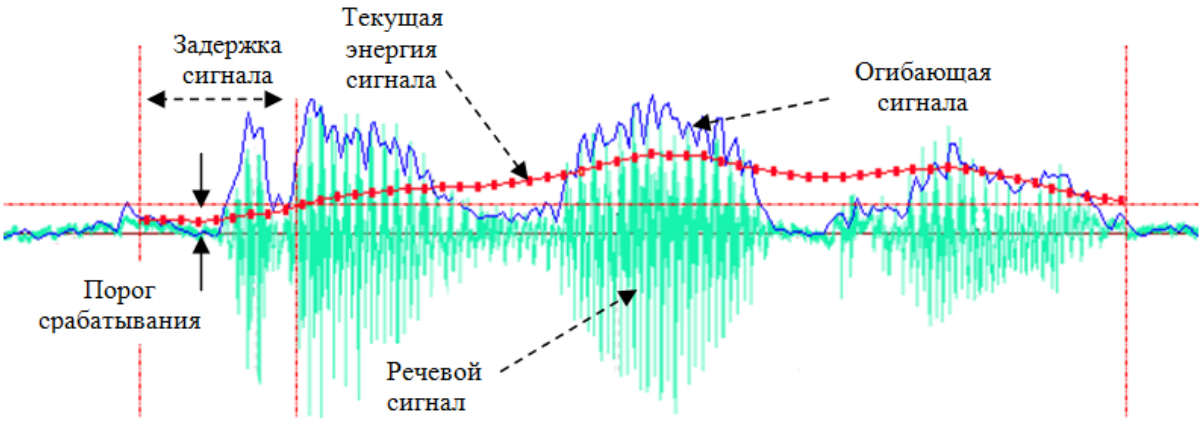


Рис. 1. Пояснения к принципу работы детектора речи

Полученный полезный сигнал в модуле сжатия и обработки сигнала подготавливается для отправки на удаленный сервер распознавания речи Google Voice. Для этого сигнал кодируется в открытом формате FLAC (Free Lossless Audio Codec) [12] с заданными характеристиками (частота дискретизации 16 кГц, моно). При этом происходит сжатие сигнала кодеком, что уменьшает объем передаваемых данных и как следствие сокращает время ожидания результата распознавания. Использование open-source-кодека имеет и другие преимущества: простоту использования сторонними разработчиками, единообразие получаемого сервером формата данных. Каждый компонент программы работает асинхронно и имеет свои пулы данных для нивелирования эффектов, связанных с задержками в работе каждого компонента. Такие задержки появляются как на этапе записи и кодирования речевого сигнала в файл, так и на этапе отправки его на удаленный сервер.

Полученная запись речевого сигнала прикрепляется к get-запросу на распознавание, отправляемому на сервер Google. Get-запрос имеет следующую форму: http://www.google.com/speech-api/v1/recognize?client=chromium&lang=ru-Ru&maxresults=3. В поле lang get-запроса указывается используемый язык (в нашем случае русский), а в поле maxresults – количество возвращаемых вариантов распознавания. Ответ на такой запрос от сервера приходит в формате JSON (JavaScript Object Notation) [13]. Ниже приведен пример ответа на запрос распознавания:

```
{
«status»:0,
«id»:«5e34348f2887c7a3cc27dc3695ab4575-1»,
«hypotheses»:[
{«utterance»:«один два три»,«confidence»:0.7581704},
{«utterance»:«1 2 3»},
{«utterance»:«один 2 три»}
}
```

В этом примере если поле status равно 0, запись успешно распознана, если 5 – запись не распознана. Поле id – это уникальный идентификатор запроса; поле hypotheses – результат распознавания, в нем есть два подполя: utterance – распознанная фраза и confidence – достоверность распознавания. Полученные данные можно анализировать и выдавать пользователю в наиболее подходящем виде. Например, числовые комбинации – в виде чисел или слов.

На рис. 2 описанный алгоритм более подробно представлен в графическом виде.

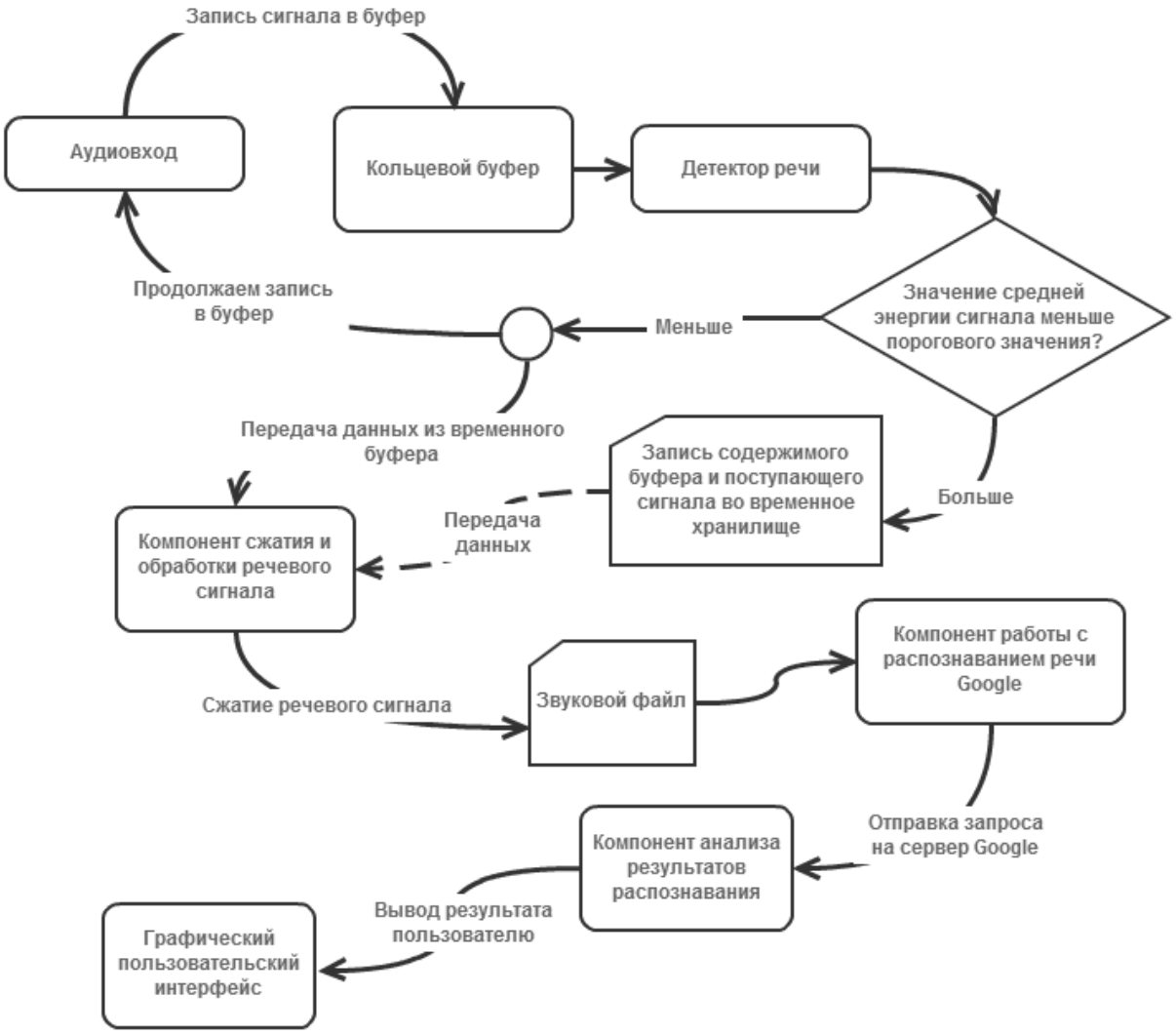


Рис. 2. Схема алгоритма работы прикладной программы STENOGRAPH

4. Пользовательский интерфейс и предварительные результаты тестирования

Рассмотрим главное окно программы STENOGRAPH (рис. 3).

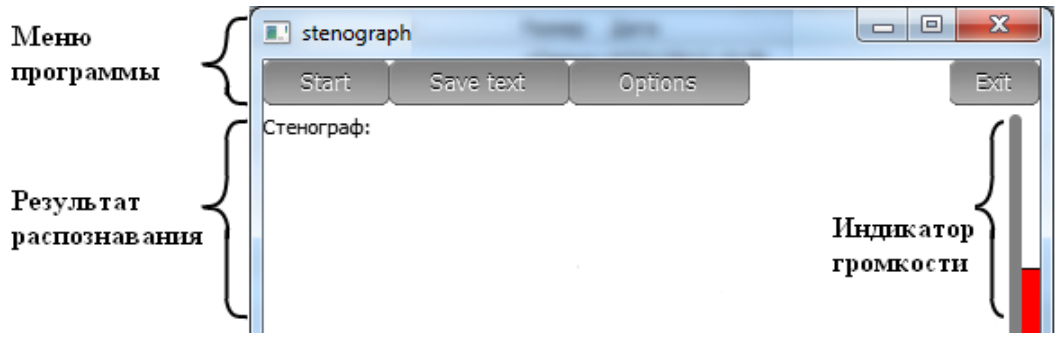


Рис. 3. Структура рабочей области программы STENOGRAPH

Меню программы включает шесть функциональных кнопок:
Start|Stop – запуск и остановка записи;

Play – проиграть последнюю запись (только в ручном режиме);
Recognize – распознать последнюю запись (только в ручном режиме);
Save text – сохранить набранный текст в файл;
Options – настройки программы;
Exit – выход.
В режиме Options открывается окно настроек программы (рис. 4).

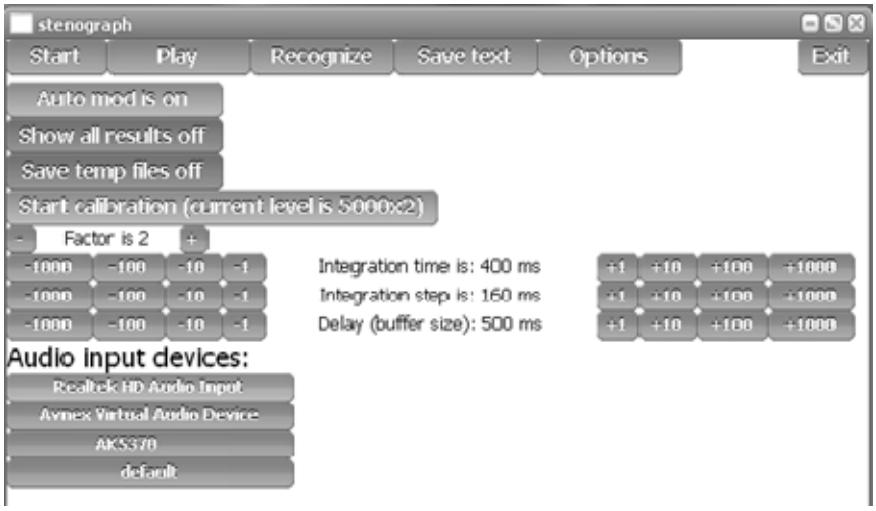


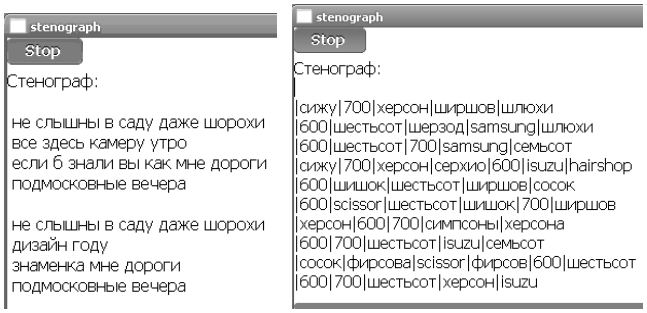
Рис. 4. Настройки программы STENOGRAPH

Окно настроек содержит:
переключатель Auto mod on/off – выбрать режим работы: автоматический либо ручной;
переключатель Show all result on/off – вывести все или только наилучшие варианты распознавания;
переключатель Save temp files on/off – сохранить или удалить произносимые звуковые файлы;
кнопку Start calibration – запустить режим калибровки уровня входного шума;
кнопку Audio input devices – выбрать тип устройства ввода речевого сигнала.
Кроме перечисленных основных кнопок окно содержит и ряд дополнительных, используемых при экспериментальной отладке программы в случае смены окружающих акустических условий (например, микрофона, устройства ввода речи) и в других случаях. К таким дополнительным кнопкам относятся:
Factor is (+/–) – для подстройки порога срабатывания детектора речи путем увеличения или уменьшения коэффициента, заданного по умолчанию 2,0;
Integration time is (+/–) – для подстройки постоянной времени интегратора детектора речи путем ее увеличения или уменьшения, заданной по умолчанию 400 мс;
Integration step is (+/–) – для подстройки временного интервала считывания сигнала путем его увеличения или уменьшения, заданного по умолчанию 160 мс;
Delay (buffer size) (+/–) – для подстройки временной задержки речевого сигнала путем ее увеличения или уменьшения, заданной по умолчанию 500 мс.
Последовательность действий при работе с программой:
1. Запустить программу stenograhp.exe.
2. Выбрать из меню пункт Options.
3. Нажать кнопку Start calibration. В данном режиме будет произведена калибровка уровня фонового шума. Для этого в течение одной секунды требуется сохранять тишину.
4. Для закрытия окна настроек повторно нажать в меню пункт Options.
5. Для начала записи выбрать из меню пункт Start.
6. Теперь можно начать надиктовку фраз. Для более корректной работы фразы должны быть не длиннее 5 с (рекомендуется, но не обязательно).

Если выбран режим Show all result off, в окне главной программы (рис. 5, *а*) появляется с новой строки наилучший вариант распознавания каждой фразы. Если же выбран режим Show all result on, в каждой строке выводятся также и другие, менее вероятные результаты (рис. 5, *б*).

7. По окончании работы необходимо выбрать из меню пункт Stop.

8. Для сохранения введенного текста нужно выбрать из меню пункт Save file. Текст сохранится в файл txt.

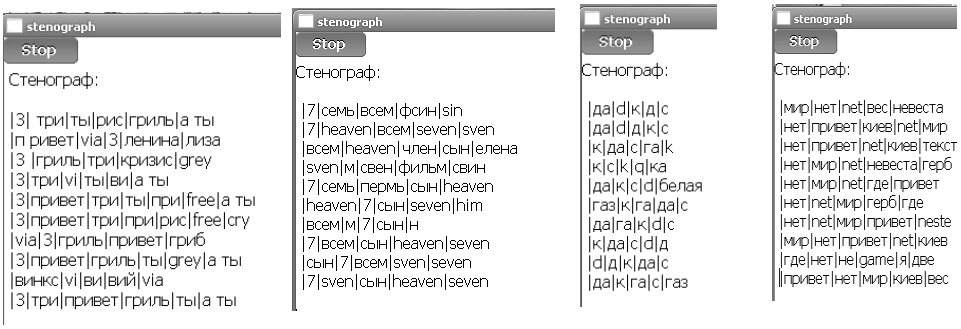


а) *б)*

Рис. 5. Примеры отображения результатов распознавания:
а) режим Show all result off – дважды повторенные четыре фразы из текста песни «Подмосковные вечера»;
б) режим Show all result on – десятикратное повторение слова «шестьсот»

Для предварительной оценки потребительских качеств программы было проведено предварительное испытание надежности распознавания для двух наиболее практически важных категорий слов: названий цифр от 0 до 9, речевых ответов «Да» и «Нет».

Испытания проводились в реальных акустических условиях лаборатории с использованием внутреннего микрофона ноутбука. В испытаниях участвовали три диктора: двое мужчин и одна женщина. Средняя надежность распознавания названий цифр и речевых ответов составила около 90 %. При этом основной массив ошибок распознавания чисел приходился не на цифры 3 и 7, а на речевой ответ «Да». Однако если в соответствии с решаемой прикладной задачей на определенных этапах диалога речь идет только о распознавании названий цифр или речевых ответов, то надежность их распознавания можно существенно повысить, если использовать режим Show all result on. Этот факт хорошо виден на рис. 6, где приведены варианты принятия решения для цифр и речевых ответов при их десятикратном повторении.

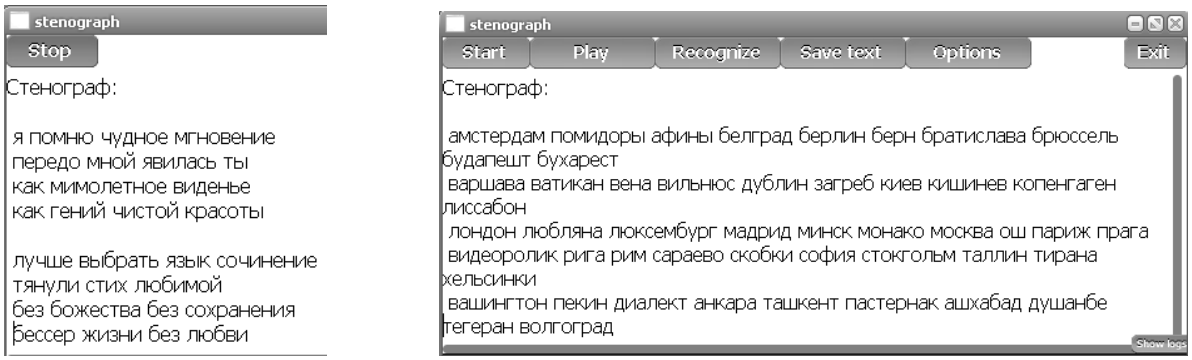


а) *б)* *в)* *г)*

Рис. 6. Варианты принятия решения для цифр и речевых ответов

Важной особенностью рассматриваемой системы является то, что надежность распознавания практически одинакова для раздельного и слитного произнесения слов. Система характеризуется также весьма обширным допустимым словарем распознавания. Однако ощутимым недостатком системы является то, что зачастую незнакомое ей слово она не отвергает, а заменяет другим словом, близким по звучанию. Так, при распознавании двух строф стихотворения Пушкина (рис. 7, *а*) первая строфа распознана безошибочно, а вторая лишь отдаленно напоми-

нает истинный ее текст, а при распознавании слитно произнесенных названий мировых столиц (рис. 7, *б*) произошли следующие замены: Андорра – помидоры, Осло – Ош, Рейкьявик – видеоролик, Скопье – скобки, Дели – диалект, Астана – пастернак.



а) *б)*

Рис. 7. Результат распознавания: *а)* двух строф стихотворения Пушкина; *б*) слитно произнесенных названий мировых столиц

Заключение

По результатам работы прикладной программы STENOGRAPH и проведенных экспериментов по распознаванию речи можно сделать следующие выводы:

– программа STENOGRAPH обеспечивает достаточно надежное автоматическое интерактивное взаимодействие пользователя с облачной системой распознавания речи Google Voice;

– использование общепринятого и открытого формата сжатия речевого сигнала (flac) не создает большой нагрузки на канал передачи данных, что позволяет создавать системы, активно реагирующие на голосовые команды пользователя;

– достаточно высокая для практики надежность распознавания может быть достигнута в случае, если в прикладной диалоговой системе используется режим Show all result on;

– отсутствие механизмов обратной связи не позволяет создавать системы с настройкой под определенного пользователя, что могло бы значительно улучшить надежность распознавания для дикторов с особенностями произношения;

– отсутствие возможности влиять на используемые акустические модели слов, а также грамматические модели фраз и предложений не позволяет производить настройку приложения на определенную предметную область, что могло бы существенно улучшить надежность распознавания.

Облачная система распознавания русской речи Google Voice обладает положительными свойствами, такими как способность к распознаванию дискретной и слитной речи, потенциальная устойчивость к смене диктора. Система использует большой по объему словарь русских слов и словоформ, однако их количество не может обеспечить безошибочное стенографирование текстов произвольной тематики.

Список литературы

1. Public Cloud Service Definition. Public Version 1.5 // WMWare, Inc. [Electronic resource]. – 2010. – Mode of access : http://www.vmware.com/files/pdf/VMware-Public-Cloud-Service-Definition.pdf. – Date of access : 01.08.2012.

2. Mell, P. The NIST Definition of Cloud Computing. Recommendations of the National Institute of Standards and Technology / P. Mell, T. Grance // U.S. Department of Commerce [Electronic resource]. – NIST Special Publication, 2011. – Mode of access : http://csrc.nist.gov/ publications/nistpubs/800-145/SP800-145.pdf. – Date of access : 01.08.2012.

3. Schalk, C. Google Cloud Technologies Overview / C. Schalk // Cloud Computing Expo, 2010 [Electronic resource]. – Mode of access : http://www.slideshare.net/cschalk/google-cloud-technologies-overview. – Date of access : 01.08.2012.

4. Lobanov, B.M. Continuous Speech Recognizer for Aircraft Application / B.M. Lobanov, T.V. Levkovskaya // Proc. of the 2nd Intern. Workshop «Speech and Computer» – SPECOM'97. – Cluj-Napoca, 1997. – P. 97–102.

5. An Intelligent Answering System Using Speech Recognition / B.M. Lobanov [et al.] // Proc. of the 5th European Conf. on Speech Communication and Technology – EUROSPEECH'97. – Rhodes-Greece, 1997. – Vol. 4. – P. 1803–1806.

6. Распознавание речи // Центр речевых технологий [Электронный ресурс]. – 2001–2012. – Режим доступа : http://www.speechpro.ru/technologies/recognition. – Дата доступа : 01.08.2012.

7. Маковкин, К.А. Удаленная система автоматического распознавания речи / К.А. Маковкин // Речевые технологии. – № 4. – 2009. – С. 70–96.

8. Manjoo, F. Now You're Talking / Farhad Manjoo // The Slate Group. – Washington Post Company [Electronic resource]. – 2012. – Mode of access : http://www.slate.com/articles/technology/technology/2011/04/now_youre_talking.single.html. – Date of access : 01.08.2012.

9. Singhal, A. Knocking down barriers to knowledge / Amit Singhal // Google Official Blog [Electronic resource]. – 2011. – Mode of access : http://googleblog.blogspot.com/2011/06/knocking-down-barriers-to-knowledge.html. – Date of access : 01.08.2012.

10. Enge, E. Search Algorithms with Google Director of Research Peter Norvig / E. Enge // Stone Temple Consulting [Electronic resource]. – 2011. – Mode of access : http://www.stonetemple.com/search-algorithms-with-google-director-of-research-peter-norvig. – Date of access : 01.08.2012.

Поступила 16.08.12

Объединенный институт проблем
информатики НАН Беларуси,
Минск, Сурганова, 6
e-mail: lobanov@newman.bas-net.by
zhitko.vladimir@gmail.com

B.M. Lobanov, V.A. Zhitko

APPLICATION OF CLOUDY TECHNOLOGIES IN SPEECH RECOGNITION

Features of cloud Internet technologies in automatic speech recognition are considered. Positive peculiarities and basic models of modern cloud Internet technologies are briefly described. The existing methods, technology and architecture of automatic speech recognition, and also features of cloud technology for speech recognition provided by Google are considered. STENOGRAPH, experimental applied software based on this technology, its user interface and preliminary results of the testing are described.

ПРИКЛАДНЫЕ ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

УДК 004.4:004.7

М.К. Буза, О.М. Кондратьева, В.О. Шукело

ГРАФИЧЕСКАЯ СИСТЕМА ПОДГОТОВКИ И УПРАВЛЕНИЯ ЗАДАНИЯМИ ПОЛЬЗОВАТЕЛЕЙ НА КЛАСТЕРЕ

Предлагается приложение, которое позволяет управлять прохождением задач на суперкомпьютере и имеет графический интерфейс. Приложение поддерживает реализации на языке C/C++ на основе технологий MPI, OpenMP, POSIX Threads.

Введение

Применение высокопроизводительных распределенных вычислений открывает широкие возможности для решения сложных задач. Это обусловливает возникновение различных проблем: от распараллеливания алгоритмов и оптимизации их работы на современных компьютерах до разработки средств поддержки доступа к вычислительным системам.

В настоящее время доступ к вычислительным ресурсам кластера предъявляет достаточно высокие требования к квалификации пользователей. Им необходимо владеть многими дополнительными навыками, такими как копирование файлов на суперкомпьютер и обратно, получение доступа к командной строке кластера, умение работать с командной строкой. Поэтому актуальной является задача разработки графического интерфейса для упрощения доступа пользователей к кластеру. Такой интерфейс позволит расширить круг пользователей суперкомпьютеров.

Задача интерфейса – обеспечить выполнение всех действий, необходимых пользователю, только средствами интерфейса. Правильно организованный интерфейс позволяет абстрагироваться от многих деталей, связанных с доступом к кластеру.

В данной работе рассматривается графическая система, которая автоматизирует основные операции, выполняемые пользователями кластера: управление файлами, компиляцию и запуск задач, а также мониторинг вычислительного процесса. Проект выполнен при частичной поддержке ГПНИ «Информатика и космос». Апробация всего функционала разработанного интерфейса осуществлялась на суперкомпьютере СКИФ БГУ.

1. Суперкомпьютер СКИФ БГУ

На суперкомпьютере СКИФ-1000-2 установлена операционная система (ОС) Linux Fedora 8.0, которая обеспечивает основу для выполнения всех остальных программ. Удаленный доступ к суперкомпьютеру СКИФ осуществляется по протоколу SSH (Secure SHell).

Для запуска заданий пользователей на кластере используется система управления заданиями Torque PBS (Portable Batch System). Основная функция PBS – запуск задач в вычислительной среде по расписанию. Torque PBS наиболее часто используется для управления вычислительным процессом в кластерах, построенных из компьютеров под управлением ОС GNU/Linux и других Unix-подобных ОС.

Доступ к суперкомпьютеру осуществляется с любого компьютера локальной сети БГУ. Для работы на суперкомпьютере необходимо установить на персональном компьютере программы, которые обеспечат удаленный доступ к командной строке сервера и возможность обмена файлами между компьютером и сервером.

К основным операциям, которые выполняют пользователи кластера, относятся: файловые, компиляция и запуск приложений, слежение за процессом выполнения задачи. Для обмена файлами между компьютером и сервером существуют программы с графическим интерфейсом. Например, пользователь, у которого на локальной машине установлена ОС Windows, может

использовать для файловых операций программы WinSCP или FileZilla, а для подключения к командному интерфейсу кластера – SSH-клиент PUTTY.

Подключившись к кластеру, пользователь осуществляет традиционные операции: компиляцию и запуск приложений, просмотр результатов, слежение за процессом выполнения. Управлять суперкомпьютером приходится посредством интерфейса командной строки, что для неподготовленного пользователя может оказаться неудобным и сложным.

2. Средства поддержки доступа к кластеру

В последнее время начинают разрабатываться средства для упрощения доступа к вычислительным ресурсам кластера. Существуют некоторые разработки для запуска заданий пользователей на кластере посредством графического интерфейса, например графический интерфейс управления задачами ОС Clustrx [1] и система управления суперкомпьютером компании Мелкон [2].

Суперкомпьютерная ОС Clustrx является разработкой российской компании T-Massive Computing. ОС имеет интерактивный графический веб-интерфейс пользователя, который реализует функции управления файлами, задания параметров и запуска задачи на выполнение, слежения за состоянием очереди задач. Возможности, предоставляемые интерфейсом, различаются для пользователей разного уровня привилегированности.

Система управления суперкомпьютером украинской компании Мелкон представляет собой веб-портал для управления кластером. Система объединяет интерфейс пользователя и средства администратора. Интерфейс пользователя предоставляет сервисы компиляции и запуска задач, работы с файлами пользователя, обмена сообщениями с другими пользователями и администраторами. Интерфейс администратора имеет возможности управления пользователями, мониторинга кластера, диагностики, просмотра статистики использования ресурсов, уведомления о неполадках.

Обе системы реализуют необходимый функционал и имеют графический интерфейс. Однако первая представляет собой специализированную ОС, а для управления второй требуется участие системного администратора. Эти проекты поддерживают доступ к кластеру через графический интерфейс, тем не менее сохраняя при этом стиль ОС Linux. Кроме того, они являются коммерческими программными продуктами. Поэтому задача разработки собственной графической системы, покрывающей функции разноуровневого пользователя, является актуальной и практически значимой.

В данной работе предлагается система управления кластером, которая обладает следующими свойствами:

– упрощает подготовку и управление заданиями пользователей;

– ориентирована на пользователей с неадминистративным уровнем полномочий, работа которых с кластером заключается в выполнении на нем вычислительных задач;

– имеет удобный графический интерфейс, через который осуществляется взаимодействие с системным программным обеспечением суперкомпьютера;

– не нуждается в дополнительной настройке со стороны сервера;

– требует для работы только установленную виртуальную машину Java (JVM версии 1.6 и выше).

3. Структура и функции графической системы

Система подготовки и управления заданиями состоит из двух основных частей: ядра и пользовательского интерфейса. Ядро системы отвечает за установление подключения к кластеру. Оно взаимодействует с ОС и PBS, выполняя всю работу по обслуживанию запросов от интерфейса пользователя. Основой визуальной части системы является настольное приложение.

Система поддерживает программные реализации на языке C/C++ на основе технологий MPI, OpenMP, POSIX Threads. Ее легко можно настроить на работу с прикладными пакетами, например с пакетом молекулярной динамики LAMMPS.

Основным объектом, с которым работает система, является задача. Она выполняется на сервере и представляет собой программу с ее входными данными и параметрами для обработки. Система создает и поддерживает на сервере базу данных, содержащую сведения о доступных задачах. Каждая задача характеризуется набором параметров. Значения одних параметров задаются через интерфейс (пользователем или по умолчанию), других – определяются системой. Параметры и их назначение приведены в таблице.

Параметры задачи

Название	Описание
Name	Имя задачи
Status	Состояние задачи
Task id	Идентификатор задачи, присваиваемый системой при создании задачи и используемый как имя директории на сервере
Source file	Путь к файлу или директории с исходным кодом задачи на локальном компьютере
Source mode	Тип задачи, определяемый используемой технологией (MPI, OpenMP, POSIX Threads)
Input file	Путь к файлу или директории с входными данными для задачи на локальном компьютере
Output file	Путь к файлу или директории для выходных данных на локальном компьютере
Node count/list	Количество или список узлов, запрашиваемых пользователем
Walltime	Время, которое отводится на выполнение задачи. В случае если задача не успеет выполниться за это время, она будет принудительно завершена
Add-al cmdpars	Параметры командной строки
Last run time	Время выполнения задачи
Completed date	Дата и время завершения задачи

Задача имеет различные представления: запись в базе данных, директория на сервере, запись в Torque PBS. Во время своего жизненного цикла задача может находиться в одном из следующих состояний:

created – созданная;
compilation-failed – с ошибками компиляции;
compiled – успешно скомпилированная;
scheduled – добавленная в очередь на выполнение;
running – выполняемая в данный момент;
completed – нормально завершившаяся;
aborted – завершившаяся неудачно или отмененная;
purged – удаленная с сервера.

4. Графический интерфейс пользователя

Интерфейс и возможности системы соответствуют пользователям двух типов:

– ученые-исследователи, которые работают на кластере с готовыми пакетами и выполняют подготовку файлов входных данных, запуск приложений и просмотр файлов выходных данных;

– прикладные программисты, которые используют кластер как инструмент настройки параллельных программ. Программистам необходимо компилировать и запускать задачи и следить за процессом их выполнения.

Рассмотрим действия, которые необходимо выполнять всем пользователям: подключение к кластеру и управление своими задачами.

Подключение к суперкомпьютеру. После запуска графического интерфейса появляется окно входа в систему. Здесь пользователь вводит имя и пароль. Каждому потенциальному пользователю логин и пароль выдаются администратором кластера по письменному заявлению. Уровень доступа определяется статусом пользователя. Единые идентификационные данные для группового доступа не предусмотрены.

После подключения к суперкомпьютеру открывается главное окно интерфейса (рис. 1). В нем отображается список собственных задач пользователя. Для каждой задачи в этом списке указываются: состояние; расположение исходного кода на локальном компьютере; имя, данное пользователем, и идентификатор, присвоенный системой. Список задач автоматически обновляется при изменении их состояния.

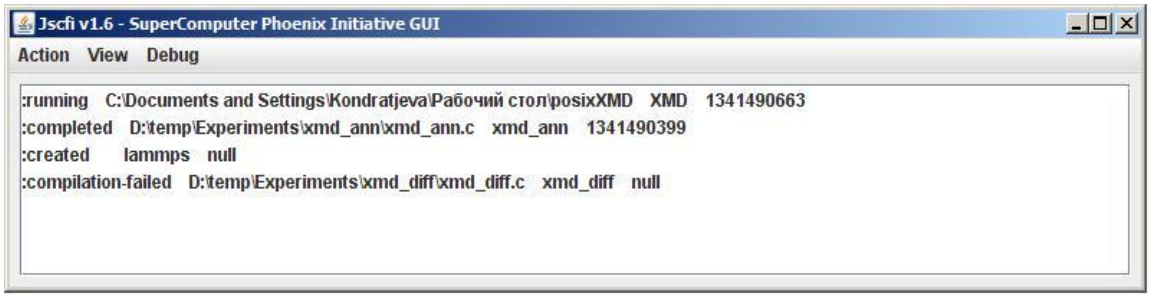


Рис. 1. Главное окно системы

Управление задачами включает следующие акции.

Создание новой задачи. При создании задачи пользователь определяет значения ее параметров в соответствующем окне (рис. 2). После создания задача переходит в состояние created. Значения параметров, связанных с местоположением файлов, задаются относительно локального компьютера. Система сама при необходимости выполняет файловые операции между локальным компьютером и суперкомпьютером, а пользователь работает с файлами, используя привычное для себя программное обеспечение.

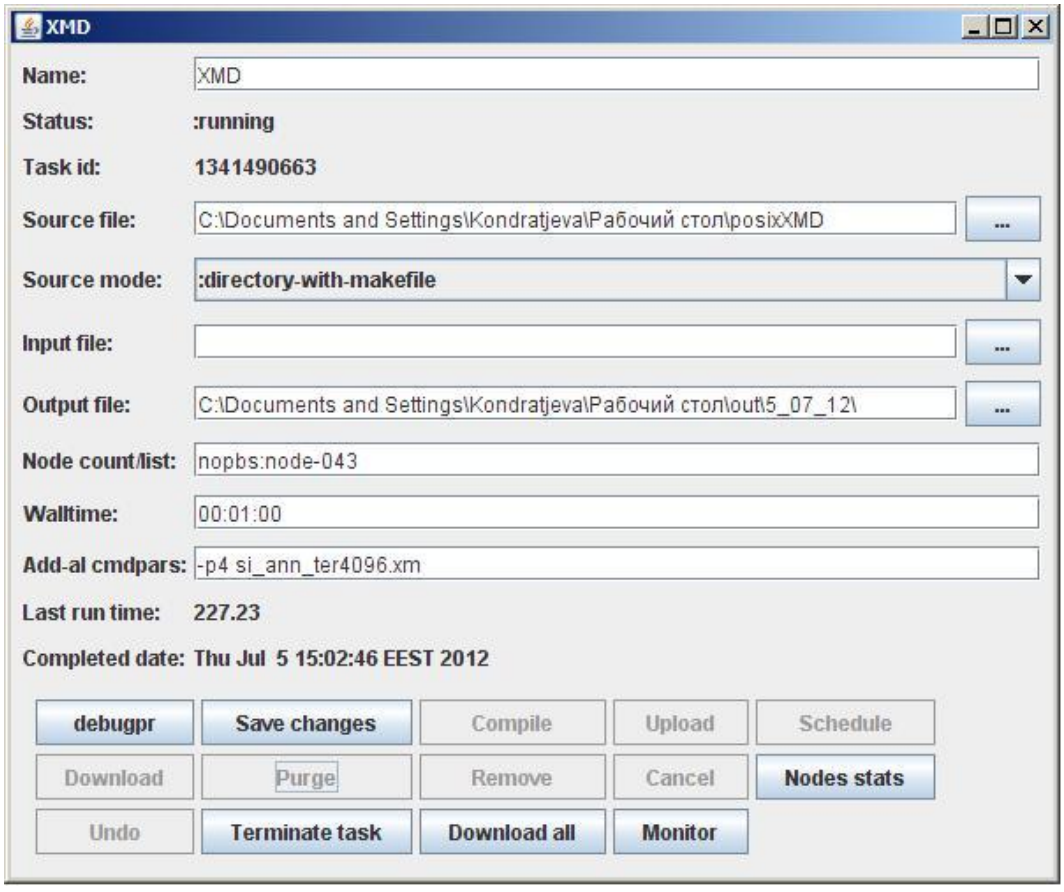


Рис. 2. Окно создания новой задачи

Компиляция и выполнение задачи. Созданную задачу нужно откомпилировать и в случае успешной компиляции запустить на выполнение.

Файлы с исходными текстами программ, расположенные на локальном компьютере, сначала копируются на кластер, а затем компилируются. После успешной компиляции задача переходит в состояние compiled.

При ошибках компиляции задача переходит в состояние compilation-failed. Если пользователь вносит изменения в файлы с исходными текстами программ, необходимо обновить эти файлы на кластере (команда Upload) и провести повторную компиляцию (Compile).

Для задачи в состоянии compiled пользователь может:

– скопировать входные данные на сервер (Upload);

– запустить задачу на выполнение (Schedule).

При выборе второго действия система создает скрипт для запуска и помещает задачу в очередь на выполнение. Задача переходит в состояние scheduled. После того как задача начинает выполняться, она переходит в состояние running. В случае успешного завершения задачи ее статус меняется на completed и в поле Last run time отображается время, которое прошло от старта до завершения задачи. В случае неуспешного или принудительного (Terminate task) завершения задача переходит в состояние aborted и в поле Last run time отображаются время выполнения и код завершения задачи.

Для задачи в состоянии completed пользователь может:

– скопировать выходные данные на локальный компьютер (Download);

– подготовить новые входные данные и запустить задачу;

– удалить файлы задачи с сервера (Purge), задача переходит в состояние purged;

– удалить задачу из списка задач (Remove).

Копирование результатов. Вычислительная задача пользователя может выводить данные в файлы, находящиеся на кластере. Чтобы скопировать эти файлы на локальную машину, пользователь должен выполнить команду Download.

Таким образом, пользователи, работающие с готовыми пакетами, могут придерживаться следующего сценария:

– создавать требуемые задачи;

– запускать созданные задачи на выполнение;

– наблюдать за своими задачами при необходимости;

– просматривать файлы выходных данных для завершившихся задач.

5. Мониторинг вычислительного процесса

Анализ эффективности параллельной программы представляет собой сложный процесс, для поддержки которого существуют различные вспомогательные программные средства [3], например мониторинг вычислительного процесса. Средства мониторинга, как правило, собирают и накапливают данные о функционировании и производительности приложений. Эти данные используются для анализа работы приложений и могут быть полезны как ученым-исследователям, так и прикладным программистам для оптимизации процесса вычислений.

Средства мониторинга зависят от того, кто и для чего их использует. Это может быть анализ работы вычислительной системы в целом [4]. Пользователям суперкомпьютера необходим мониторинг своих приложений. И здесь возможны варианты. Одним достаточно знать только время выполнения задачи, другим требуется более детальная информация для исследования параллельных алгоритмов и оптимизации их реализаций: сколько времени приложение находилось в каждом из состояний, на каких узлах оно выполнялось, как выполнялось приложение по сравнению с предыдущими запусками.

Мониторинг системы можно осуществлять, используя команды ОС Linux [5, 6]. Например, команда top выводит в консоль список процессов с сортировкой по процессорному времени. В этом списке отображается, в частности, состояние процесса (выполняется, ожидает и т. д.) и занимаемое им процессорное время. Система управления заданиями Torque PBS позволяет получить аналогичную информацию с помощью команды qstat. Периодически вызывая команду

top или qstat, можно контролировать состояние своих задач, что требует некоторых усилий со стороны пользователя. При этом пользователь видит не только свои задачи, но и задачи других пользователей. В предлагаемой системе в главном окне отображается обновляемый список собственных задач пользователя (см. рис. 1).

Команда time ОС Linux позволяет выполнить программу и получить информацию о времени, которое прошло от запуска до завершения задачи. В разработанном интерфейсе эта информация после завершения задачи выводится в поле Last run time (см. рис. 2).

По команде pbsnodes системы управления заданиями Torque PBS отображается текущее состояние вычислительных узлов кластера. Это действие также реализовано в разработанной системе (Nodes stats).

На эффективность параллельной программы влияют, в частности, конфигурация компьютера и системное программное обеспечение. Поэтому программисты исследуют свои реализации и адаптируют их к конкретным многопроцессорным вычислительным системам. Если время расчета задачи оказывается неудовлетворительным, программист нуждается в дополнительных данных о функционировании программы. Это могут быть уровень загрузки процессоров, объем занятой задачей оперативной памяти, интенсивность ввода/вывода. ОС Linux ведет системные журналы, в которых накапливает необходимые сведения. Доступ к этим журналам требует определенных навыков. Предлагаемая система позволяет запустить приложение в режиме мониторинга, в котором для вычислительных узлов задачи сохраняются данные для анализа производительности приложения.

Просмотреть результаты мониторинга можно в текстовом и графическом виде. В качестве примера на рис. 3 представлены данные по загрузке вычислительного узла, на котором работает двухпоточное приложение. Пользователи могут подключать к системе свои визуализаторы.

```
node-043 1341485637.527430 174.207
node-043 1341485641.529430 179.410
node-043 1341485645.531450 171.413
node-043 1341485649.533190 161.430
node-043 1341485653.534200 157.460
node-043 1341485657.537440 159.121
node-043 1341485661.538430 158.961
node-043 1341485665.540190 156.931
node-043 1341485669.542210 158.420
node-043 1341485673.543200 156.211
```

а)

б)

Рис. 3. Данные мониторинга «Загрузка вычислительного узла»: *а)* фрагмент текстового файла; *б)* график

Разработанная система предоставляет пользователю возможность запустить свое задание как под управлением PBS, так и без него. Во втором случае программист должен перечислить узлы, на которых он хочет запустить задачу. Используя различные варианты запусков, можно получить реальные данные о влиянии конфигурации компьютера на программу.

Заключение

Разработанная графическая система автоматизирует основные операции, выполняемые пользователями суперкомпьютера. Система предоставляет исследователям возможность работать на кластере со своими пакетами, используя привычное программное обеспечение, а программистам оказывает помощь в разработке действительно эффективных параллельных программ.

Полагаем, что разработанный графический интерфейс будет способствовать более широкому использованию отечественных многопроцессорных вычислительных систем за счет существенного упрощения доступа пользователей к кластеру.

Список литературы

1. Кластерный комплекс МГУ «Ломоносов». Руководство пользователя [Электронный ресурс]. – 2012. – Режим доступа : http://parallel.ru/sites/default/files/cluster/ T500_user_guide-3.pdf. – Дата доступа : 29.06.2012.

2. Система управления кластером [Электронный ресурс]. – 2012. – Режим доступа : http://melkon.com.ua/ru/cms/sistema-upravleniya-klasterom.html. – Дата доступа : 29.06.2012.

3. Воеводин, В.В. Параллельные вычисления / В.В. Воеводин, Вл.В. Воеводин. – СПб. : БХВ-Петербург, 2002. – 608 с.

4. Кореньков, В.В. Архитектура и пути реализации системы локального мониторинга ресурсного центра / В.В. Кореньков, П.В. Дмитриенко // Электронный журнал «Системный анализ в науке и образовании». – 2011. – № 3.

5. Колисниченко, Д. Linux. От новичка к профессионалу / Д. Колисниченко. – СПб. : БХВ-Петербург, 2011. – 656 с.

6. Фуско, Дж. Linux. Руководство программиста / Дж. Фуско. – СПб. : Питер, 2011. – 448 с.

Поступила 06.07.2012

Белорусский государственный университет,
Минск, пр. Независимости, 4
e-mail: bouza@bsu.by
kondratjeva@bsu.by
vi0oss@gmail.com

M.K. Bouza, O.M. Kondratjeva, V.O. Shukelo

GRAPHICAL SYSTEM FOR PREPARE AND CONTROL OF USER TASKS ON A CLUSTER

An application for control of tasks on a supercomputer is developed. The application has a graphical interface and supports C/C++ realization on the basis of MPI, OpenMP and POSIX Threads technologies.

УДК 57:007:001.891.57

Т.В. Кирис, А.В. Тузиков

АЛГОРИТМ ПРЕДСКАЗАНИЯ ВЗАИМОДЕЙСТВИЯ БЕЛКОВ НА ОСНОВЕ СТРУКТУРНОЙ ГОМОЛОГИИ

Предлагается оригинальный алгоритм предсказания взаимодействия белков, основанный на структурной схожести с экспериментально определенным интерфейсом белкового комплекса, отобранным из базы интерфейсов белковых комплексов. Пара свободных белков выравнивается на интерфейс из библиотеки белок-белковых интерфейсов. Выравнивание выполняется с помощью метода динамического программирования путем максимизации корреляции между матрицами расстояний отрезков интерфейса и белка.

Введение

Задача предсказания взаимодействия белков является центральной в компьютерной биологии. Знания о взаимодействиях белков в организме являются необходимыми при создании лекарств и вакцин, а также для предсказания функций белков.

Несмотря на разнообразие алгоритмов белок-белкового докинга [1], задача предсказания комплекса белков по их свободным структурам остается нерешенной [2]. В большинстве алгоритмов белок-белкового докинга можно выделить три стадии: генерирование конформаций комплекса, их улучшение, выбор лучшей модели.

Генерация конформаций комплекса происходит либо на основе свободного докинга, когда структура одного белка перемещается вокруг структуры другого белка [3–6], либо на основе шаблонов (гомологии) [7–12]. Различные методы отличаются способами наложения структур белков на шаблоны, улучшения начальных моделей и целевой функцией. Значение методов на основе шаблонов возрастает с увеличением количества экспериментально полученных структур белок-белковых комплексов. Данные методы верно предсказывают комплекс белков при наличии схожего закристаллизованного комплекса.

Различные техники улучшения используют метод Монте-Карло, имитационный отжиг и др. [13–20]. Выбор лучшей модели осуществляется на основе целевой функции [2, 13, 18, 21–24]. Существуют целевые функции на основе статистических потенциалов [23, 25, 26], а также энергетические [2, 21, 27], которые могут учитывать электростатику, взаимодействия ван дер Ваальса, десольватации и др.

Поскольку белки при взаимодействии часто претерпевают конформационные изменения [28], которые включают изменения основной и боковых цепей, другим важным аспектом докинга является учет гибкости белка [15, 20, 29, 30]. Обзор существующих методов гибкого докинга содержится, например, в работах [29, 31].

Включение гибкости структур белка в алгоритмы докинга значительно усложняет этот процесс из-за увеличения числа степеней свободы, что приводит к значительному увеличению времени работы и большому числу ложноположительных результатов.

1. Определения и методы

Пусть данный белок A и интерфейс B представлены координатами C_α-атомов аминокислот в пространстве: $A=(a_1, a_2, ...,a_n)$, где a_i – координаты C_α-атома i-й аминокислоты, n – число аминокислотных остатков белка; интерфейс $B=(b_1, b_2, ..., b_m)$ состоит из m остатков.

В интерфейсе B выделены последовательные отрезки непрерывной первичной цепи B_j, $j=1,\ldots,k$, длиной не меньше l остатков:

$$B_1 = \{b_{11},..., b_{1j_1}\}, \ldots, B_k = \{b_{k1}, ..., b_{kj_k}\}.$$

Объединение таких отрезков будем обозначать $\overline{B} = \bigcup_{i=1}^{k} B_i$.

Матрица расстояний M_A белка A представляет собой матрицу $n \times n$, где элемент этой матрицы $m(i\,j)$ равен евклидову расстоянию в пространстве между атомами a_i и a_j.

Сходство двух множеств одинаковой мощности $A=\{a_1, a_2, ..., a_n\}$ и $C=\{c_1, c_2, ...,c_n\}$ будем оценивать через корреляцию их матриц расстояний и обозначать Correlation(A, C). Корреляция матриц M_A и M_C размерности $n \times n$ вычисляется по следующей формуле:

$$\text{Correlation}(A, C)=\frac{\sum_{i=1}^{n}\sum_{j=1}^{n} m_A(ij)\cdot m_C(ij)}{\sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n}(m_A(ij))^2}\cdot\sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n}(m_C(ij))^2}}. \quad (1)$$

Два белка или интерфейса называются пространственно выровненными, если они совмещены в пространстве с выбранным соответствием между остатками (C_α-атомами) совмещенных структур. Пара соответствующих остатков в разных структурах называется выровненной. Схожесть пространственно выровненных интерфейсов вычисляется с помощью меры TM-score [32]:

$$\text{TM-score}=\frac{1}{L_I}\sum_{i=1}^{L_M}\frac{1}{1+\left(d_i/d_0\right)^2}, \quad (2)$$

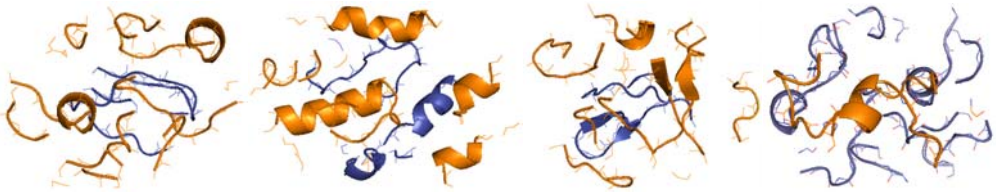
где L_I – количество остатков в интерфейсе; L_M – количество выровненных остатков; d_i – расстояние между i-й парой выровненных остатков. Коэффициент нормализации d_0 вычисляется по формуле

$$d_0=1{,}24\sqrt[3]{L_N-15}-1{,}8,$$

где L_N – количество остатков в наименьшей из двух структур.

Библиотека белок-белковых интерфейсов, использованная в данной работе для тестирования алгоритма, была построена на основе эталонного набора комплексов белков [33]. Остатки интерфейса определялись как остатки, у которых имеется по крайней мере один атом другого белка в комплексе на расстоянии не более 12Å. Библиотека состоит из 99 интерфейсов белок-белковых комплексов. Пример интерфейсов комплексов белков показан на рис. 1.

В среднем интерфейс белка состоит из 62 остатков (минимальный интерфейс содержит 17 остатков, максимальный – 219), содержит четыре отрезка непрерывной полипептидной цепи длиной более шести остатков (минимальный – 1, максимальный – 13), пять отрезков длиной более пяти остатков (минимальный – 1, максимальный – 14), пять отрезков длиной более четырех остатков (минимальный – 1, максимальный – 15) и шесть отрезков непрерывной полипептидной цепи длиной более трех остатков (минимальный – 1, максимальный – 20).



а) *б)* *в)* *г)*

Рис. 1. Интерфейсы комплексов белков: *а)* протеазы (оранжевая) и эглина С (синий), pdb-код 1cse; *б)* оксидоредуктазы (оранжевый) и адренодоксина (синий), pdb-код 1e6e; *в)* трипсина (оранжевый) и ингибитора (синий), pdb-код 1ppe; *г)* овомукоида (оранжевый) и протеазы (синий), pdb-код 1r0r. Непрерывные сегменты первичной последовательности, содержащие не меньше трех остатков, выделены жирными линиями

Качество предсказанного комплекса оценивается с помощью меры *i*-RMSD [1], которая используется в международном соревновании по предсказанию комплексов белков CAPRI [34] и вычисляется следующим образом: бóльшие по числу аминокислотных остатков белки (из двух белков в комплексе) предсказанного и экспериментально определенного комплекса совмещаются, а затем вычисляется среднеквадратичное отклонение (RMSD) между C_α-атомами интерфейсов меньших белков. Остатки интерфейса определяются как остатки, у которых имеется по крайней мере один атом другого белка в комплексе на расстоянии не более 6 Å.

2. Алгоритм

Идея предсказания взаимодействия белков A и B состоит в поиске белковых интерфейсов A' B' в базе данных, таких, что интерфейсы белков A и B подобны интерфейсам A' и B' соответственно (рис. 2). Предсказанный комплекс AB получается при наложении белка A на интерфейс A' и белка B на интерфейс B'. Такой комплекс будем называть комплексом AB, предсказанным на основе интерфейса A' B'.

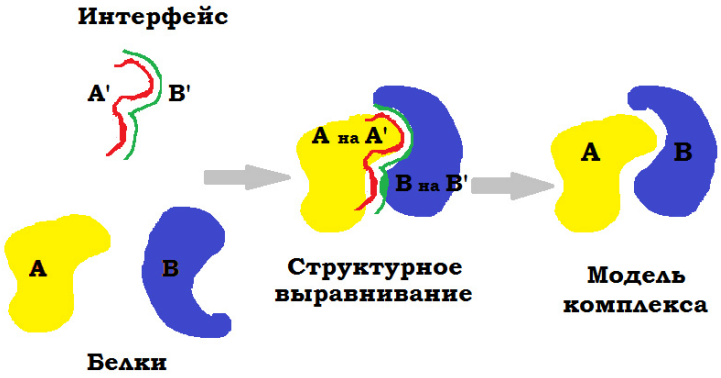


Рис. 2. Идея предсказания взаимодействия белковых комплексов на основе поиска гомологичных интерфейсов. Модель комплекса *AB*, предсказанная на основе интерфейса *A'B'*, получается путем выравнивания белка *A* на интерфейс *A'* и белка *B* на интерфейс *B'*

В интерфейсе A' выделяются последовательные непрерывные по первичной последовательности отрезки A'_i длиной не меньше l=3 остатков, $\overline{A'} = \bigcup_i A'_i$.

Для нахождения наилучшего выравнивания белка A с интерфейсом A' ищется такой отрезок $A_i \subset A$, что длина отрезка A_i равна длине отрезка A'_i и корреляция матриц расстояний между отрезками максимальна:

$$\text{Correlation}\,(M_{\bigcup_{i=1}^{k} A_i}, M_{\bigcup_{i=1}^{k} A'_i}) \to \max.$$

2.1. Выравнивание интерфейсов

Для нахождения отрезков A_j, j=1, 2,...,k, соответствующих отрезкам A'_j длины l_j, j=1, 2,...,k, с максимальной корреляцией их матриц расстояний используется метод динамического программирования (ДП). Идея ДП состоит в разбиении задачи на подзадачи, нахождении решений подзадач рекурсивно и, далее, использовании полученных решений подзадач для конструирования решения исходной задачи.

Для нахождения k отрезков A_j, j=1, 2,...,k, соответствующих отрезкам A'_j, j=1, 2,...,k, с максимальной корреляцией их матриц расстояний, будем последовательно искать сопоставле-

ние одного отрезка A'_1, двух отрезков A'_1, A'_2, трех отрезков A'_1, A'_2, A'_3 и т. д. с отрезками из A с максимальной корреляцией их матриц расстояний:

$$\text{Correlation}\ (M_{\bigcup_{j=1}^{i} A_j}, M_{\bigcup_{j=1}^{i} A'_j}) \to \max, i = 1,\ 2,..., k.$$

Отрезки A'_j, j=1, 2,...,k, будут сопоставляться с отрезками из A последовательно по первичной цепи, т. е. между отрезками A_j, j=1, 2,...,k, могут быть пропуски, но начало отрезка A_j расположено в линейной цепи после конца отрезка A_{j-1}, j=2,...,k. Отрезки A_j, j=1, 2,...,k, не пересекаются.

Введем функцию align(i, j), которая обозначает максимальную корреляцию при сопоставлении первых i отрезков A'_q, q=1,...,i, с отрезками из A так, что конец последнего отрезка A'_i сопоставляется с j-й аминокислотой a_j белка A. Сопоставление, на котором достигается эта корреляция, будет записываться в матрицу Path размерности $k \times n$ в позицию (i, j), $1 \le i \le k$, $1 \le j \le n$, т. е.

$$\text{Path}(i, j) = \bigcup_{k=1}^{i} A_k\ .$$

Для i=1 и $|A'_1| = l_1$ значение align(1, j)=Correlation(A'_1, (a_{j-l_1+1}, ... , a_j)), для i=2 значение align(2, j) равно максимальной корреляции Correlation($A'_1 \bigcup A'_2$), $A_1 \bigcup A_2$), когда конец отрезка A'_2 сопоставляется с j-й аминокислотой a_j белка A, а A_1 пробегает по всем возможным позициям от начала последовательности до пересечения с A_2.

Максимальной корреляцией отрезков A'_q, q=1, 2,...,k, с отрезками из A будет $\max_j$ align(k, j). Поскольку эта задача является NP-трудной [35], предлагается эвристика, которая находит субоптимальное решение. Рекуррентное соотношение для вычисления максимальной корреляции при сопоставлении первых i отрезков A'_q, q=1, 2,...,i, с отрезками из A определяется как максимальная корреляция найденного сопоставления i–1 отрезков и i-го отрезка A'_i:

$$\text{align}(i, j) = \max_{j_1 < j} \left\{ \text{Correlation}(\bigcup_{q=1}^{i} A'_q,\ \text{Path}(i-1, j_1)) \right\}. \qquad (3)$$

Введем массив Begin. Его i-й элемент Begin[i] – это номер остатка белка A, начиная с которого может сопоставляться конец i-го отрезка A'_i, $l_i = |A'_i|$, т. е.

$$\text{Begin}[1] = l_1,$$

а для i-го отрезка

$$\text{Begin}[i] = \text{Begin}[i-1] + l_i.$$

Алгоритм поиска сопоставления отрезков, при котором достигается максимальная корреляция их матриц расстояний, состоит из следующих шагов:

1. for (j = Begin[1]; $j \le n$; j++).
2. align(1, j) = Correlation (A'_1, (a_{j-l_1+1}, ... ,a_j)).
3. for(i = 2; $i \le k$; i++).

4. for (j = Begin[i]; $j \le n$; j++).

5. $\text{align}(i, j) = \max_{j_1 < j} \text{Correlation}(\bigcup_{q=1}^{i} A'_q, \text{Path}(i-1, j_1) \bigcup A_i)$.

6. $\text{Path}(i, j) = \text{Path}(i-1, \arg \text{align}(i, j)) \bigcup A_i$.

7. Path(k, arg max$_i$ align(k, i)).

Алгоритм определяет отрезки A_q, q=1, 2,...,k, соответствующие отрезкам A'_q, q=1, 2,...,k, корреляция матриц расстояний которых максимальна. В строке 2 вычисляются значения корреляции между отрезком A'_1 и отрезками из A. В следующем цикле по формуле (3) вычисляется значение корреляции сопоставления первых i отрезков A'_q, q=1, 2,...,i, и отрезков из A (строка 5) и определяются отрезки A_q, q=1, 2,...,i, соответствующие отрезкам A'_q, q=1, 2,...,i, на которых достигается максимальная корреляция (строка 6). В строке 7 определяется сопоставление между отрезками A'_q, q=1, 2,...,k, интерфейса A' и отрезками A_q, q=1, 2,...,k.

После того как найдено сопоставление между отрезками, вычисляется оптимальное движение, минимизирующее среднеквадратичное отклонение между всеми сопоставленными остатками.

Белок A выравнивается на интерфейс A' по найденному движению. Аналогичным образом находится сопоставление между отрезками B'_q, q=1, 2,...,k, интерфейса B' и отрезками B_q, q=1, 2,...,k, белка B, и белок B выравнивается на интерфейс B' по движению, минимизирующему среднеквадратичное отклонение между сопоставленными остатками.

2.2. Вычисление оптимального движения

Для двух равномощных множеств с известным соответствием между точками множеств существует оптимальное движение, минимизирующее среднеквадратичное отклонение между ними [36].

Пусть даны два равномощных множества точек в пространстве $P, M \subset \boldsymbol{R}^3$, их мощность $| P | = | M | = N$, и задано соответствие между точками множеств.

Вначале оба множества переносятся таким образом, чтобы их центроиды оказались в начале координат. Далее задача сводится к поиску матрицы поворота R, минимизирующей среднеквадратичное отклонение между соответствующими точками множеств:

$$\sum_{i=1}^{N} | p_i - R m_i |^2 \to \max .$$

Левую часть этого выражения можно переписать в виде

$$\sum_{i=1}^{N} | p_i |^2 - 2tr(R^T \sum_{i=1}^{N} p_i m_i^T) + \sum_{i=1}^{N} | m_i |^2 .$$

Матрица $K = \sum_{i=1}^{N} p_i m_i^T$. Задачу можно переформулировать так: дана матрица K, найти матрицу поворота R, такую, что

$$tr(R^T K) \to \max .$$

Если $K = U \Sigma V^T$ – разложение по сингулярным значениям, то значение $tr(R^T K)$ максимизируется матрицей $R = U V^T$.

Лемма [37]

Если K – действительная матрица $l \times n$, то существуют ортогональные матрицы

$$U = [u_1 \dots u_l] \subset \boldsymbol{R}^{l \times l};$$

$$V = [v_1 \ldots v_n] \subset \boldsymbol{R}^{n \times n},$$

такие, что

$$U^T K V = \Sigma = diag\,(\sigma_1,\ldots,\sigma_p) \in \boldsymbol{R}^{l \times n}, \quad (4)$$

где p=min(l, n) и $\sigma_1 \geq \ldots \geq \sigma_p \geq 0$*, или*

$$K = U \Sigma V^T. \quad (5)$$

Определим ортогональную матрицу $Z = U^T R^T V$. Получим $tr(R^T K) = tr(R^T V (V^T K U) U^T) =$ $= tr(R^T V \Sigma U^T) = tr(Z\Sigma) = \sum_{i=1}^{p} z_{ii} \sigma_i$. В связи с тем что Z – ортогональная матрица, выражение максимально при единичной матрице. Следовательно, $R = UV^T$.

Итак, для нахождения оптимального движения необходимо:

– перенести множества таким образом, чтобы их центроиды оказались в центре координат;

– вычислить матрицу корреляции между точками множества;

– разложить ее по сингулярным значениям;

– вычислить матрицу поворота.

3. Экспериментальные результаты

Предложенный алгоритм был протестирован на эталонном наборе комплексов белков [33]. В данный набор входят 99 белок-белковых комплексов, для которых известны структуры обоих белков в свободном состоянии. Тестирование было проведено на структурах белков в свободном состоянии.

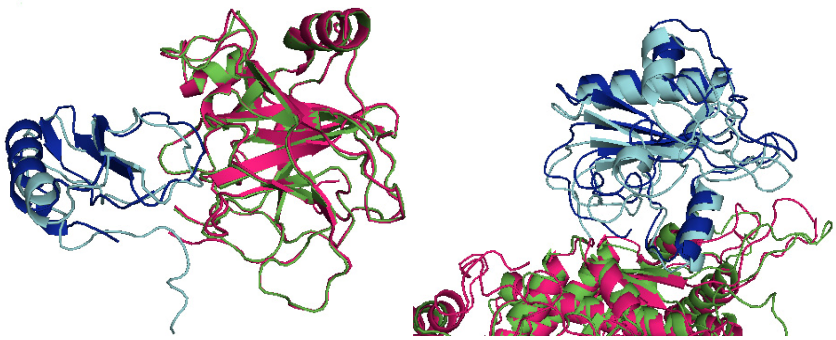
Пара свободных белков выравнивалась на каждый интерфейс из библиотеки белок-белковых интерфейсов. Схожесть выравнивания белков *A* и *B* на интерфейс $A'\,B'$ вычислялась как сумма TM-score (2) между интерфейсом A и A' и интерфейсом *B* и B'. Для каждой пары свободных белков строилось 10 моделей комплекса на основе интерфейсов с наибольшим значением TM-score.

Качество предсказанных комплексов оценивалось с помощью *i*-RMSD [2]. Согласно правилам международных соревнований по предсказанию комплексов белков CAPRI [34] комплекс считается «хорошо» предсказанным, если значение *i*-RMSD не привышает 5 Å хотя бы для одной из 10 построенных моделей.

Доля «хороших» предсказаний данного алгоритма на эталонном наборе комплексов белков [33] равняется 43, 42, 39 и 42 % при выделении в интерфейсе отрезков длиной 3, 4, 5 и 6 соответственно. Предлагается выбирать отрезки длиной 3, поскольку при выборе отрезков длиной *l* конформации отрезков интерфейса меньшей длины не принимаются во внимание и могут остаться без сопоставления. Доля «хороших» предсказаний алгоритма ZRANK [27] на схожем эталонном множестве [38] составляет 35 %.

Время предсказания комплекса белка из 300 аминокислот на основе одного интерфейса составляет в среднем 0,3 с на компьютере 800 MHz AMD Phenom(tm) II x4.

Алгоритм предсказывает правильный интерфейс в комплексе даже при большом (≥ 3 Å) изменении основной цепи интерфейса. На рис. 3 показаны примеры комплексов, построенных на основе интерфейсов с большим изменением основной цепи. Мера TM-score комплексов 1acb/1aro равнялась 0,99, 0,87 из 2,0 (сумме TM-score обоих белков комплекса); RMSD C_α-атомов между интерфейсами белков и интерфейсами, по которым они смоделированы, равно 3,6 и 5,8 Å соответственно. Несмотря на достаточно большую разницу конформаций петель в интерфейсах, структуры комплексов были предсказаны верно.



а) *б)*

Рис. 3. Закристаллизованные и предсказанные комплексы белков: *а)* эглина С (синий/голубой) и альфа-химотрипсина (розовый/зеленый), pdb-код комплекса 1acb; *б)* лизоцима (синий/голубой) и РНК полимеразы (розовый/зеленый), pdb-код комплекса 1aro

Заключение

Разработан алгоритм предсказания взаимодействия белков, основанный на структурной схожести с белок-белковым интерфейсом из базы данных. Пара свободных белков выравнивается на каждый интерфейс из библиотеки белок-белковых интерфейсов. Выравнивание выполняется с помощью метода динамического программирования путем максимизации корреляции между матрицами расстояний отрезков интерфейса и белка. Для каждой пары свободных белков строится 10 моделей комплекса на основе интерфейсов с наибольшей схожестью. Схожесть выравнивания белков на интерфейс вычисляется как сумма меры TM-score между интерфейсами.

Доля «хороших» предсказаний данного алгоритма на эталонном наборе комплексов белков [33] равняется 42 %. Разработанный алгоритм предсказывает правильный интерфейс в комплексе даже при некоторой гибкости конформации интерфейса.

Роль предложенного алгоритма предсказания взаимодействия белков по структурной гомологии с экспериментально определенными интерфейсами белков будет возрастать с увеличением количества экспериментально определенных комплексов белков.

Список литературы

1. Tramontano, A. The Ten Most Wanted Solutions in Protein Bioinformatics / A. Tramontano. – Boca Raton : Chapman and Hall/CRC, 2005. – 216 p.

2. Lensink, M.F. Docking and scoring protein interactions: CAPRI 2009 / M.F. Lensink, S.J. Wodak // Proteins. – 2010. – Vol. 78, № 15. – P. 3073-3084.

3. The performance of ZDOCK and ZRANK in rounds 6–11 of CAPRI / K. Wiehe [et al.] // Proteins. – 2007. – Vol. 69. – P. 719–725.

4. Schneidman-Duhovny, D. Automatic prediction of protein interactions with large scale motion / D. Schneidman-Duhovny, R. Nussinov, H. Wolfson // Proteins. – 2007. – Vol. 69. – P. 764–773.

5. PIPER: an FFT-based protein docking program with pairwise potentials / D. Kovakov [et al.] // Proteins. – 2006. – Vol. 65. – P. 392–406.

6. Kowalsman, N. Inherent limitations in protein-protein docking procedures / N. Kowalsman, M. Eisenstein // Bioinformatics. – 2007. – Vol. 23. – P. 421–426.

7. Lu, L. MULTIPROSPECTOR: an algorithm for the prediction of protein-protein interactions by multimeric threading / L. Lu, H. Lu, J. Skolnick // Proteins. – 2002. – Vol. 49. – P. 350–364.

8. Kundrotas, P.J. Predicting 3d structures of transient protein_protein complexes by homology / P.J. Kundrotas, E. Alexov // Biochimica et Biophysica Acta. – 2006. – Vol. 1764. – P. 1498–1511.

9. Fast and accurate modeling of protein-protein interactions by combining template-interface-based docking with flexible refinement / N. Tuncbag [et al.] // Proteins. – 2012. – Vol. 80. – P. 1239 – 1249.

10. A structural perspective on protein-protein interactions / R. Russel [et al.] // Curr Opin Struct Biol. – 2004. – Vol. 14. – P. 313–324.

11. Sinha, R. Docking by structural similarity at protein-protein interfaces / R. Sinha, P. Kundrotas, I. Vakser // Proteins. – 2010. – Vol. 78. – P. 3235–3241.

12. Vreven, T. Integrating atom-based and residue-based scoring functions for protein-protein docking / T. Vreven, H. Hwang, Z. Weng // Protein Science. – 2011. – Vol. 20. – P. 1576–1586.

13. Protein-protein docking with simultaneous optimization of rigid-body displacement and side-chain conformations / J.J. Gray [et al.] // J. Mol. Biol. – 2003. – Vol. 331, № 1. – P. 281–299.

14. FireDock: a web server for fast interaction re-nement in molecular docking / E. Maschiach [et al.] // Nucleic Acids Res. – 2008. – Vol. 36. – P. W229–W232.

15. Fernandez-Recio, J. ICM-DISCO docking by global energy minimization with fully flexible side-chains / J. Fernandez-Recio, M. Totrov, R. Abagyan // Proteins. – 2003. – Vol. 52. – P. 113–117.

16. Dominguez, C. HADDOCK: a protein-protein docking approach based on biochemical and/or biophysical information / C. Dominguez, R. Boelens, A. Bonvin // J. Am. Chem. Soc. – 2003. – Vol. 125. – P. 1731–1737.

17. Kozakov, D. Discrimination of near-native structures in protein-protein docking by testing the stability of local minima / D. Kozakov, O. Schueler-Furman, S. Vajda // Proteins. – 2008. – Vol. 72. – P. 993–1004.

18. Pierce, B. A combination of rescoring and refinement significantly improves protein docking performance / B. Pierce, Z. Weng // Proteins. – 2008. – Vol. 72. – P. 270–279.

19. Lorenzen, S. Monte Carlo refinement of rigid-body protein docking structures with back-bone displacement and side-chain optimization / S. Lorenzen, Y. Zhang // Proteins. – 2007. – Vol. 16. – P. 2716–2725.

20. Cavasotto, C. Representing receptor flexibility in ligand docking through relevant normal modes / C. Cavasotto, J. Kovacs, R. Abagyan // J. Am. Chem. Soc. – 2005.– Vol. 127. – P. 9632–9640.

21. Fiorucci, S. Binding site prediction and improved scoring during flexible protein-protein docking with ATTRACT / S. Fiorucci, M. Zacharias // Proteins. – 2010. – Vol. 78. – P. 3131–3139.

22. Kastritis, P. Are scoring functions in protein-protein docking ready to predict interactomes? Clues from a novel binding affinity benchmark / P. Kastritis, A. Bonvin // J. Proteome Res. – 2010. – Vol. 9. – P. 2216–2225.

23. A knowledge-based energy function for protein-ligand, protein-protein, and protein-DNA complexes / C. Zhang [et al.] // J. Med. Chem. – 2005. – Vol. 48. – P. 2325–2335.

24. Scoring by intermolecular pairwise propensities of exposed residues (SIPPER): A new efficient potential for protein-protein docking / C. Pons [et al.] // J. Chem. Inf. Model. – 2011. – Vol. 51. – P. 370–377.

25. Shen, M. Statistical potential for assessment and prediction of protein structures / M. Shen, A. Sali // Protein Science. – 2006. – Vol. 15. – P. 2507–2524.

26. Liu, S. DECK: Distance and environment-dependent, coarse-grained, knowledge-based potentials for protein-protein docking / S. Liu, I. Vakser // BMC Bioinformatics. – 2011. – Vol. 12. – P. 280–286.

27. Pierce, B. ZRANK: Reranking protein docking predictions with an optimized energy function / B. Pierce, Z. Weng // PROTEINS: Structure, Function, and Bioinformatics. – 2007. – Vol. 67. – P. 1078–1086.

28. Betts, M. An analysis of conformational changes on protein-protein association: implications for predictive docking / M. Betts, M. Sternberg // Protein Eng. – 1999. – Vol. 12. – P. 271–289.

29. Zacharias, M. Accounting for conformational changes during protein-protein docking / M. Zacharias // Curr Opin Struct Biol. – 2010. – Vol. 20, № 2. – P. 180–186.

30. Ding, F. Rapid flexible docking using a stochastic rotamer library of ligands / F. Ding, S. Yin, N.V. Dokholyan // J. Chem. Inf. Model. – 2010. – Vol. 50, № 9. – P. 1623–1632.

31. Principles of flexible protein-protein docking / N. Andrusier [et al.] // Proteins. – 2008. – Vol. 73, № 2. – P. 271–289.

32. Zhang, Y. Scoring function for automated assessment of protein structure template quality / Y. Zhang, J. Skolnick // Proteins. – 2004. – Vol. 57. – P. 702–710.

33. Dockground system of databases for protein recognition studies: Unbound structures for docking / Y. Gao [et al.] // Proteins. – 2007. – Vol. 69, № 4. – P. 845–851.

34. Assessment of CAPRI predictions in rounds 3–5 shows progress in docking procedures / R. Menndez [et al.] // Proteins. – 2005. – Vol. 60. – P. 150–169.

35. Kolodny, R. Approximate protein structural alignment in polynomial time / R. Kolodny, N. Linial // PNAS. – 2004. – Vol. 101. – P. 12201–12206.

36. Kabsch, W.A. A solution for the best rotation to relate two sets of vectors / W.A. Kabsch // Acta Cryst. – 1976. – Vol. 32. – P. 922–923.

37. Horn, R. Matrix Analysis / R. Horn, C. Johnson. – Cambridge University Press, 1985.

38. Protein-protein docking benchmark 2.0: an update / J. Mintseris [et al.] // Proteins. – 2005. – Vol. 60. – P. 214–216.

Поступила 10.09.12

Объединенный институт проблем
информатики НАН Беларуси,
Минск, Сурганова, 6
e-mail: tatsiana.kirys@gmail.com
tuzikov@newman.bas-net.by

T.V. Kirys, A.V. Tuzikov

ALGORITHM OF PROTEIN INTERACTION PREDICTION BASED ON STRUCTURAL HOMOLOGY

An original algorithm of protein interaction prediction is suggested. It is based on structural similarity with experimentally determined protein-protein interfaces, selected from interface database. Two proteins are structurally aligned on protein-protein interface. The alignment is performed by a dynamic programming algorithm that maximizes correlation of the protein distance matrices.

ИНФОРМАТИКА

2012 **октябрь-декабрь** **№ 4**

ПО МАТЕРИАЛАМ ПЯТОЙ МЕЖДУНАРОДНОЙ КОНФЕРЕНЦИИ «ТАНАЕВСКИЕ ЧТЕНИЯ»

УДК 681.32

О. Голами, Ю.Н. Сотсков

ЭВРИСТИЧЕСКИЕ АЛГОРИТМЫ ДЛЯ ПОСТРОЕНИЯ РАСПИСАНИЙ ОБСЛУЖИВАНИЯ ТРЕБОВАНИЙ С РАЗЛИЧНЫМИ МАРШРУТАМИ

Задача построения оптимального расписания обслуживания m приборами n требований с различными маршрутами является NP-трудной при любом m > 2 для всех регулярных критериев, рассматриваемых в теории расписаний. Для ее решения разработаны эвристические алгоритмы для трех регулярных критериев: минимизации общего времени обслуживания заданных требований; минимизации суммарного времени обслуживания n требований и минимизации суммарного запаздывания обслуживания n требований. Экспериментальное сравнение разработанных программ с одним из наиболее эффективных эвристических алгоритмов показало их превосходство по времени реализации и достаточно близкие результаты по качеству получаемых расписаний в случае, когда число m больше числа n. Неравенство m > n выполняется, в частности, для задач, возникающих при составлении оптимальных расписаний движения поездов по одноколейным железным дорогам.

Введение

В задачах оперативно-календарного планирования, возникающих на транспорте, в промышленности, в компьютерных технологиях, при планировании сервиса, в логистике и т. п., требуется выполнить множество работ на соответствующих рабочих местах (т. е. обслужить множество требований приборами) без прерывания операций и при соблюдении заданных временных и ресурсных ограничений. Временные ограничения указывают на то, что одни операции должны быть закончены раньше, чем начаты другие. Ресурсные ограничения задают недопустимость одновременного выполнения двух и более операций по обслуживанию требований одним и тем же прибором.

В подобного рода задачах требуется построить оптимальное расписание, т. е. определить, когда должна начаться каждая из заданных непрерываемых операций, чтобы соблюдались как временные, так и ресурсные ограничения, причем не убывающая от моментов завершения обслуживания требований целевая функция принимала бы минимальное значение. Такой критерий оптимальности расписаний принято называть регулярным [1, 2]. Используя терминологию теории расписаний [1], задачу можно сформулировать следующим образом.

Множество требований $J = \{J_1, J_2, ..., J_n\}$ необходимо обслужить приборами из множества $M = \{M_1, M_2, ..., M_m\}$ так, чтобы заданная целевая функция принимала наименьшее из возможных значений. Приборы множества M попарно различны по их назначению, т. е. для каждой операции по обслуживанию каждого требования заранее известен прибор из множества M, который предназначен для выполнения этой операции. Соответственно технологический маршрут $O_i = (O_{i1}, O_{i2}, ..., O_{in_i})$ обслуживания каждого требования $J_i \in J$ заранее определен, причем для разных требований из множества J могут быть априори заданы различные технологические маршруты. Время p_{ij} выполнения операции O_{ij} по обслуживанию требования $J_i \in J$ соответствующим прибором $M_v \in M$ известно до начала составления расписания обслуживания требований из множества J.

Для многостадийных обслуживающих систем [1] чаще других регулярных критериев рассматривается критерий $C_{\max}$, при котором требуется построить оптимальное по быстродействию расписание обслуживания требований J, т. е. расписание с наименьшей продолжительно-

стью. Иными словами, необходимо минимизировать общее время $C_{\max} = \max\{C_i : J_i \in J\}$ обслуживания требований из множества J [1, 2]. Здесь и далее C_i обозначает момент завершения обслуживания требования $J_i \in J$.

В англоязычной литературе [2] сформулированную задачу принято называть задачей job-shop (цех работ) с критерием $C_{\max}$. В общепринятой трехпозиционной классификации задач теории расписаний [2] такая задача обозначается $J \| C_{\max}$.

Для решения задачи $J \| C_{\max}$ разработаны как точные, так и эвристические алгоритмы. Среди точных алгоритмов следует отметить алгоритмы типа ветвей и границ (branch-and-bound), которые позволили найти оптимальные расписания для рекордных по размерности тестовых задач $J \| C_{\max}$ с доказательством оптимальности полученных решений [3, 4]. Следует, однако, признать, что размерности задач job-shop, для которых удается найти оптимальные расписания и доказать их оптимальность, все еще не столь велики, чтобы точные алгоритмы можно было широко использовать в практике оперативно-календарного планирования.

По-видимому, в ближайшем будущем «проблема размерности» так и не будет решена, поскольку задача $J \| C_{\max}$ является NP-трудной уже при $m > 2$ [1, 2], а в случае, когда допускаются повторения приборов в маршруте обслуживания требования, задача $J \| C_{\max}$ становится NP-трудной при $n = m = 3$ [5].

Среди эвристических алгоритмов, разработанных для решения задачи $J \| C_{\max}$, отметим алгоритмы сдвига узкого места (shifting bottleneck [6, 7]), алгоритмы моделируемого отжига (simulated annealing), алгоритмы поиска с запретами (tabu search [8−10]), а также генетические (genetic) алгоритмы [11].

1. Эвристические алгоритмы для решения задачи $J \| C_{\max}$

Разработанный специально для задачи $J \| C_{\max}$ алгоритм сдвига узкого места [6, 7] считается одним из наиболее эффективных (по качеству получаемых приближенных решений) алгоритмов для эвристического решения задачи $J \| C_{\max}$. В основе этого алгоритма лежит поиск наилучшего расписания обслуживания требований одним из приборов множества $M = \{M_1, M_2, ..., M_m\}$, который определяет узкое место (bottleneck) для всего процесса обслуживания требований множества J. Для определения такого прибора вначале вычисляется время t_i, необходимое для обслуживания каждого требования $J_i \in J$ без учета ресурсных ограничений (т. е. без учета недопустимости одновременного выполнения двух и более операций по обслуживанию требований одним и тем же прибором). Обозначим $t = \max\{t_i : J_i \in J\}$.

Минимальное из возможных запаздываний относительно времени t расписания, построенного с учетом ресурсных ограничений для одного (каждого) прибора из множества M, вычисляется в результате поиска последовательности выполняемых на приборе операций, при которой уменьшается максимальное запаздывание относительно t всех обслуживаемых на этом приборе требований. Затем строится оптимальное расписание (или расписание, близкое к оптимальному) обслуживания требований прибором, определяющим узкое место. При этом решается соответствующая задача $1 | r_i, prec | L_{\max}$ минимизации максимального временного смещения $L_{\max} = \max\{C_i - d_i : J_i \in J\}$ при обслуживании частично упорядоченного множества требований J_i, готовых к обслуживанию в моменты времени $r_i \geq 0$ и для которых установлены директивные сроки d_i обслуживания требований $J_i \in J$. Здесь и далее *prec* обозначает, что на множестве требований J априори задано отношение строгого порядка, которое определяет допустимые последовательности обслуживания требований (соответствующие временные ограничения). Поскольку и задача $1 | r_i, prec | L_{\max}$ является NP-трудной, для ее решения, как правило, применяется эвристическая процедура, которая является достаточно эффективной как по

временным затратам, так и по качеству получаемого эвристического решения задачи $1|r_i, prec|L_{\max}$.

В результате решения (точного или приближенного) задачи $1|r_i, prec|L_{\max}$ получаются новые временные ограничения, которые добавляются к исходным временным ограничениям задачи $J||C_{\max}$. В результате добавления новых временных ограничений задача $J||C_{\max}$ превращается в задачу $J|prec|C_{\max}$. Затем из множества еще не рассмотренных на данный момент приборов множества $M=\{M_1,M_2,...,M_m\}$ выделяется прибор, определяющий следующее узкое место, и для него аналогично решается (точно или приближенно) соответствующая задача $1|r_i, prec|L_{\max}$.

Описанный выше процесс продолжается до тех пор, пока либо не будет решена задача $1|r_i, prec|L_{\max}$ для каждого из приборов множества $M=\{M_1,M_2,...,M_m\}$, либо максимальное запаздывание относительно последнего полученного времени t не окажется равным нулю для всех нерассмотренных приборов из множества M.

Как показали проведенные на ноутбуке вычислительные эксперименты, применение описанного выше алгоритма сдвига узкого места для решения задач $J|r_i|C_{\max}$, для которых выполняется неравенство $m>n$, требует значительных затрат процессорного времени.

Алгоритм поиска с запретами представляет собой локальный поиск, широко используемый при решении многих оптимизационных задач [8]. Алгоритм поиска с запретами осуществляет локальный поиск с памятью, представляемой в виде «запрещенного списка» шагов (tabu list), которые были сделаны на предыдущих итерациях алгоритма локального поиска и которые запрещаются либо на всех последующих итерациях, либо на некоторых итерациях локального поиска. Запрещенный шаг из tabu list может быть снова разрешен (несмотря на то, что ранее он был запрещен), если будут соблюдены определенные условия (например, если решение, полученное на ранее запрещенном шаге, окажется лучше наилучшего из решений, полученных до текущей итерации алгоритма).

В алгоритме поиска с запретами структура и размеры рассматриваемых в локальном поиске окрестностей возможных решений играют существенную роль для достижения хорошего расписания в смысле его качества (близости к оптимальному расписанию) и времени работы компьютера при реализации алгоритма поиска с запретами. Так, при выборе малых по размеру окрестностей расписание, которое сильно отличается от исходного расписания, не может быть найдено алгоритмом поиска с запретами. Наоборот, выбор больших по размеру окрестностей в алгоритме поиска с запретами может привести к большим затратам процессорного времени для того, чтобы достичь хорошее по качеству расписание [9].

Комбинация алгоритма сдвига узкого места и алгоритма поиска с запретами была предложена и исследована в [10]. Алгоритм сдвига узкого места использовался для получения начального эвристического решения задачи job-shop, а алгоритм поиска с запретами использовался для того, чтобы минимизировать общее взвешенное запаздывание в задаче job-shop.

2. Быстрый эвристический алгоритм для решения задачи $J|r_i|C_{\max}$ при $m>n$

Цель данного исследования состояла в разработке и апробации на тестовых примерах эвристического алгоритма для решения задачи $J|r_i|C_{\max}$, в которой требования $J_i \in J$ готовы к обслуживанию в заранее заданные моменты времени $r_i \geq 0$. Рассмотренная во введении и разд. 1 задача $J||C_{\max}$ является частным случаем задачи $J|r_i|C_{\max}$, поскольку в задаче $J||C_{\max}$ все требования считаются готовыми к обслуживанию одновременно в момент времени $t=0$, т. е. $r_i=0$ для всех требований $J_i \in J$.

С учетом специфики планируемого применения алгоритма для решения практических задач $J|r_i|C_{\max}$ было необходимо, чтобы алгоритм не требовал много процессорного времени при решении задач job-shop с входными данными большой размерности при выполнении неравенства $m>n$ и при условии, что каждый прибор может выполнять не более одной операции в

каждом маршруте O_i по обслуживанию требования $J_i \in J$. Задача job-shop с указанными особенностями возникает при составлении оптимального расписания движения поездов по одноколейной железной дороге [12].

В Иране и других странах Ближнего Востока сети железных дорог в основном одноколейные. Сеть железных дорог называют одноколейной, если пара соседних станций может соединяться только одним железнодорожным путем. Будем называть железнодорожной секцией (или просто «секцией») участок железнодорожного пути, соединяющий две соседние станции. В соответствующей задаче job-shop требуется найти наилучшее расписание (т. е. расписание с наименьшей продолжительностью) движения поездов среди всех расписаний, для которых в любой момент времени по железнодорожному пути (секции), связывающему две соседние станции, может двигаться не более одного поезда.

Как было впервые замечено в статье [12], задача составления оптимального по быстродействию расписания движения поездов по одноколейной железной дороге эквивалентна задаче $J \,||\, C_{\max}$. При этом поезда и секции можно рассматривать как синонимы соответственно требований $J_i \in J$ и приборов $M_v \in M$ в эквивалентной задаче job-shop. Операция O_{ij} представляется как движение поезда $J_i \in J$ по железнодорожной секции $M_v \in M$ одноколейной железной дороги (здесь прибор M_v предназначен для выполнения операции O_{ij}).

Известно, что генетические алгоритмы [11] требуют слишком много процессорного времени для получения приемлемого по качеству расписания. Использование релаксации Лагранжа задачи $J \,|\, r_i \,|\, C_{\max}$ или алгоритма моделируемого отжига позволяет лишь незначительно уменьшить процессорное время, требуемое для получения приемлемого по качеству эвристического решения задачи job-shop большой размерности. Другие известные эвристические алгоритмы либо требуют слишком много процессорного времени, либо позволяют получать расписания для задачи job-shop, которые оказываются далекими (по значению целевой функции $C_{\max} = \max\{C_i : J_i \in J\}$) от оптимального расписания.

В процессе реализации на компьютере алгоритма сдвига узкого места [6, 7] оказалось, что требуется слишком много вычислений рекурсивных функций для получения промежуточных данных, используемых на каждой итерации алгоритма. Эти промежуточные данные включают директивные сроки обслуживания требований из множества J, время готовности к выполнению каждой операции, а также критический путь в орграфе $(Q, A, \varnothing)$ смешанного графа $G = (Q, A, E)$, представляющего исходные данные задачи job-shop [1, 3, 4]. Число итераций алгоритма сдвига узкого места асимптотически равно числу *m* обслуживающих приборов, поэтому время решения задачи job-shop растет с ростом *m*, и этот рост значителен для задач с числом приборов, существенно превышающим число обслуживаемых требований.

В смешанном графе $G = (Q, A, E)$ множество вершин Q представляет собой множество всех операций

$$Q = \{O, O_{1,1}, O_{1,2}, ..., O_{1n_1}, O_{2,1}, O_{2,2}, ..., O_{2n_2}, ..., O_{n1}, O_{n2}, ..., O_{nn_n} O_*\},$$

включающее также фиктивную начальную операцию O и фиктивную конечную операцию O_*. Множество дуг A определяет заданные временные ограничения, а множество ребер E – заданные ресурсные ограничения в задаче $J \,|\, r_i \,|\, C_{\max}$ [1, 2]. Каждой дуге $(O_{ij}, O_{uv}) \in A$ приписывается время (вес) p_{ij}, необходимое для выполнения операции $O_{ij} \in Q$ соответствующим прибором из множества M. Каждому ребру $[O_{ij}, O_{xz}] \in E$ приписывается пара времен (весов) p_{ij} и p_{xz}, необходимых для выполнения операции $O_{ij} \in Q$ и операции $O_{xz} \in Q$ соответственно. Дуге $(O_{in_i}, O_*) \in A$ приписывается время (вес) p_{in_i}, необходимое для выполнения операции $O_{in_i} \in Q$. Дуге $(O, O_{i1}) \in A$, $J_i \in J$, приписывается заданное время (вес) $r_i \geq 0$ готовности требования $J_i \in J$ к обслуживанию.

Наиболее раннее время начала (или наименьшее время готовности к выполнению) r_{ij} операции $O_{ij} \in Q$ может быть определено как наибольший суммарный вес пути из вершины $O \in Q$ в вершину O_{ij} в орграфе $(Q, A, \varnothing)$. Поскольку допустимый орграф $(Q, A, \varnothing)$ не должен содержать контуров, то все времена r_{ij} являются конечными и могут быть определены методом критического пути за линейное от суммы $|Q|+|A|$ число элементарных действий.

Наименьшее время готовности к выполнению (shortest release time SRT) операции $O_{ij} \in Q$ используется как приоритет (SRT-приоритет) в разработанном SRT-алгоритме для приближенного решения задачи $J \,|\, r_i \,|\, C_{\max}$. В отличие от алгоритма сдвига узкого места [6, 7], в котором на каждой итерации решается задача $1 \,|\, r_i, prec \,|\, L_{\max}$ с одним прибором из множества M, определяющим узкое место, на каждой итерации SRT-алгоритма решается задача по обслуживанию одного требования, которое является критическим на данной итерации алгоритма. Иными словами, для всех операций по обслуживанию критического требования определяется порядок их выполнения относительно операций по обслуживанию других требований из множества J на соответствующих приборах из множества M.

На первой итерации SRT-алгоритма критическим является требование $J_i \in J$, для которого последняя операция O_{in_i} в маршруте O_i является предпоследней операцией критического пути в орграфе $(Q, A, \varnothing)$ (или одного из критических путей, если таких путей в орграфе $(Q, A, \varnothing)$ несколько). На первой итерации SRT-алгоритма выполняются следующие два шага:

Шаг 1. Наименьшее время готовности r_{ij} для каждой операции $O_{ij} \in Q$ вычисляется согласно рекурсивной функции

$$r_{ij} = r_{i,j-1} + p_{i,j-1}.$$

При этом наименьшее время готовности начальной операции O полагается равным нулю. Для вычисления всех времен r_{ij}, $O_{ij} \in Q$, $J_i \in J$, требуется линейное время от числа операций $|Q|$.

Шаг 2. Начиная с первой операции O_{i1} в маршруте O_i требования $J_i \in J$, SRT-алгоритм определяет прибор $M_v \in M$, необходимый для выполнения операции O_{i1}, а также множество всех других операций $O(v) \subset Q$, которые выполняются прибором $M_v \in M$. Все ребра $[O_{i1}, O_{jk}] \in E$, для которых выполняется включение $O_{jk} \in Q(v)$, ориентируются согласно SRT-приоритету: если $r_{i1} \le r_{jk}$, то дуга (O_{i1}, O_{jk}) добавляется в орграфе $(Q, A, \varnothing)$; в противном случае симметрическая дуга (O_{jk}, O_{i1}) добавляется в орграфе $(Q, A, \varnothing)$. В обоих случаях ребро $[O_{i1}, O_{jk}]$ удаляется из смешанного графа G. Описанная процедура повторяется для операции O_{i2}, затем для операции O_{i3} и так далее до последней операции O_{in_i} в маршруте O_i обслуживания критического требования $J_i \in J$. В результате таких замен ребер из множества E на дуги смешанный граф $G = (Q, A, E)$ преобразуется в смешанный граф $G_i = (Q, A \cup A_i, E \setminus E_i)$. Конец шага 2.

На второй итерации SRT-алгоритма рассматривается «второе критическое» требование, т. е. требование $J_u \in J \setminus \{J_i\}$, для которого операция O_{un_u} в маршруте O_u имеет наибольшее время завершения в орграфе $(Q, A \cup A_i, \varnothing)$ среди всех требований из множества $J \setminus \{J_i\}$. На второй итерации требования $J_i \in J$ шаги 1 и 2 выполняются для требования J_u, орграфа $(Q, A \cup A_i, \varnothing)$ и смешанного графа G_i соответственно. При необходимости наименьшее время r_{ij} готовности операции $O_{ij} \in Q$ должно быть повторно рассчитано по следующей рекурсивной формуле:

$$r_{uj} := \max_{(O_{kl}, O_{uj}) \in A \cup A_i} \{r_{uj}, r_{kl} + p_{kl}\}.$$

Описанный процесс повторяется для «третьего критического» требования, затем для «четвертого критического» требования и так далее, пока все требования множества J не будут рассмотрены. В результате смешанный граф G превращается в орграф $G_d = (Q, A \cup A_d, \varnothing)$, здесь J_d – то требование из множества J, которое было рассмотрено на последней n-й итерации SRT-алгоритма.

Нетрудно убедиться в том, что использование SRT-приоритета при ориентации ребер E смешанного графа $G = (Q, A, E)$ не может привести к возникновению контура ни в одном из орграфов $(Q, A \cup A_i, \varnothing)$, …, $(Q, A \cup A_d, \varnothing)$, построенных на каждой итерации SRT-алгоритма. Полученный бесконтурный орграф G_d однозначно определяет активное (semi-active) расписание [2], которое может быть построено методом критического пути за время $O(n+|A_d|)$.

Помимо SRT-алгоритма авторами был разработан SCT-алгоритм, отличающийся от SRT-алгоритма только использованием в алгоритме (на шаге 2) другого правила приоритета операций при выборе порядка их выполнения в искомом расписании. Вместо SRT-приоритета в SCT-алго-ритме используется SCT-приоритет (от англ. Shortest Completion Time, т. е. кратчайшее время завершения). Первая итерация SCT-алгоритма начинается с операции O_{i1} в маршруте O_i критического требования $J_i \in J$. Согласно SCT-приоритету на шаге 2 выполняется следующая замена ребра на дугу. Если $c_{i1} \le c_{jk}$, то дуга (O_{i1}, O_{jk}) добавляется к орграфу $(Q, A, \varnothing)$. В противном случае симметрическая дуга (O_{jk}, O_{i1}) добавляется к орграфу $(Q, A, \varnothing)$.

Аналогично был разработан SDD-алгоритм, который отличается от SRT-алгоритма и SCT-алгоритма другим правилом приоритета операций. В SDD-алгоритме используется SDD-приоритет (от англ. Shortest Due Date, т. е. кратчайший директивный срок). Первая итерация SDD-алгоритма начинается с операции O_{i1} в маршруте O_i критического требования $J_i \in J$. Согласно SDD-приоритету если $d_{i1} \le d_{jk}$, то дуга (O_{i1}, O_{jk}) добавляется к орграфу $(Q, A, \varnothing)$. В противном случае симметрическая дуга (O_{jk}, O_{i1}) добавляется к орграфу $(Q, A, \varnothing)$.

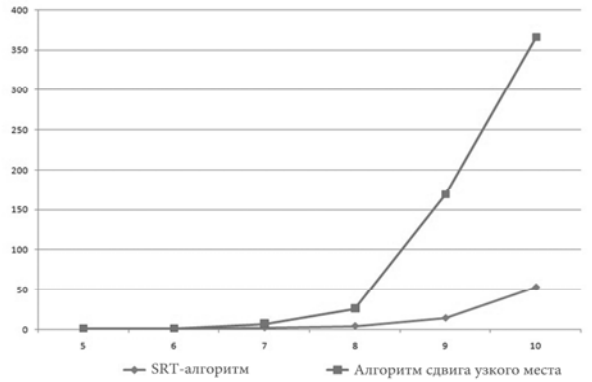
3. Результаты вычислительного эксперимента по решению случайно сгенерированных задач $J \mid r_i \mid C_{\max}$

Разработанные алгоритмы, а также алгоритм сдвига узкого места [3, 4] были закодированы на алгоритмическом языке Delphi. Для того чтобы проверить их эффективность, использовался ноутбук следующей конфигурации: Интел®, coreTM 2 Duo, CPU T6400, 2.00 GHz, 2GB Internal Memory, Windows 7, Ultimate 32 bit.

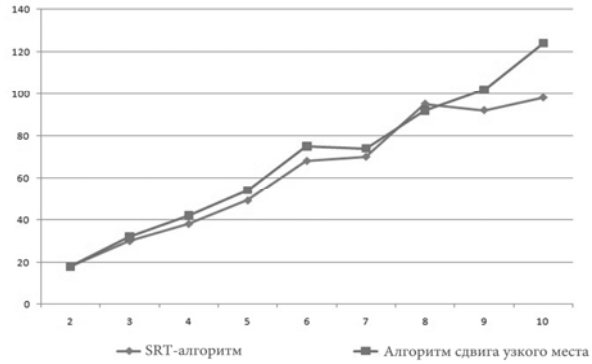
Вначале было проведено сравнение процессорного времени, которое требовалось SRT-алгоритму и алгоритму сдвига узкого места для эвристического решения одних и тех же серий случайно сгенерированных примеров задачи $J \mid\mid C_{\max}$ различной размерности $n \times m$, где $n = m$ (например, 6×6 или 10×10). Сравнивались значения критерия $C_{\max}$ для расписаний, полученных обоими алгоритмами для одинаковых входных данных. Результаты проведенного вычислительного эксперимента показаны на рис. 1. По оси абсцисс графика указаны совпадающие значения $n = m$, а по оси ординат − среднее процессорное время для серии тестовых примеров в секундах (рис. 1, *а*) и среднее значение целевой функции $C_{\max} = \max\{C_i : J_i \in J\}$ (рис. 1, *б*).

В проведенном эксперименте также оценивалось влияние добавления приборов или требований на процессорное время, необходимое для приближенного решения задачи $J \mid r_i \mid C_{\max}$ SRT-алгоритмом. Вначале решалась задача $J \mid r_i \mid C_{\max}$ с размерностью $(n = 5) \times (5 = m)$ и затем добавлялись новые требования, чтобы экспериментально оценить рост процессорного времени, необходимого для реализации SRT-алгоритма. Затем фиксировалось число требований и до-

бавлялись новые приборы в множество M для оценки влияния роста размерности задачи на процессорное время, необходимое для реализации SRT-алгоритма.



а)



б)

Рис. 1. Сравнение SRT-алгоритма и алгоритма сдвига узкого места: *а)* по процессорному времени; *б)* по значениям критерия $C_{\max}$

Как следует из эксперимента, результаты которого представлены на рис. 2, для приближенного решения задачи $J \mid r_i \mid C_{\max}$ размерности 5×20 SRT-алгоритм требует 10 с процессорного времени, а для решения задачи размерности 20×5 – 1929 с. Такое различие процессорного времени объясняется тем, что добавление требований для обслуживания в задаче job-shop влечет значительное увеличению количества вычислений приоритета в SRT-алгоритме.

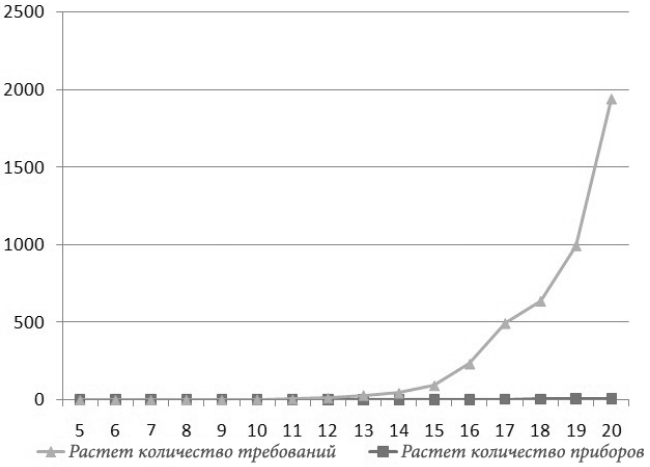


Рис. 2. Рост процессорного времени, необходимого для решения задачи $J \mid\mid C_{\max}$ SRT-алгоритмом при увеличении числа требований (или приборов)

SRT-алгоритм показывает лучшие вычислительные результаты, когда число приборов превышает число требований. Поскольку значения целевой функции $C_{\max} = \max\{C_i : J_i \in J\}$ отличаются незначительно для сравниваемых алгоритмов (алгоритма сдвига узкого места и SRT-алгоритма), можно утверждать, что SRT-алгоритм является более предпочтительным по сравнению с алгоритмом сдвига узкого места [6, 7] при решении задачи $J \,||\, C_{\max}$, для которой выполняется неравенство $m > n$.

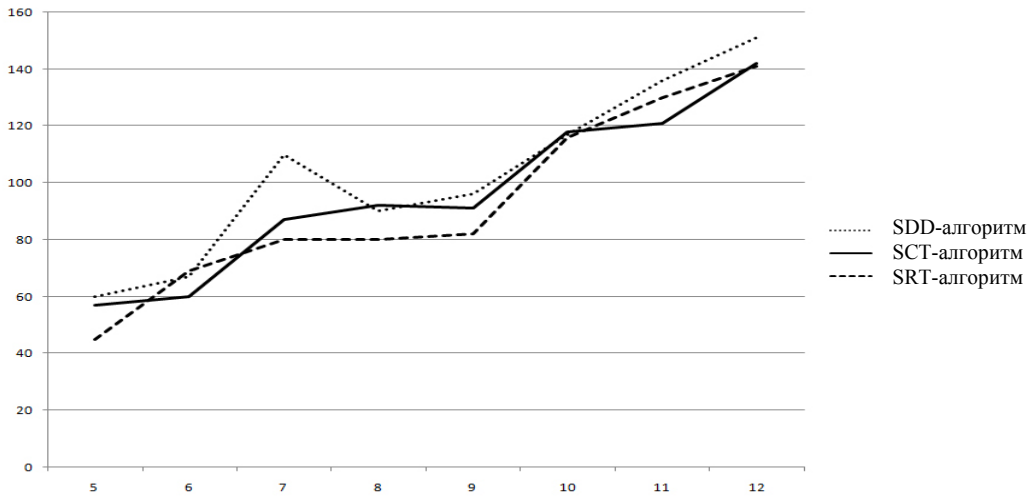


Рис. 3. Значения целевой функции $C_{\max} = \max\{C_i : J_i \in J\}$ для расписаний, полученных в результате применения SRT-, SCT- и SDD-алгоритмов

Были проведены вычислительные эксперименты и для сравнения качества расписаний (относительно критерия $C_{\max}$), полученных SRT-, SCT- и SDD-алгоритмами. Результаты этих экспериментов представлены на рис. 3. Из рисунка видно, что с помощью SRT-алгоритма строятся расписания с меньшими значениями целевой функции $C_{\max} = \max\{C_i : J_i \in J\}$ по сравнению с расписаниями, которые строит SDD-алгоритм. Качество расписаний, которые строят SRT- и SCT-алгоритмы, отличается незначительно. В большинстве проведенных экспериментов SRT-алгоритм незначительно превосходил SCT-алгоритм по значениям целевой функции $C_{\max} = \max\{C_i : J_i \in J\}$. В остальных экспериментах, наоборот, SCT-алгоритм незначительно превосходил по качеству решений SRT-алгоритм.

4. Результаты вычислительных экспериментов по решению задач $J \,|\, r_i \,|\, \sum C_i$ и $J \,|\, r_i \,|\, \sum T_i$

При оптимизации расписания движения поездов по одноколейным железным дорогам важными критериями, наряду с критерием $C_{\max}$, являются критерий $\sum C_i = \sum_{i=1}^{n} C_i$ минимизации суммарного времени обслуживания заданных требований и критерий $\sum T_i = \sum_{i=1}^{n} T_i$ минимизации суммарного запаздывания обслуживания требований. Запаздывание T_i обслуживания требования $J_i \in J$ определяется по формуле $T_i = \max\{0, C_i - d_i\}$, где d_i – заданный директивный срок, к которому желательно обслужить требование $J_i \in J$.

В задаче $J \,|\, r_i \,|\, \sum T_i$, которая возникает при планировании движения поездов, момент времени $r_i \geq 0$ готовности требования $J_i \in J$ к обслуживанию интерпретируется как планируемое (наиболее раннее) время отправления поезда $J_i \in J$ с первой станции в его маршруте, а директивный срок d_i – как планируемое время прибытия поезда $J_i \in J$ на конечную станцию.

В связи с важностью указанных критериев для оптимального планирования железнодорожных перевозок было проведено экспериментальное сравнение значений критериев $\sum C_i$ и $\sum T_i$ для расписаний, построенных SRT-, SCT- и SDD-алгоритмами.

Результаты сравнения построенных расписаний по значениям целевой функции $\sum C_i = \sum_{i=1}^{n} C_i$ показали (рис. 4), что SCT-алгоритм существенно превосходит SDD-алгоритм по качеству приближенных решений задачи $J \mid r_i \mid \sum C_i$ и незначительно превосходит SRT-алгоритм.

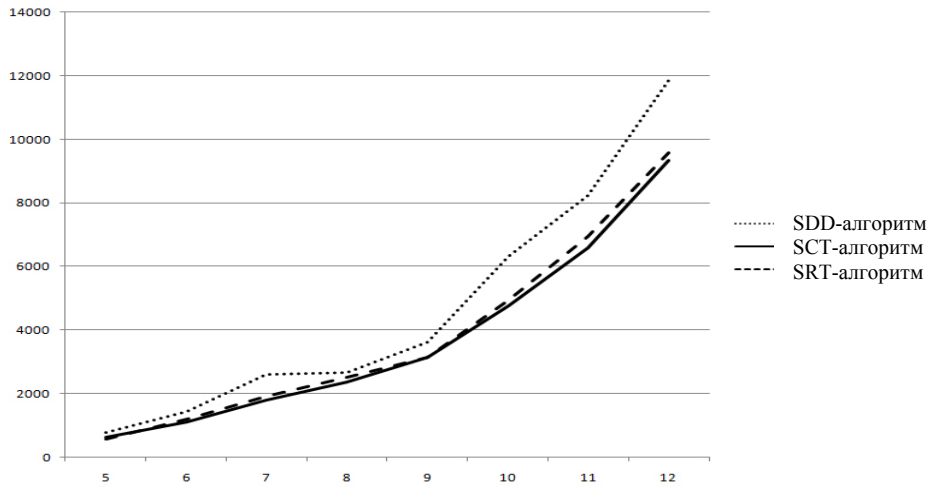


Рис. 4. Значения целевой функции $\sum C_i = \sum_{i=1}^{n} C_i$ для расписаний, полученных в результате применения SRT-, SCT- и SDD-алгоритмов

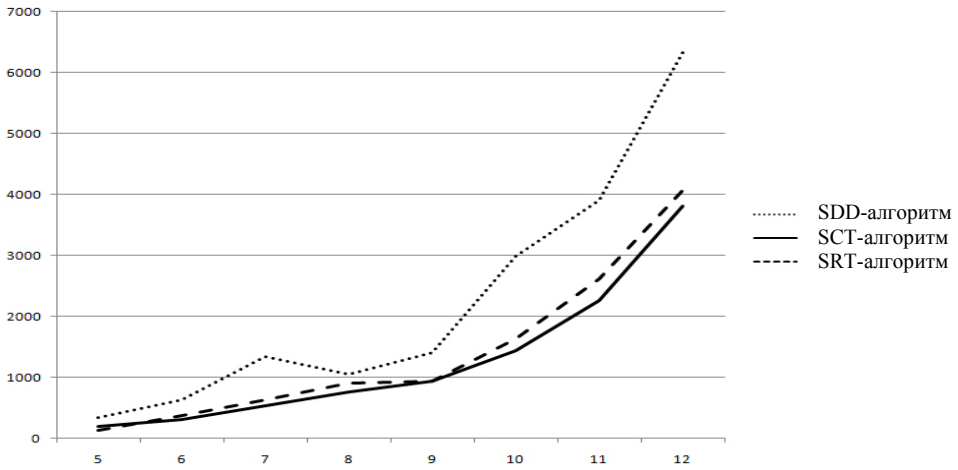


Рис. 5. Значения целевой функции $T_i = \max\{0, C_i - d_i\}$ для расписаний, полученных в результате применения SRT-, SCT- и SDD-алгоритмов

Результаты сравнения построенных расписаний по значениям целевой функции $\sum T_i = \sum_{i=1}^{n} T_i$ показали (рис. 5), что SCT-алгоритм незначительно превосходит SRT-алгоритм и существенно превосходит SDD-алгоритм по качеству получаемых приближенных решений задачи $J \mid r_i \mid \sum T_i$.

Заключение

Разработаны эвристические алгоритмы (и программы на алгоритмическом языке Delphi) для решения задачи job-shop построения оптимального расписания обслуживания m приборами n требований с различными маршрутами для следующих регулярных критериев оптимальности: минимизации общего времени обслуживания требований $C_{\max} = \max\{C_i : J_i \in J\}$; минимизации суммарного времени обслуживания требований $\sum C_i = \sum_{i=1}^{n} C_i$ и минимизации суммарного запаздывания обслуживания требований $\sum T_i = \sum_{i=1}^{n} T_i$. Исследованные критерии являются важными при решении практических задач составления оптимальных расписаний движения поездов по одноколейным железным дорогам. Особенности такого класса практических задач состоят в том, что число m обслуживающих приборов (железнодорожных станций), как правило, больше числа n обслуживаемых требований (поездов), причем каждый прибор (станция) встречается в маршруте обслуживания каждого требования (т. е. в маршруте движения поезда) не более одного раза.

Были проведены вычислительные эксперименты на ноутбуке, которые показали превосходство разработанных алгоритмов для указанного класса задач job-shop по времени реализации по сравнению с широко известным алгоритмом сдвига узкого места. По качеству получаемых расписаний были получены достаточно близкие результаты.

Статистический анализ вычислительных результатов показал, что хотя алгоритм сдвига узкого места значительно сложнее разработанных эвристик, он не имеет существенного превосходства по качеству получаемых решений по сравнению с более простыми и, следовательно, менее трудоемкими алгоритмами в классе задач job-shop при выполнении условия $m > n$. Поэтому при составлении оптимальных расписаний движения поездов по одноколейным железным дорогам целесообразно использовать алгоритмы и программы, описанные в разд. 2–4.

При экспериментальном сравнении программных реализаций разработанных алгоритмов между собой оказалось, что один из трех алгоритмов, а именно SCT-алгоритм, существенно превосходит SDD-алгоритм по качеству решений для двух из трех рассмотренных критериев (для задач $J \mid r_i \mid \sum C_i$ и $J \mid r_i \mid \sum T_i$) и незначительно превосходит SRT-алгоритм для тех же критериев. Процессорное время, которое требуется для реализации SCT-, SDD- и SRT-алгоритмов, практически одинаково.

Список литературы

1. Танаев, В.С. Теория расписаний. Многостадийные системы / В.С. Танаев, Ю.Н. Сотсков, В.А. Струсевич. – М. : Наука. Гл. ред. физ.-мат. лит, 1989. – 328 с.

2. Sequencing and scheduling: Algorithms and complexity / E.L. Lawler [et al.] // Handbooks in Operations Research and Management Science. Logistics of Production and Inventory. – North-Holland, 1993. – P. 445–522.

3. Carlier, J. An algorithm for solving the job shop problem / J. Carlier, E. Pinson // Management Science. – 1989. – Vol. 35, № 2. – P. 164–176.

4. Brucker, P. The job-shop problem and immediate selection / P. Brucker, B. Jurusch, A. Kramer // Annals of Operations Research. – 1994. – Vol. 50. – P. 73–114.

5. Sotskov, Yu.N. NP-hardness of shop-scheduling problems with three jobs / Yu.N. Sotskov, N.V. Shakhlevich // Discrete Applied Mathematics. – 1995. – Vol. 59, № 3. – P. 237–266.

6. Adams, J. The shifting bottleneck procedure for jobshop scheduling / J. Adams, E. Balas, D. Zawack // Management Science. – 1988. – Vol. 34, № 3. – P. 391–401.

7. Balas, E. Guided local search with shifting bottleneck job shop scheduling / E. Balas, A. Vazacopoulos // Management Science. – 1998. – Vol. 44, № 2. – P. 262–275.

8. Glover, F. Tabu search – part 1 / F. Glover // ORSA Journal on Computing. – 1989. – Vol. 1, № 2. – P. 190–206.

9. Dell'Amico, M. Applying tabu search to the job-shop scheduling problem / M. Dell'Amico, M. Trubian // Annals of Operations Research. – 1993. – Vol. 41. – P. 231–252.

10. Britto, R.A. Combined approach of the shifting bottleneck and tabu search heuristics for minimizing total weighted tardiness in job shop scheduling problems / R.A. Britto, G.M. Delgadillo, J.P. Villalobos // Third International Conference on Production Research (ICPR-AM06). – Brazil, 2006.

11. Werner, F. Genetic algorithms for shop scheduling problems: a survey / F. Werner. – Germany, Magdeburg, 2011. – (Preprint 31/11, FMA. Otto-von-Guericke-University).

12. Szpigel, B. Optimal train scheduling on a single line railway / B. Szpigel // Operations Research. – 1973. – Vol. 72. – P. 344–351.

Поступила 23.08.12

Объединенный институт проблем
информатики НАН Беларуси,
Минск, Сурганова, 6
e-mail: gholami_iran@yahoo.com
sotskov@newman.bas-net.by

O. Gholami, Yu.N. Sotskov

HEURISTIC ALGORITHMS FOR JOB SHOP SCHEDULING

As a job-shop problem is NP-hard, one cannot design an optimal schedule for a large job-shop problem instance in a reasonable time. Moreover, many known heuristic algorithms need a lot of CPU time to construct a sufficiently good schedule. In this paper, we present fast heuristics and computationally compare them with one of the most famous heuristics for a job-shop scheduling problem.

ИНФОРМАТИКА

2012 **октябрь-декабрь** **№ 4**

УДК 519.8

Н.Н. Гущинский

ДЕКОМПОЗИЦИОННЫЙ ПОДХОД К ОПТИМИЗАЦИИ ПАРАМЕТРОВ ДУГ ПОСЛЕДОВАТЕЛЬНО-ПАРАЛЛЕЛЬНЫХ ОРГРАФОВ

Рассматривается задача оптимизации параметров технических систем, структура которых уже определена и может быть представлена последовательно-параллельными орграфами. Предполагается, что совокупность искомых проектных параметров системы может быть приписана дугам орграфа, а условия функционирования системы могут быть заданы ограничениями (равенствами и неравенствами) на значения некоторых функций, монотонно зависящих от искомых параметров. В качестве критерия выбора оптимальных решений рассматривается минимизация квазисепарабельной функции, зависящей как от параметров дуг орграфа, так и от их суммарных значений.

Введение

В комплексе оптимизационных задач, возникающих в системах автоматизированного проектирования, планирования и управления, значительное место занимают задачи, которые могут быть достаточно адекватно сформулированы в терминах оптимизации параметров дуг сети при надлежащем выборе целевой функции и ограничений. В частности, математическими моделями такого вида могут быть представлены задачи оптимального выбора параметров многозвенных механических трансмиссий различной структуры на начальном этапе их проектирования [1]. Эти модели могут быть как детерминированными, в которых случайный характер различных проектных и эксплуатационных факторов учитывается опосредованно, так и вероятностными, предполагающими прямой учет имеющейся информации о случайном характере всех либо части этих факторов. Ниже рассматривается один из классов таких задач.

1. Постановка задачи

Рассматривается класс *последовательно-параллельных орграфов* (ПП-орграфов), который является подмножеством класса ориентированных мультиграфов с одним источником и одним стоком. Этот класс может быть определен следующим образом [4]. Орграф G, состоящий из двух вершин s и t, которые соединены дугой (s, t), является *базовым*, или *минимальным*, ПП-орграфом. Вершина s называется *источником*, а вершина t – *стоком* ПП-орграфа G соответственно. В дальнейшем будем обозначать их через $s(G)$ и $t(G)$. *Составной* ПП-орграф может быть получен из двух ПП-орграфов G_1 и G_2 с помощью следующих двух операций композиции. *Последовательная* композиция заключается в отождествлении стока $t(G_1)$ орграфа G_1 с источником $s(G_2)$ орграфа G_2 ($s(G)=s(G_1)$, $t(G)=t(G_2)$), а *параллельная* композиция – в отождествлении источников $s(G_1)$ и $s(G_2)$ и стоков $t(G_1)$ и $t(G_2)$ ($s(G)=s(G_1)=s(G_2)$, $t(G)=t(G_1)=t(G_2)$).

ПП-орграфы принято представлять в виде *двоичного декомпозиционного дерева* [4], в котором листьями (висячими вершинами) являются дуги исходного графа G, остальными вершинами – операции последовательной (+) и параллельной (//) композиции, а ребра соединяют подграфы, участвующие в композиции соответствующего подграфа более высокого уровня. Данное представление не единственное, для одного и того же ПП-графа может существовать несколько его различных декомпозиционных деревьев. В [5] предложен алгоритм, по которому можно найти некоторое такое представление за линейное время от числа дуг.

В настоящей работе, как и в [3], используется метод представления ПП-орграфов, основанный на операциях последовательной и параллельной композиции над несколькими графами. При последовательной композиции ПП-орграфов G_1, G_2, …, G_m последовательно для $i = 1, 2, \ldots, m-1$ отождествляется сток $t(G_i)$ орграфа G_i с источником $s(G_{i+1})$ орграфа G_{i+1} ($s(G)=s(G_1)$, $t(G)=t(G_m)$), а при их параллельной композиции отождествляются между собой источники $s(G_i)$ ($s(G)=s(G_1)=s(G_2)=\ldots=s(G_{m-1})=s(G_m)$) и стоки $t(G_i)$ ($t(G)=t(G_1)=t(G_2)=\ldots=t(G_{m-1})=t(G_m)$) соответственно. В этом случае ПП-орграф представляется единственным

образом его *декомпозиционным деревом*, в котором по-прежнему листьями являются дуги исходного графа G, остальными вершинами – операции последовательной и параллельной композиции, а ребра соединяют подграфы, участвующие в композиции соответствующего подграфа более высокого уровня. Для этого дерева характерным является то, что на каждом *уровне* его вершин (длине пути из корня дерева) используется одна и та же операция композиции (последовательной или параллельной).

Пусть R_1, R_2 и R_3 – некоторые подмножества действительных чисел. На множествах R_1 и R_2 заданы бинарные операции $\bullet$ и $\oplus$ соответственно, а на множестве R_3 – бинарные операции $\otimes$ и $\circ$. Предполагается, что все операции обладают свойствами *ассоциативности* и *коммутативности* и не убывают по своим операндам, т. е. $a \bullet b \le c \bullet b$ и $a \bullet b \le a \bullet d$, если $a < c$ и $b < d$. Будем предполагать также, что операции $\bullet$ и $\oplus$ образуют *абелеву группу* на соответствующем подмножестве. В этом случае в множествах R_1 и R_2 существуют *нейтральные элементы* $1(\bullet)$ и $1(\oplus)$ (называемые также *единицами*), для которых $a \bullet 1(\bullet) = a$ и $b \oplus 1(\oplus) = b$ для любых элементов a из R_1 и b из R_2, а также существуют такие *обратные элементы* a^{-1} и b^{-1}, что $a \bullet a^{-1} = 1(\bullet)$ и $b \oplus b^{-1} = 1(\oplus)$. Заметим, что при сделанном предположении операции $\bullet$ и $\oplus$ строго возрастают в силу единственности обратного элемента в абелевой группе.

Задан ПП-орграф $G=(V,E)$ с множеством вершин V и множеством дуг E с источником s и стоком t. Каждой вершине $v \in V$ сопоставлена функция $M_v: R_1 \to R_2$, а дуге $e \in E$ – отрезки $[\underline{x}(e), \overline{x}(e)]$ из R_1 и $[\underline{p}(e), \overline{p}(e)]$ из R_3, а также неубывающая по последнему аргументу функция $M_e: R_1 \times R_1 \times R_3 \to R_2$.

Обозначим соответственно через $v_1(e)$ и $v_2(e)$ начальную и конечную вершины дуги $e \in E$, а через $L(v_1,v_2)=\{L_k(v_1,v_2) | k=1, \ldots, r(v_1,v_2)\}$ – множество ориентированных путей в орграфе G из вершины v_1 в вершину v_2. Положим $L_k(v)=L_k(s,v)$ и $r(v)=r(s,v)$.

Пусть $x=(x(e)|e \in E)$ – вектор с компонентами $x(e)$ из R_1. Для пути $L_k(v)=(s=v_0, e_1, v_1, \ldots, v_{r-1}, e_r, v_r, \ldots, v_{l-1}, e_l, v_l=v)$, $v_{r-1}=v_1(e_r)$, $v_r=v_2(e_r)$, $r=1, \ldots, l$, и вектора x определим функцию $c_k(v, x)$ с помощью следующих рекуррентных соотношений:

$$c_k(v, x) = c_k^l(v, x); \tag{1}$$

$$c_k^r(v, x) = c_k^{r-1}(v, x) \bullet x(e_r),\ r=1,\ldots,l; \tag{2}$$

$$c_k^0(v, x) = 1(\bullet). \tag{3}$$

Предполагается также, что на множестве векторов вида $p=(x(e)|e \in E)$ с компонентами $p(e)$ из R_3 задана функция Φ, которая определяется операциями $\otimes$ и $\circ$ рекуррентным образом в соответствии с деревом декомпозиции орграфа G.

Для произвольного подграфа $G^0=(V^0,E^0)$ орграфа G обозначим через $\Phi_{G^0}(p)$ сужение функции Φ на множество дуг E^0. Положим $\Phi_{G(\{e\})}(p)=p(e)$, где $G(\{e\})$ – подграф, порожденный дугой e. Если подграфы G^1, G^2, …, G^m представляют собой элементы последовательной или параллельной композиции орграфа G, то $\Phi_G(p) = \Phi_{G^1}(p) \otimes \Phi_{G^2}(p) \otimes \ldots \otimes \Phi_{G^m}(p)$ и $\Phi_G(p) = \Phi_{G^1}(p) \circ \Phi_{G^2}(p) \circ \ldots \circ \Phi_{G^m}(p)$ соответственно.

Примерами функции $\Phi(p)$ являются функции

$$\Phi(p) = \min\{p(e) | e \in E\}; \tag{4}$$

$$\Phi(p) = \prod_{e \in E} p(e); \tag{5}$$

$$\Phi(p) = \min\{\prod_{e \in L_k(t)} p(e) | k=1,\ldots,r(s,t)\}. \tag{6}$$

Нетрудно показать, что функция (4) может быть определена как функция $\Phi_G(p)$, если в качестве операций $\otimes$ и $\circ$ принять операцию взятия минимума. Аналогично функция (5) может быть определена операциями умножения в качестве операций $\otimes$ и $\circ$, а функция (6) – операцией умножения в качестве операции $\otimes$ и операцией взятия минимума в качестве операции $\circ$.

Исходная задача заключается в отыскании для заданного действительного числа P_0 из R_3 параметров $x(e)$ и $p(e)$ дуг $e \in E$, удовлетворяющих следующей системе соотношений:

$$g(x,p) = \bigoplus_{e\in E} M_e(c_1(v_1(e),x),x(e),p(e)) \oplus \bigoplus_{v\in V} M_v(c_1(v,x)) \to \min; \tag{7}$$

$$c_k(t,x) = c^*,\ k=1,\ldots,r(s,t); \tag{8}$$

$$x(e) \in [\underline{x}(e), \bar{x}(e)],\ e \in E; \tag{9}$$

$$\Phi_G(p) \geq P_0; \tag{10}$$

$$p(e) \in [\underline{p}(e), \bar{p}(e)],\ e \in E. \tag{11}$$

Подобные задачи с операциями сложения в качестве операций + и $\oplus$ и функцией (4) в качестве $\Phi_G(p)$ рассматривались в [1].

С целью упрощения обозначений далее используются символы: «+» для обозначения операции $\bullet$, $\sum$ для этой операции над несколькими операндами, «–» для ее обратной операции и «0» для ее нейтрального элемента.

2. Анализ задачи

Выясним сначала вопрос о существовании допустимого решения задачи (7)–(11), т. е. совместности ограничений (8)–(11). В силу неубывания функции Ф по своим аргументам условия (10), (11), очевидно, выполняются тогда и только тогда, когда $\Phi(\bar{p}) \geq P_0$, где $\bar{p} = (\bar{p}(e), e \in E)$.

Перейдем теперь к исследованию совместности ограничений (8), (9). Обозначим через X множество векторов x, удовлетворяющих этим ограничениям.

Положим

$$\underline{c}(v_1,v_2) = \max\{\sum_{e\in L_k(v_1,v_2)} \underline{x}(e) | k=1,\ldots,r(v_1,v_2)\};$$

$$\bar{c}(v_1,v_2) = \min\{\sum_{e\in L_k(v_1,v_2)} \bar{x}(e) | k=1,\ldots,r(v_1,v_2)\}.$$

Нетрудно показать, что справедливо

Утверждение 1. Если граф G является ПП-орграфом, то $X \neq \varnothing$ тогда и только тогда, когда выполняются следующие соотношения:

$$\underline{c}(s,t) \leq c^* \leq \bar{c}(s,t); \tag{12}$$

$$\underline{c}(v_1,v_2) \leq \bar{c}(v_1,v_2),\ v_1,v_2 \in V,\ L(v_1,v_2) \neq \varnothing. \tag{13}$$

Следует заметить, что условия (12) и (13) не являются достаточными для непустоты множества X, если G не является ПП-орграфом.

Пример на рис. 1 показывает, что условия (12) и (13) не являются достаточными для непустоты множества X, если G не является ПП-орграфом. Здесь в качестве множества R_1 рассматривается множество действительных чисел, а в качестве операции «+» – обычная операция суммирования.

В квадратных скобках приведены допустимые диапазоны $[\underline{x}(e), \bar{x}(e)]$ для дуг орграфа G. Условия (12) и (13) выполняются для графа G при $c^* \in [6, 11]$, поскольку $\underline{c}(s,1)=1$, $\bar{c}(s,1)=4$,

$\underline{c}(s,2)=5$, $\overline{c}(s,2)=6$, $\underline{c}(s,3)=3$, $\overline{c}(s,3)=3$, $\underline{c}(s,t)=6$, $\overline{c}(s,t)=11$, $\underline{c}(1,2)=1$, $\overline{c}(1,2)=2$, $\underline{c}(1,3)=2$, $\overline{c}(1,3)=2$, $\underline{c}(1,t)=3$, $\overline{c}(1,t)=7$, $\underline{c}(2,t)=1$, $\overline{c}(2,t)=5$, $\underline{c}(3,t)=1$, $\overline{c}(3,t)=20$. Здесь и далее курсивом показаны номера вершин орграфов.

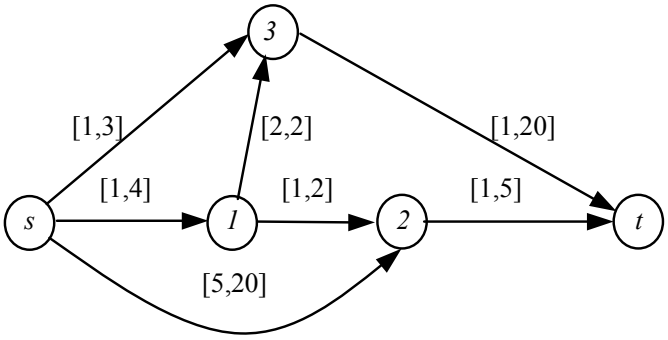


Рис. 1. Пример к утверждению 1

Для всех $x\in X$ значение $\sum_{e\in L_k(v_1,v_2)} x(e)$ должно быть одно и то же для всех путей $\{L_k(v_1,v_2)|$ $k=1,\ldots,r(v_1,v_2)\}$ и пар вершин $v_1,v_2\in V$, а такого вектора не существует. Действительно, $x(e)$ должно быть не меньше пяти для дуги из вершины s в вершину 2. Следовательно, значение $x(e)$ должно быть не меньше трех для дуги из вершины s в вершину 1 и $x(e)$ должно быть не меньше пяти для дуги из вершины s в вершину 3, что невозможно, поскольку это значение не может быть больше трех.

Значения $\underline{c}(v_1,v_2)$ и $\overline{c}(v_1,v_2)$ могут быть найдены с помощью алгоритма поиска кратчайших (критических) путей между всеми вершинами, трудоемкость которого $O(|V|^3)$. Ниже предлагается алгоритм проверки совместности ограничений (8) и (9) трудоемкости $O(|E|)$, использующий декомпозиционное дерево орграфа G. Алгоритм заключается в последовательном вычислении для каждого уровня декомпозиционного дерева (начиная с самого нижнего уровня) характеристик $\underline{x}(G^i)$ и $\overline{x}(G^i)$ составляющих его подграфов и проверке условия $\underline{x}(G^i) \le \overline{x}(G^i)$. Характеристики $\underline{x}(G)$ и $\overline{x}(G)$ вычисляются следующим образом.

Пусть орграф G получен в результате последовательной или параллельной композиции подграфов G^i, $i=1,\ldots,m$. Положим $\underline{x}(G)=\sum_{i=1}^{m} \underline{x}(G^i)$ и $\overline{x}(G)=\sum_{i=1}^{m} \overline{x}(G^i)$ в первом случае, а во втором – $\underline{x}(G)=\max\{\underline{x}(G^i)|i=1,\ldots,m\}$ и $\overline{x}(G)=\min\{\overline{x}(G^i)|\ i=1,\ldots,m\}$. Если подграф G^i состоит из единственной дуги e, то полагается $\underline{x}(G^i)=\underline{x}(e)$ и $\overline{x}(G^i)=\overline{x}(e)$. Нетрудно видеть, что $\underline{x}(G)=\underline{c}(s(G),t(G))$ и $\overline{x}(G)=\overline{c}(s(G),t(G))$. Следовательно, если $\underline{x}(G^i)>\overline{x}(G^i)$ для некоторого подграфа G^i, то задача (7)–(11) не имеет решения, так как в этом случае нарушается необходимое условие (13) непустоты множества X.

Определим для каждой вершины $v\in V$ значения $\underline{\beta}(v)$ и $\overline{\beta}(v)$ с помощью следующего алгоритма. Как будет показано ниже, эти значения определяют диапазон возможных значений $c_1(v,x)$ для векторов $x\in X$. В рассматриваемом алгоритме $\{G_k^i|i=1,\ldots,n_k\}$ – множество подграфов исходного графа, соответствующих k-му уровню его декомпозиционного дерева, а $\{G_k^{ij}|j=1,\ldots,n_{ki}\}$ – множество подграфов графа G_k^i, полученных при его декомпозиции.

Алгоритм определения диапазонов значений $c_1(v, x)$

Шаг 1. Положить $\underline{\beta}(s)=\overline{\beta}(s)=0$, $\underline{\beta}(t)=\overline{\beta}(t)=c^*$.

Шаг 2. Для каждого k-го уровня последовательной декомпозиции в порядке убывания k и для каждого $i=1,\ldots,n_k$ положить:

а) $\underline{c}=0$, $\overline{c}=0$;

б) для каждого $j=1,\ldots,n_{ki}-1$

$$\underline{c}=\underline{c}+\underline{c}(s(G_k^{ij}),t(G_k^{ij})),\ \overline{c}=\overline{c}+\overline{c}(s(G_k^{ij}),t(G_k^{ij}));$$

$$\underline{c}(s(G_k^i),t(G_k^{ij}))=\underline{c},\ \overline{c}\,(s(G_k^i),t(G_k^{ij}))=\overline{c}\,;$$

$$\underline{c}(t(G_k^{ij}),t(G^i))=\underline{x}(G_k^i)-\underline{c},\ \overline{c}\,(t(G_k^{ij}),t(G_k^i))=\overline{x}\,(G_k^i)-\overline{c}\,;$$

$$\underline{\beta}(t(G_k^{ij}))=\max\{\underline{\beta}(s(G_k^i))+\underline{c}(s(G_k^i),t(G_k^{ij})),\ \underline{\beta}(t(G_k^i))-\overline{c}\,(s(G_k^{ij}),t(G_k^i))\};\qquad(14)$$

$$\overline{\beta}\,(t(G_k^{ij}))=\min\{\overline{\beta}\,(s(G_k^i))+\overline{c}\,(s(G_k^i),t(G_k^{ij})),\ \overline{\beta}\,(t(G_k^i))-\underline{c}(s(G_k^{ij}),t(G_k^i))\}.\qquad(15)$$

Продемонстрируем работу алгоритма на примере ПП-орграфа G (рис. 2), на котором в квадратных скобках приведены допустимые диапазоны $[\underline{x}(e), \overline{x}\,(e)]$ его дуг. В табл. 1 приведены значения $\underline{x}(G^i)$ и $\overline{x}\,(G^i)$ составляющих подграфов декомпозиционного дерева орграфа G (рис. 3). Результаты вычислений приведены в табл. 2 для $c^*=10$. Поскольку $\underline{c}(s,4)=4$, $\overline{c}\,(s,4)=6$, $\underline{c}(4,t)=4$ и $\overline{c}\,(4,t)=5$, то в силу (14) $\underline{\beta}(4)=\max\{\underline{\beta}(s)+\underline{c}(s,4),\ \underline{\beta}(t)-\overline{c}\,(4,t)\}=\max\{0+4,10-5\}=5$, а в силу (15) $\overline{\beta}\,(4)=\min\{\overline{\beta}\,(s)+\overline{c}\,(s,4),\ \overline{\beta}\,(t)-\underline{c}(4,t)\}=\min\{0+6,10-4\}=6$. Аналогичным образом $\underline{c}(s,2)=1$, $\overline{c}\,(2,4)=6$, $\underline{\beta}(2)=\max\{\underline{\beta}(s)+\underline{c}(s,2),\underline{\beta}(4)-\overline{c}\,(2,4)\}=\max\{0+1,5-6\}=1$, $\overline{\beta}\,(2)=\min\{\overline{\beta}\,(s)+\overline{c}\,(s,2),\ \overline{\beta}\,(4)-\underline{c}(2,4)\}=\min\{0+3,\ 6-2\}=3$, $\underline{c}(s,3)=2$, $\overline{c}\,(3,4)=2$, $\underline{\beta}(3)=\max\{\underline{\beta}(s)+\underline{c}(s,3),\ \underline{\beta}(4)-\overline{c}\,(3,4)\}=\max\{0+2,\ 5-2\}=3$, $\overline{\beta}\,(3)=\min\{\overline{\beta}\,(s)+\overline{c}\,(s,3),\overline{\beta}\,(4)-\underline{c}(3,4)\}=\min\{0+7,\ 6-1\}=5$.

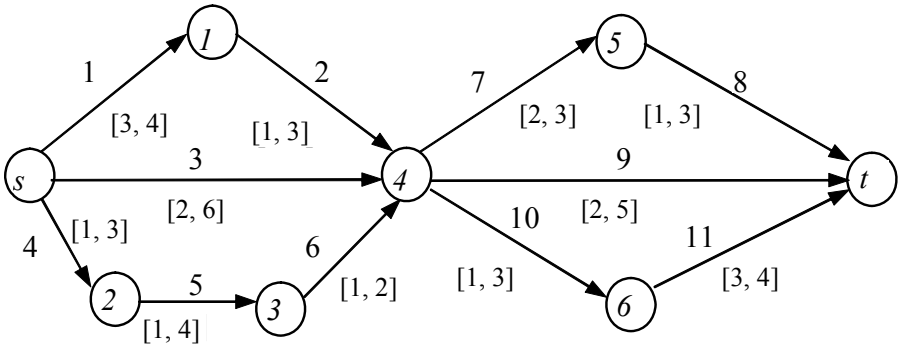


Рис. 2. ПП-орграф G

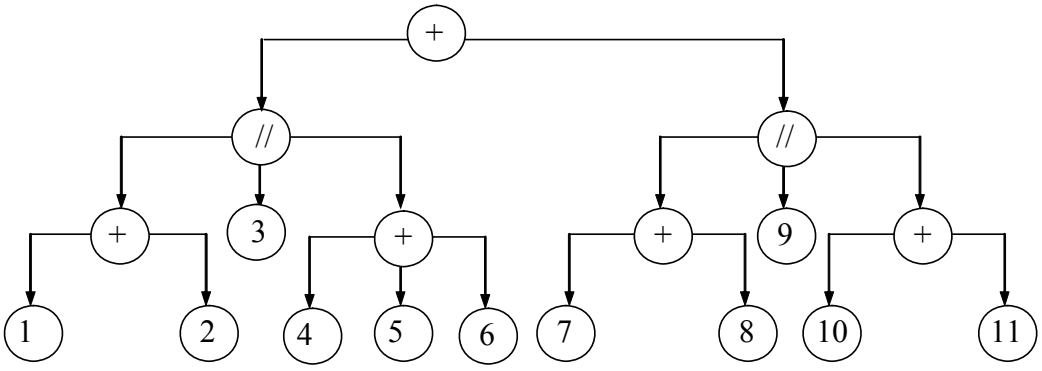


Рис. 3. Декомпозиционное дерево ПП-орграфа G

Таблица 1

Характеристики декомпозиционного дерева ПП-орграфа G

$E(G^i)$	{1},{2}	{4},{5},{6}	{7},{8}	{10},{11}	{3}	{1},{2},{3},{4},{5},{6}	{9}	{7},{8},{9},{10},{11}	{1},{2},{3},{4},{5},{6},{7},{8},{9},{10},{11}
$\underline{x}(G^i)$	4	3	3	4	2	4	2	4	8
$\overline{x}(G^i)$	7	9	6	7	6	6	5	5	11

Таблица 2

Результаты работы алгоритма определения диапазонов значений $c_1(v, x)$

v	s	*1*	*2*	*3*	*4*	*5*	*6*	t
$\underline{\beta}(v)$	0	3	1	3	5	7	6	10
$\overline{\beta}(v)$	0	4	3	5	6	9	7	10

Из соотношений (14) и (15) непосредственно следует, что справедливо

Утверждение 2. Для всех $v \in V$

$$\underline{\beta}(v)=\max\{\underline{c}(s,v),\ c^*-\overline{c}\,(v,t)\};$$

$$\overline{\beta}\,(v)=\min\{\overline{c}\,(s,v),\ c^*-\underline{c}(v,t)\}.$$

Утверждение 3. Множество $X\neq\varnothing$ тогда и только тогда, когда $\underline{\beta}(v_2(e))-\underline{\beta}(v_1(e))\in[\underline{x}(e), \overline{x}\,(e)]$.

Доказательство. Достаточность очевидна, поскольку в этом случае можно положить $x=(\underline{\beta}(v_2(e))-\underline{\beta}(v_1(e)),e\in E)$. Тогда $x\in X$, поскольку $c_k(v,x)=\underline{\beta}(v)$ и, следовательно, $c_k(t,x)=c^*$.

Докажем необходимость. Пусть граф G является последовательной композицией подграфов $G^i=(V^i,E^i)$, $i=1,\ldots,m$, и вершины $v_i=t(G^i)$ – их стоки.

Покажем сначала, что $\beta(v_i)-\beta(s)\in[\underline{c}(s,v_i), \overline{c}\,(s,v_i)]$ и $\beta(t)-\beta(v_i)\in[\underline{c}(v_i,t), \overline{c}\,(v_i,\ t)]$ для всех $i=1,\ldots,m$.

Если $\beta(v_i)=\underline{c}(s,v_i)$, то $\beta(v_i)-\beta(s)=\underline{c}(s,v_i)\le\overline{c}\,(s,v_i)$ и $\beta(t)-\beta(v_i)=c^*-\underline{c}(s,v_i)\ge\underline{c}(s,t)-\underline{c}(s,v_i)=$ $=\underline{c}(s,v_i)+\underline{c}(v_i,t)-\underline{c}(s,v_i)=\underline{c}(v_i,t)\le\overline{c}\,(v_i,t)$ в силу утверждения 2.

Пусть $\beta(v_i)=c^*-\overline{c}\,(v_i,t)>\underline{c}(s,v_i)$. Тогда $\beta(t)-\beta(v_i)=c^*-c^*+\overline{c}\,(v_i,t)=\overline{c}\,(v_i,t)\ge\underline{c}(v_i,t)$ и $\beta(v_i)-\beta(s)=c^*-\overline{c}\,(v_i,t)\le\overline{c}\,(s,t)-\overline{c}\,(v_i,t)=\overline{c}\,(s,v_i)+\overline{c}\,(v_i,t)-\overline{c}\,(v_i,t)=\overline{c}\,(s,v_i)$.

Рассмотрим теперь вершины v_i и v_{i+1}. Возможны следующие ситуации:

а) если $\beta(v_i)=\underline{c}(s,v_i)$, $\beta(v_{i+1})=\underline{c}(s,v_{i+1})$, то $\beta(v_{i+1})-\beta(v_i)=\underline{c}(s,v_{i+1})-\underline{c}(s,v_i)=\underline{c}(v_i,v_{i+1})\le$ $\le\overline{c}\,(v_i,v_{i+1})$;

б) если $\beta(v_i)=c^*-\overline{c}\,(v_i,t)$, $\beta(v_{i+1})=c^*-\overline{c}\,(v_{i+1},t)$, то $\beta(v_{i+1})-\beta(v_i)=c^*-\overline{c}\,(v_{i+1},t)-c^*+\overline{c}\,(v_i,t)=$ $=\overline{c}\,(v_i,v_{i+1})\ge\underline{c}(v_i,v_{i+1})$;

в) если $\beta(v_i)=\underline{c}(s,v_i)$, $\alpha(v_{i+1})=c^*-\overline{c}\,(v_{i+1},t)$, то $\beta(v_{i+1})-\beta(v_i)=c^*-\overline{c}\,(v_{i+1},t)-\underline{c}(s,v_i)\ge\underline{c}(s,v_{i+1})-$ $-\underline{c}(s,v_i)=\underline{c}(v_i,v_{i+1})$. С другой стороны, $\beta(v_{i+1})-\beta(v_i)=c^*-\overline{c}\,(v_{i+1},t)-\underline{c}(s,v_i)\le c^*-\overline{c}\,(v_{i+1},t)-c^*+$ $+\overline{c}\,(v_i,t)=\overline{c}\,(v_i,v_{i+1})$;

г) $\beta(v_i)=c^*-\overline{c}\,(v_i,t)>\underline{c}(s,v_i)$, $\beta(v_{i+1})=\underline{c}(s,v_{i+1})>c^*-\overline{c}\,(v_{i+1},t)$. Данный случай невозможен, поскольку тогда одновременно $\beta(v_{i+1})-\beta(v_i)=\underline{c}(s,v_{i+1})-c^*+\overline{c}\,(v_i,t)>c^*-\overline{c}\,(v_{i+1},t)-c^*+\overline{c}\,(v_i,t)=$ $\overline{c}\,(v_i,v_{i+1})$ и $\beta(v_{i+1})-\beta(v_i)=\underline{c}(s,v_{i+1})-c^*+\overline{c}\,(v_i,t)<\underline{c}(s,v_{i+1})-\underline{c}(s,v_i)=\underline{c}(v_i,v_{i+1})$.

Таким образом, исходная задача разбивается на m независимых подзадач того же класса для подграфов $G^i=(V^i,E^i)$, $i=1,\ldots,m$, являющихся компонентами последовательного разложения графа G, со значениями $(c^*)^i=\beta(t(G^i))-\beta(s(G^i))$ соответственно. Заметим также, что $\sum_{i=1}^{m}(c^*)^i=c^*$, т. е. если векторы $x^i(e)$, $e\in E^i$, удовлетворяют условию (8) для $(c^*)^i$, то вектор $x(e)$, являющийся объединением векторов $x^i(e)$, принадлежит X.

Поскольку граф G является ПП-орграфом, то, продолжая этот процесс до конца (пока подграфы G^i не станут дугами), получим, что $\beta(v_2(e)) - \beta(v_1(e)) \in [\underline{c}(v_1(e),v_2(e)), \overline{c}\,(v_1(e),v_2(e))]$ для любой дуги $e \in E$. Поскольку $X \neq \varnothing$, то в силу утверждения 1 $\underline{c}(v_1(e),v_2(e)) \leq \overline{c}\,(v_1(e),v_2(e))$ и, следовательно, $\underline{\beta}(v_2(e)) - \underline{\beta}(v_1(e)) \in [\underline{x}(e), \overline{x}\,(e)]$. Утверждение полностью доказано. ■

Аналогичным образом показывается, что справедливо

Утверждение 4. Множество $X \neq \varnothing$ тогда и только тогда, когда $\overline{\beta}\,(v_2(e)) - \overline{\beta}\,(v_1(e)) \in [\underline{x}(e), \overline{x}\,(e)]$.

Более того, непосредственно из соотношений (14), (15) и строгой монотонности операции «–» следует

Утверждение 5. Если $X \neq \varnothing$, то $\underline{\beta}(v) = \min\{c_1(v,x) | x \in X\}$ и $\overline{\beta}\,(v) = \max\{c_1(v,x) | x \in X\}$ для любых $v \in V$, причем существуют единственные векторы $x^{min} \in X$ и $x^{max} \in X$, такие, что $\underline{\beta}(v) = c_1(v,x^{min})$ и $\overline{\beta}\,(v) = c_1(v,x^{max})$.

В свою очередь, из утверждения 5 вытекает

Следствие 1. Если $X \neq \varnothing$, то $\underline{\beta}(v) \leq \overline{\beta}\,(v)$ для всех $v \in V$.

2. Общая декомпозиционная схема решения

В силу предыдущих утверждений задача (7)–(11) эквивалентна задаче

$$g(x,p) = \bigoplus_{e \in E} M_e(c_1(v_1(e),x), c_1(v_2(e),x) - c_1(v_1(e),x), p(e)) \oplus \bigoplus_{v \in V} M_v(c_1(v,x)) \to \min; \tag{16}$$

$$c_1(t,x) = c^*; \tag{17}$$

$$c_1(v,x) \in [\underline{\beta}(v), \overline{\beta}\,(v)],\ v \in V \setminus \{s,t\}; \tag{18}$$

$$c_1(v_2(e),x) - c_1(v_1(e),x) \in [\underline{x}(e), \overline{x}\,(e)],\ e \in E; \tag{19}$$

$$\Phi_G(p) \geq P_0; \tag{20}$$

$$p(e) \in [\underline{p}(e), \overline{p}\,(e)],\ e \in E. \tag{21}$$

Для решения задачи (16)–(21) можно эффективно использовать метод фрагментарной параметрической декомпозиции [2]. Введем в качестве дополнительных переменных параметры $y(v) = c_1(v,x)$ для всех $v \in V$. Тогда параметризованная задача **C** получается из исходной задачи заменой в соотношениях (16)–(21) фрагментов $c_1(v, x)$ параметрами $y(v)$. Для выбранной схемы параметризации естественным образом выполняются достаточные условия [2], при которых описанная декомпозиционная схема обеспечивает получение решения исходной задачи.

В результате задача (16)–(21) преобразуется в параметризованную задачу **C** отыскания наборов $(y(v), v \in V)$ и $(p(e), e \in E)$, удовлетворяющих следующей системе соотношений:

$$F(y,p) = \bigoplus_{e \in E} M_e^*(y(v_1(e)), y(v_2(e)), p(e)) \oplus \bigoplus_{v \in V} M_v^*(y(v)) \to \min; \tag{22}$$

$$y(v_1(e)) + \underline{x}(e) \leq y(v_2(e)),\ e \in E; \tag{23}$$

$$y(v_1(e)) + \overline{x}\,(e) \geq y(v_2(e)),\ e \in E; \tag{24}$$

$$y(v) \in [\underline{\beta}(v), \overline{\beta}\,(v)],\ v \in V \setminus \{s,t\}; \tag{25}$$

$$y(s) = 0,\ y(t) = c^*; \tag{26}$$

$$\Phi_G(p) \geq P_0; \tag{27}$$

$$p(e)\in[\underline{p}(e),\ \overline{p}\,(e)],\ e\in E, \tag{28}$$

где $M_e^*(y(v_1(e)),y(v_2(e)),p(e)) = M_e(y(v_1(e)),y(v_2(e))-y(v_1(e)),p(e))$.

3. Методы решения параметризованной задачи

Будем предполагать далее, что в множестве R_3 существуют нейтральные элементы $1(\otimes)$ и $1(\circ)$ для операций $\otimes$ и $\circ$ соответственно, т. е. $a\otimes 1(\otimes)=a$ и $a\circ 1(\circ)=a$ для любого $a\in R_3$. Напомним, что функция Φ определяется операциями $\otimes$ и $\circ$ рекуррентным способом в соответствии с деревом декомпозиции графа G.

Предлагаемые методы решения задачи **C** базируются на использовании этих свойств функции Φ, а также свойств операций $+$, $\oplus$, $\otimes$ и $\circ$, позволяющих получить решение задачи **C** с помощью рекурсивной процедуры. Каждый шаг этой процедуры соответствует некоторому уровню дерева декомпозиции графа G.

Рассмотрим основные шаги этой декомпозиционной процедуры более детально, начиная с последовательной декомпозиции.

Пусть исходный граф G является последовательной композицией подграфов $G_k=(V_k,E_k)$ и $v_{k-1}=s(G_{k-1})$, $v_k=t(G_{k-1})$, $k=1,\ldots,m$.

Для $k = 1,\ldots,m$ положим

$$\underline{\pi}_k=\max\left[\bigotimes_{j=1}^{k}\Phi_{G_j}(\underline{p}),\ \min\left\{a\in R_3\mid a\otimes\bigotimes_{j=k+1}^{m}\Phi_{G_j}(\overline{p})\ge P_0\right\}\right];$$

$$\overline{\pi}_k=\min\left[\bigotimes_{j=1}^{k}\Phi_{G_j}(\overline{p}),\ \max\left\{a\in R_3\mid a\otimes\bigotimes_{j=k+1}^{m}\Phi_{G_j}(\underline{p})\ge P_0\right\}\right];$$

$$\underline{\pi}_0=\overline{\pi}_0=1(\otimes),\ \underline{\pi}_{m+1}=\max\left[\bigotimes_{j=1}^{m+1}\Phi_{G_j}(\underline{p}),\ P_0\right]\ \text{и}\ \overline{\pi}_{m+1}=\bigotimes_{j=1}^{m+1}\Phi_{G_j}(\overline{p}).$$

Для любого вектора $p\in\mathbf{P}$ определим набор $\pi(p)=(\pi_0(p),\pi_1(p),\ldots,\pi_m(p))$, такой что $\pi_0(p) = 1(\otimes)$ и $\pi_k(p) = \bigotimes_{j=1}^{k}\Phi_{G_j}(p)$, $k=1,\ldots,m$.

Пусть $\mathbf{\Pi} = \{\pi(p)\mid p\in\mathbf{P}\}$. Справедливо следующее

Утверждение 6. $\mathbf{\Pi} = \{(\pi_0,\pi_1,\ldots,\pi_m)\mid\pi_k\in[\underline{\pi}_k,\overline{\pi}_k],\ k = 0,\ldots,m,\ \pi_{k-1}\otimes\Phi_{G_k}(\underline{p})\le\pi_k$ и $\pi_{k-1}\otimes\otimes\Phi_{G_k}(\overline{p})\ge\pi_k,\ k=1,\ldots,m\}$.

Доказательство. Пусть $p\in\mathbf{P}$. Покажем, что $\pi(p)\in\mathbf{\Pi}$. Поскольку по определению $\pi_k(p)= \pi_{k-1}(p)\otimes\Phi_{G_k}(p)$ и $p(e)\in[\underline{p}(e),\ \overline{p}\,(e)]$ для всех $e\in E_k$, то $\pi_{k-1}\otimes\Phi_{G_k}(\underline{p})\le\pi_k$ и $\pi_{k-1}\otimes\Phi_{G_k}(\overline{p})\ge\pi_k$ для $k=1,\ldots,m$ в силу неубывания операции $\otimes$. Осталось показать, что $\pi_k(p)\in[\underline{\pi}_k,\overline{\pi}_k]$. По определению $\pi_0(p)=\underline{\pi}_0=\overline{\pi}_0$.

В силу свойств операции $\otimes$ следует, что $\bigotimes_{j=1}^{k}\Phi_{G_j}(\underline{p})\le\pi_k(p)\le\bigotimes_{j=1}^{k}\Phi_{G_j}(\overline{p})$ для $k=1,\ldots,m$. Предположим от противного, что $\pi_k(p)<\underline{\pi}_k$. Это означает, что $\pi_k(p)< \min\left\{a\in R_3\mid a\otimes\bigotimes_{j=k+1}^{m}\Phi_{G_j}(\overline{p})\ge P_0\right\}$ и, следовательно, $\pi_k(p)\otimes\bigotimes_{j=k+1}^{m}\Phi_{G_j}(\overline{p})<P_0$. Тогда $\Phi_G(p)=\pi_m(p)=\bigotimes_{j=1}^{m}\Phi_{G_j}(p)=\bigotimes_{j=1}^{k}\Phi_{G_j}(p)\otimes\bigotimes_{j=k+1}^{m}\Phi_{G_j}(p)\le\pi_k(p)\otimes\bigotimes_{j=k+1}^{m}\Phi_{G_j}(\overline{p})<P_0$, т. е. для век-

тора p не выполняется условие (27). Аналогичным образом показывается, что $\pi_k(p) \le \overline{\pi}_k$.

С другой стороны, для любого набора $\pi=(\pi_0,\pi_1,\ldots,\pi_m)\in\mathbf{\Pi}$ построим вектор $p(\pi)$ с компонентами $p(\pi,E_k)=\min\{a\in R_3 | \pi_k \ge \pi_{k-1}\otimes a,\ a\in[\Phi_{G_j}(\underline{p}), \Phi_{G_j}(\overline{p})]\}$ для $k=1,\ldots,m$. Такой вектор существует, поскольку $\pi_{k-1}\otimes\Phi_{G_j}(\underline{p}) \le \pi_k$ и $\pi_{k-1}\otimes\Phi_{G_j}(\overline{p}) \ge \pi_k$. Кроме того, $\Phi_G(p(\pi))=\pi_m \ge P_0$. ■

Поскольку значения функций $M_s^*(y(s))$ и $M_t^*(y(t))$ одни и те же для любых допустимых значений вектора y, задача **C** может быть представлена в виде

$$\bigoplus_{k=1}^{m} F_k^*(y(v_{k-1}),y(v_k),\pi_{k-1},\pi_k) \oplus \bigoplus_{k=1}^{m-1} M_{v_k}^*(y(v_k)) \to \min; \tag{29}$$

$$y(v_k)\in[\underline{\beta}(v), \overline{\beta}(v)],\ k=1,\ldots,m-1; \tag{30}$$

$$y(v_0) = 0; \tag{31}$$

$$y(v_m) = c^*; \tag{32}$$

$$\pi\in\mathbf{\Pi}, \tag{33}$$

где значения функции $F_k^*(y',y'',\pi',\pi'')$ определяются в результате решения следующей задачи $\mathbf{C}_k(y',y'',\pi',\pi'')$ при фиксированных значениях параметров y', y'', π' и π'':

$$\bigoplus_{e\in E_k} M_e^*(y(v_1(e)),y(v_2(e)),p(e)) \oplus \bigoplus_{v\in V_k\setminus\{v_{k-1},v_k\}} M_v^*(y(v)) \to \min; \tag{34}$$

$$y(v_1(e))+\underline{x}(e) \le y(v_2(e)),\ e\in E_k; \tag{35}$$

$$y(v_1(e))+\overline{x}(e) \ge y(v_2(e)),\ e\in E_k; \tag{36}$$

$$y(v)\in[\underline{\beta}(v), \overline{\beta}(v)],\ v\in V_k\setminus\{v_{k-1},v_k\}; \tag{37}$$

$$y(v_{k-1}) = y',\ y(v_k) = y''; \tag{38}$$

$$\Phi_{G_k}(p) \ge \min\{a\in R_3 | \pi' \otimes a \ge \pi''\}; \tag{39}$$

$$p(e)\in[\underline{p}(e), \overline{p}(e)],\ e\in E_k. \tag{40}$$

Предполагается, что функция $F_k^*(y',y'',\pi',\pi'')$ принимает достаточно большое значение, если ограничения (35)–(38) несовместны. Как и в случае исходной задачи, совместность этих ограничений может быть установлена с помощью алгоритма определения диапазонов значений $c_1(v, x)$.

Таким образом, решение задачи (34)–(40) может быть получено с помощью следующего рекуррентного соотношения:

$$\Im_k(y',\pi') = M_{v_k}^*(y') \oplus \min\{\Im_{k-1}(y'',\pi'') \oplus M_{e_k}^*(y'',y',a) | y''\in[\underline{\beta}(v_{k-1}), \overline{\beta}(v_{k-1})],$$

$$y'\in[y''+\underline{x}(e_k), y''+\overline{x}(e_k)],\ \pi''\in[\underline{\pi}_{k-1}, \overline{\pi}_{k-1}],\ \pi''\otimes a\ge\pi',\ a\in[\Phi_{G_j}(\underline{p}), \Phi_{G_j}(\overline{p})]\},\ k=1,\ldots,m, \tag{41}$$

где $\Im_0(y_0,\pi_0)=0$.

При этом целесообразно ввести дополнительные ограничения

$$y(v_{k-1})+\underline{\varepsilon}_k \le y(v_k),\ k=1,\ldots,m+1; \tag{42}$$

$$y(v_{k-1})+\overline{\varepsilon}_k \ge y(v_k),\ k=1,\ldots,m+1, \tag{43}$$

где

$$\underline{\varepsilon}_k =\max\{\sum_{e\in L_k(v_{i-1},v_i)} \underline{x}(e)\}|i=1,\ldots,r(v_{k-1},v_k)\}; \tag{44}$$

$$\overline{\varepsilon}_k =\min\{\sum_{e\in L_k(v_{i-1},v_i)} \overline{x}(e)\}|i=1,\ldots,r(v_{k-1},v_k)\}. \tag{45}$$

Значения $\underline{\varepsilon}_i$ и $\overline{\varepsilon}_i$, $i=1,\ldots,m+1$, могут быть легко найдены с помощью алгоритмов, аналогичных алгоритмов поиска кратчайшего пути в ациклическом орграфе. Справедливо следующее

Утверждение 7. Если значения $y(v_{i-1})=y'$ и $y(v_i)=y''$ являются компонентами некоторого вектора y из области $\mathbf{Y}$ возможных значений задачи $\mathbf{C}$, то $y'+\underline{\varepsilon}_i \le y''$ и $y'+\overline{\varepsilon}_i \ge y''$.

Доказательство. Предположим противное, т. е. для некоторого вектора $y\in\mathbf{Y}$ существуют компоненты $y(v_{i-1})=y'$ и $y(v_i)=y''$, такие, что $y'+\underline{\varepsilon}_i > y''$. В силу соотношения (44) существует путь, скажем $L_{k'}(v_{i-1},v_i)$, из вершины v_{i-1} в вершину v_i, для которого $\underline{\varepsilon}_i = \sum_{e\in L_k(v_{i-1},v_i)} \underline{x}(e)$.

Поскольку при сделанном предположении $y'+\underline{\varepsilon}_i > y''$, то в силу строгой монотонности операции «+» немедленно получаем, что $y(v_1(e)) + \underline{x}(e) > y(v_2(e))$ для некоторой дуги e, принадлежащей пути $L_{k'}(v_{i-1},v_i)$. Аналогично показывается и справедливость второго неравенства. ■

Для конкретных задач вместо всего множества $\mathbf{\Pi}$ можно рассматривать его некоторое подмножество. Так, для задачи с функцией Ф, определяемой соотношением (4), можно положить $\underline{\pi}_0=\overline{\pi}_0=\max\{\overline{p}(e)|e\in E\}$, $\underline{\pi}_k=\overline{\pi}_k=\max[P_0,\min\{\underline{p}(e)|e\in E_j,\ j=1,\ldots,k\}]$ для $k=1,\ldots,m$, а в случае соотношений (5) и (6) – $\underline{\pi}_0=\overline{\pi}_0=1$, $\underline{\pi}_k=\max\left[\prod_{j=1}^{k}\prod_{e\in E_j}\underline{p}(e), \frac{\overline{P}_0}{\prod_{j=k+1}^{m}\prod_{e\in E_j}\overline{p}(e)}\right]$,

$\overline{\pi}_k=\min\left[\prod_{j=1}^{k}\prod_{e\in E_j}\overline{p}(e), \frac{\overline{P}_0}{\prod_{j=k+1}^{m}\prod_{e\in E_j}\underline{p}(e)}\right]$ для $k=1,\ldots,m-1$, $\underline{\pi}_m=\overline{\pi}_m=\overline{P}_0$, где $\overline{P}_0=\max\left[\prod_{j=1}^{m}\prod_{e\in E_j}\underline{p}(e), P_0\right]$.

Задача (34)–(40) для каждого подграфа G_k, $k=1,\ldots,m$, аналогична исходной задаче, но имеет меньшую размерность. Поскольку исходный граф G является последовательно-параллельным, его подграф G_k является либо параллельно разложимым, либо состоит из единственной дуги между вершинами v_{k-1} и v_k. В последнем случае $F_k^*(y',y'',\pi',\pi'')=M_e^*(y',y'',p^*(e))$, где $p^*(e)=\min\{p(e)\in[\underline{p}(e),\overline{p}(e)]|\pi'\otimes p(e)\ge\pi''\}$.

Пусть граф G_k параллельно разложим на подграфы $G_{kj}=(V_{kj},E_{kj})$, $j=1,\ldots,l_k$.

Обозначим через $\otimes_{\min}(\pi',\pi'')=\min\{a\in R_3|\pi'\otimes a\ge\pi''\}$ и положим для $i=1,\ldots,l=l_k$

$$\underline{\theta}_i=\max\left[\mathop{\circ}_{j=1}^{i}\Phi_{G_{kj}}(\underline{p}), \min\left\{a\in R_3 \mid a\circ\mathop{\circ}_{j=i+1}^{l}\Phi_{G_{kj}}(\overline{p})\ge\otimes_{\min}(\pi',\pi'')\right\}\right];$$

$$\overline{\theta}_i=\min\left[\mathop{\circ}_{j=1}^{i}\Phi_{G_{kj}}(\overline{p}),\ \max\left\{a\in R_3\mid\otimes_{\min}(\pi',\pi'')\le a\circ\mathop{\circ}_{j=i+1}^{l}\Phi_{G_{kj}}(\underline{p})\le\mathop{\circ}_{j=1}^{l}\Phi_{G_{kj}}(\overline{p})\right\}\right];$$

$$\underline{\theta}_0=\overline{\theta}_0=0(\circ),\ \underline{\theta}_l=\max\left[\mathop{\circ}_{j=1}^{l}\Phi_{G_{kj}}(\underline{p}),\ \otimes_{\min}(\pi',\pi'')\right]\ \text{и}\ \overline{\theta}_l=\mathop{\circ}_{j=1}^{l}\Phi_{G_{kj}}(\overline{p}).$$

Пусть Θ – множество таких наборов $\theta=(\theta_0,\theta_1,\ldots,\theta_l)$, что $\theta_i\in[\underline{\theta}_i,\overline{\theta}_i]$ для $i=0,\ldots,l$, $\theta_{i-1}\circ\Phi_{G_{ki}}(\underline{p})\le\theta_i$ и $\theta_{i-1}\circ\Phi_{G_{ki}}(\overline{p})\ge\theta_i$ для $i=1,\ldots,l$. Как и ранее, нетрудно показать, что каждому вектору $p_k=(p(e),e\in E_k)$, удовлетворяющему условиям (39) и (40), соответствует вектор $\theta=(\theta_0,\theta_1,\ldots,\theta_l)\in\Theta$, такой, что $\theta_i=\mathop{\circ}_{j=1}^{i}\Phi_{G_{kj}}(p)$ для $i=1,\ldots,l$.

Тогда вычисление значения функции $F_k^*(y',y'',\pi',\pi'')$ при фиксированных значениях ее аргументов сводится к решению следующей задачи:

$$\mathop{\oplus}_{i=1}^{l}F_{ki}^*(y',y'',\theta_{i-1},\theta_i)\to\min, \tag{46}$$

$$\theta\in\Theta, \tag{47}$$

где значения $F_{ki}^*(y',y'',\theta_{i-1},\theta_i)$ соответствуют минимальному значению функции

$$\mathop{\oplus}_{e\in E_{ki}}M_e^*(y(v_1(e)),y(v_2(e)),p(e))\oplus\mathop{\oplus}_{v\in V_{ki}\setminus\{v_{k-1},v_k\}}M_v^*(y(v)) \tag{48}$$

при выполнении ограничений

$$y(v_1(e))+\underline{x}(e)\le y(v_2(e)),\ e\in E_{ki}; \tag{49}$$

$$y(v_1(e))+\overline{x}(e)\ge y(v_2(e)),\ e\in E_{ki}; \tag{50}$$

$$y(v)\in[\underline{\beta}(v),\overline{\beta}(v)],\ v\in V_{ki}\setminus\{v_{k-1},v_k\}; \tag{51}$$

$$y(v_{k-1})=y',\ y(v_k)=y''; \tag{52}$$

$$\Phi_{G_{ki}}(p)\ge\min\{a\in R_3\mid\theta_{i-1}\circ a\ge\theta_i\}; \tag{53}$$

$$p(e)\in[\underline{p}(e),\overline{p}(e)],\ e\in E_{ki}. \tag{54}$$

Пусть $f_{ki}(\theta_i)$ – наименьшее значение функции $\mathop{\oplus}_{j=1}^{i}F_{kj}^*(y',y'',\theta_{j-1},\theta_j)$ по всем таким наборам $(p(e)\mid e\in E_{kj})$, что $p(e)\in[\underline{p}(e),\overline{p}(e)]$ и $\mathop{\circ}_{j=1}^{i}\Phi_{G_{kj}}(p)\ge\theta_i$. Значение функции $F_k^*(y',y'',\pi',\pi'')=f_{kl}(\otimes_{\min}(\pi',\pi''))$ может быть получено из следующего рекуррентного соотношения:

$$f_{ki}(\theta_i)=\min\{f_{ki-1}(\theta_{j-1})\oplus F_{ki}^*(y',y'',\theta_{i-1},\theta_i)\mid$$
$$\theta_{i-1}\in[\underline{\theta}_{i-1},\overline{\theta}_{i-1}],\ \theta_{i-1}\circ\Phi_{G_{ki}}(\underline{p})\ge\theta_i,\ \theta_{i-1}\circ\Phi_{G_{ki}}(\overline{p})\le\theta_i\},\ i=1,\ldots,l, \tag{55}$$

где $\theta_0=1(\circ)$ и $f_{k0}(\theta_0)=1(\oplus)$.

При решении конкретных задач (46), (47) вместо множества Θ можно рассматривать его некоторое подмножество. Видно, что для задачи с функцией Ф, определяемой соотношением (4), можно положить $\underline{\theta}_0=\overline{\theta}_0=W$, $\underline{\theta}_i=\overline{\theta}_i=\max[\pi'',\min\{\underline{p}(e)|e\in E_{kj}, j=1,\ldots,i\}]$ для $i=1,\ldots,l_k$, в случае соотношения (5) – $\underline{\theta}_0=\overline{\theta}_0=1$, $\underline{\theta}_i=\max\left[\prod\limits_{j=1}^{i}\prod\limits_{e\in E_{kj}}\underline{p}(e),\dfrac{\pi''/\pi'}{\prod\limits_{j=i+1}^{l_k}\prod\limits_{e\in E_{kj}}\overline{p}(e)}\right]$, $\overline{\theta}_i=\min\left[\prod\limits_{j=1}^{i}\prod\limits_{e\in E_{kj}}\overline{p}(e),\dfrac{\pi''/\pi'}{\prod\limits_{j=i+1}^{l_k}\prod\limits_{e\in E_{kj}}\underline{p}(e)}\right]$ для $i=1,\ldots,l_k-1$, $\underline{\theta}_{l_k}=\overline{\theta}_{l_k}=\pi''/\pi'$, а в случае соотношения (6) – $\underline{\theta}_0=\overline{\theta}_0=W$, $\underline{\theta}_i=\overline{\theta}_i=\max[\min\{\Phi_{G_{kj}}(\underline{p})|j=1,\ldots,i\},\ \pi''/\pi']$ для $i=1,\ldots,l_k$, где W – достаточно большое число. Так как задача (46), (47) решается при фиксированных значениях y' и y'', то в случае соотношений (4) и (6) ее решение может быть получено в результате решения серии независимых подзадач.

Поскольку каждый из подграфов G_{kj}, $j=1,\ldots,l$, в свою очередь, является последовательно-параллельным, то к его подграфам также применимы рассмотренные методы решения возникающих подзадач.

Таким образом, решение исходной задачи **C** может быть получено в результате применения рекурсивной процедуры последовательно-параллельной декомпозиции, на каждом шаге которой получаемая подзадача решается алгоритмами динамического программирования. Следует отметить, что в общем случае на шаге последовательной декомпозиции алгоритм динамического программирования работает в двухмерном пространстве состояний, а на шаге параллельной декомпозиции – только в одномерном.

Следует отметить, что требование существования нейтральных элементов $1(\otimes)$ и $1(\circ)$ не является обязательным. Оно только позволяет упростить изложение материала, поскольку в противном случае необходимо было бы выделять задачи для подграфов G_1 и G_{k1}.

Заключение

В статье предложена декомпозиционная схема решения задачи оптимизации параметров технических систем с последовательно-параллельной структурой, в которой целевая функция представляет собой общую сумму «весов» дуг и вершин орграфа. Разработаны алгоритмы проверки совместности ограничений, базирующиеся на представлении последовательно-параллельных орграфов в виде его дерева декомпозиции. Показано, что для решения получаемых подзадач могут быть использованы методы динамического программирования.

Список литературы

1. Гущинский, Н.Н. Поддержка принятия решений при проектировании силовых трансмиссий / Н.Н. Гущинский, Г.М. Левин, А.Б. Долгий. – Минск : Белорусская наука, 2006. – 262 с.

2. Левин, Г.М. Декомпозиционные методы оптимизации проектных решений / Г.М. Левин, В.С. Танаев. – Минск : Наука и техника, 1978.– 240 с.

3. Лепин, В.В. Алгоритмы для нахождения мультикликовой и бикликовой степени последовательно-параллельного графа / В.В. Лепин // Тр. И-та матем. – 2010. – Т. 18, № 2. – С. 60–78.

4. Eppstein, D. Parallel recognition of series-parallel graphs / D. Eppstein // Information and Computation. – 1992. – Vol. 98. – P. 41–55.

5. Schoenmakers, B. A new algorithm for the recognition of series parallel graphs / B. Schoenmakers // Technical report, No. CS-59504 / Centrum voor Wiskunde en Informatica. – Amsterdam, The Netherlands [Electronic resource]. – 1995. – Mode of access : ttp://www.cwi.nl/ftp/CWIreports/AA/CS-R9504.ps.Z. – Date of access : 22.09.2012.

Поступила 21.06.2012

Объединенный институт проблем
информатики НАН Беларуси,
Минск, Сурганова, 6
e-mail:gyshin@newman.bas-net.by

N.N. Guschinsky

DECOMPOSITION APPROACH TO OPTIMIZE ARC PARAMETERS OF SERIES-PARALLEL DIGRAPHS

An optimization problem of engineering systems is considered. It is supposed that system structure has been already defined and it can be presented by series-parallel digraphs. Each arc of the digraph is assigned design parameters of the system and its operation conditions are modeled by equality and inequality constraints on monotone functions of design parameters. The objective function is quasi-separable and depends both on arc parameters and their total values.

УДК 681.32

Н.Г. Егорова, Ю.Н. Сотсков, А.А. Косенков

ПЕРЕСТАНОВКА С НАИБОЛЬШИМ ПАРАЛЛЕЛЕПИПЕДОМ УСТОЙЧИВОСТИ ДЛЯ ОБСЛУЖИВАНИЯ ТРЕБОВАНИЙ С ИНТЕРВАЛЬНЫМИ ДЛИТЕЛЬНОСТЯМИ ОПЕРАЦИЙ

Рассматривается задача минимизации суммы взвешенных моментов завершения обслуживания требований одним прибором при условии, что для каждой длительности обслуживания требования заданы нижняя и верхняя границы возможных значений. Разрабатывается алгоритм сложности $O(n\log n)$ для построения перестановки с наибольшей размерностью и наибольшим объемом параллелепипеда устойчивости.

Введение

Рассматривается задача построения оптимального расписания обслуживания требований множества $J=\{J_1,J_2,...,J_n\}$ одним прибором с критерием $\sum w_iC_i$ минимизации суммы взвешенных моментов C_i завершения обслуживания требований $J_i\in J$ при условии, что на момент построения расписания известны только нижняя граница $a_i>0$ и верхняя граница $b_i\ge a_i$ возможных длительностей p_i обслуживания требований $J_i\in J$. Сформулированная задача является неопределенной и обозначается $1|a_i\le p_i\le b_i|\sum w_iC_i$ в соответствии с трехпозиционной формой $\alpha|\beta|\gamma$ классификации задач теории расписаний [1].

Задача $1|a_i\le p_i\le b_i|\sum w_iC_i$ возникает, например, при планировании рабочего времени на некоторый период времени при условии, что множество планируемых к выполнению работ заранее определено и существенно не меняется в ходе реализации расписания. Для длительностей планируемых работ на момент построения расписания известны только диапазоны (отрезки) их возможных действительных значений. Критерий $\sum w_iC_i$ можно рассматривать как суммарный показатель эффективности выполнения работником заданного множества работ [2].

1. Постановка задачи

В обслуживающую систему, состоящую из одного прибора, в момент времени $t=0$ поступает множество требований $J=\{J_1,J_2,...,J_n\}$. Каждое требование $J_i\in J$ должно быть обслужено прибором без прерываний в течение времени $p_i\in[a_i,b_i]$. Здесь и далее p_i обозначает случайную величину, которая в процессе реализации расписания может оказаться равной любому действительному числу, принадлежащему заданному отрезку $[a_i,b_i]$. Закон распределения случайной величины p_i на интервале (a_i,b_i) к моменту построения расписания неизвестен. Каждому требованию $J_i\in J$ приписан вес $w_i>0$, определяющий относительную значимость этого требования. В задаче $1|a_i\le p_i\le b_i|\sum w_iC_i$ требуется построить расписание (перестановку) обслуживания требований множества J, при котором взвешенная сумма $\sum\limits_{i=1}^{n}w_iC_i$ моментов завершения обслуживания требований принимает наименьшее значение.

Если верхняя граница $a_i>0$ и нижняя граница b_i длительностей обслуживания каждого требования совпадают, т. е. $b_i=a_i$, то неопределенная задача $1|a_i\le p_i\le b_i|\sum w_iC_i$ превращается в детерминированную задачу $1||\sum w_iC_i$, для решения которой существует полиномиальный алгоритм сложности $O(n\log n)$ [3]. Множество допустимых векторов длительностей обслуживания требований $p=\{p_1,p_2,...,p_n\}$ обозначим

$$T = \{p \in \mathrm{R}_+^{\mathrm{n}} : a_i \le p_i \le b_i, i \in \{1,...,n\}\},$$

где $\mathrm{R}_+^{\mathrm{n}}$ – множество всех неотрицательных действительных векторов размерности n.

Вектор $p \in T$ возможных длительностей обслуживания требований будем называть сценарием. Для неопределенной задачи $1 \mid a_i \le p_i \le b_i \mid \sum w_i C_i$ может не существовать единственной перестановки, которая является оптимальной для всех детерминированных задач $1 \| \sum w_i C_i$, получаемых из неопределенной задачи в результате фиксации того или иного сценария из множества T. Пусть S – множество всех $n!$ перестановок $\pi_i = (J_{i_1}, J_{i_2}, ..., J_{i_n})$ требований множества J: $S = \{\pi_1, \pi_2, ..., \pi_{n!}\}$. Под решением неопределенной задачи $1 \mid a_i \le p_i \le b_i \mid \sum w_i C_i$ будем подразумевать минимальное (по включению) множество перестановок $S(T) \subseteq S$ согласно следующему определению [4].

О п р е д е л е н и е 1. Решением неопределенной задачи $1 \mid a_i \le p_i \le b_i \mid \sum w_i C_i$ будем называть минимальное по включению множество перестановок $S(T) \subseteq S$, такое, что для любого допустимого сценария $p \in T$ существует перестановка $\pi_j \in S(T)$, которая является оптимальной для детерминированной задачи $1 \| \sum w_i C_i$ со сценарием p.

2. Перестановки с наибольшим параллелепипедом устойчивости

Для практической реализации из множества $S(T)$ предлагается выбирать перестановку $\pi_k = (J_{k_1}, J_{k_2}, ..., J_{k_n}) \in S(T)$ с максимальной размерностью и максимальным объемом параллелепипеда устойчивости. Для определения параллелепипеда устойчивости оптимальной перестановки требований множества $J = \{J_1, J_2, ..., J_n\}$ введем следующие обозначения.

Через $1 \mid p \mid \sum w_i C_i$ обозначим детерминированную задачу со сценарием $p \in T$. Пусть $J^-[k_i] = \{J_{k_1}, ..., J_{k_{i-1}}\}$ и $J^+[k_i] = \{J_{k_{i+1}}, ..., J_{k_n}\}$. Множество S_{k_i} – это множество перестановок $(\pi(J^-[k_i]), J_{k_i}, \pi(J^+[k_i])) \in S$ требований множества J. Перестановка $\pi(J')$ – это перестановка требований подмножества J' множества J. По аналогии с определением из работы [5] дадим следующее определение параллелепипеда устойчивости оптимальной перестановки $\pi_k = (J_{k_1}, J_{k_2}, ..., J_{k_n}) \in S$.

О п р е д е л е н и е 2. Пусть $\pi_k = (J_{k_1}, J_{k_2}, ..., J_{k_n}) \in S$ – перестановка, оптимальная хотя бы при одном сценарии из множества T. Максимальный отрезок $[l_{k_i}, u_{k_i}] \subseteq [a_{k_i}, b_{k_i}]$ будем называть *максимальной вариацией* длительности требования J_{k_i} в перестановке π_k, если для любой перестановки $\pi_e \in S_{k_i}$ и любого сценария $p = (p_1, ..., p_n) \in T$, при котором она оптимальна, перестановка π_e остается оптимальной и для любого сценария

$$p' \in \{\times_{j=1}^{k_i - 1}[p_j, p_j] \times [l_{k_i}, u_{k_i}] \times_{j=k_i+1}^{n}[p_j, p_j]\} \subseteq T,$$

причем для любого сценария $p'' = (p_1'', ..., p_n'') \in T$, $p_{k_i}'' \in [l_{k_i}, u_{k_i}]$, существует оптимальная для задачи $1 \mid p'' \mid \sum w_i C_i$ перестановка $\pi_v \in S_{k_i}$. Пусть N_k – множество индексов i всех требований $J_i \in J$ с непустыми максимальными вариациями их длительностей. Декартово произведение $SB(\pi_k, T) = \times_{k_i \in N_k}[l_{k_i}, u_{k_i}] \subseteq T$ будем называть многогранником (параллелепипедом) устойчивости перестановки π_k. Если не существует сценария $p \in T$, при котором перестановка π_k является оптимальной для задачи $1 \mid p \mid \sum w_i C_i$, то полагаем $SB(\pi_k, T) = \varnothing$.

Для нахождения параллелепипеда $SB(\pi_k, T)$ для фиксированной перестановки $\pi_k = (J_{k_1}, J_{k_2}, ..., J_{k_n})$ достаточно вычислить максимальную вариацию каждой длительности p_{k_i},

$i \in \{1, 2, ..., n\}$. Нижнюю границу $d_{k_i}^-$ и соответственно верхнюю границу $d_{k_i}^+$ максимальной вариации отношения веса к длительности можно вычислить следующим образом:

$$d_{k_i}^- = \max\left\{\frac{w_{k_i}}{b_{k_i}}, \max_{i<j\le n}\left\{\frac{w_{k_j}}{a_{k_j}}\right\}\right\}; \quad (1)$$

$$d_{k_i}^+ = \min\left\{\frac{w_{k_i}}{a_{k_i}}, \min_{1\le j<i}\left\{\frac{w_{k_j}}{b_{k_j}}\right\}\right\}. \quad (2)$$

Поясним понятие параллелепипеда устойчивости на следующем примере неопределенной задачи с четырьмя требованиями $J = \{J_1, J_2, J_3, J_4\}$, $n = 4$. Пусть исходные данные задачи $1 \mid a_i \le p_i \le b_i \mid \sum w_i C_i$ определены в столбцах 1–4 приведенной ниже таблицы. Значения $d_{k_i}^-$ и $d_{k_i}^+$, вычисленные для перестановки $\pi_k = (J_1, J_2, J_3, J_4)$, даны в столбцах 5 и 6, а диапазоны максимальных вариаций длительностей, при которых сохраняется оптимальность перестановки π_k, – в столбцах 7 и 8.

Исходные данные для примера задачи $1 \mid a_i \le p_i \le b_i \mid \sum w_i C_i$

i	a_i	b_i	w_i	d_i^-	d_i^+	w_i / d_i^+	w_i / d_i^-
1	4	5	400	90	100	4	$4\frac{4}{9}$
2	6	9	540	60	80	$6\frac{3}{4}$	9
3	4	10	200	40	50	4	5
4	3	4	120	30	20	-	-

Максимальные вариации отношений весов к длительностям требований, при которых сохраняется оптимальность перестановки π_k, на рис. 1 заштрихованы.

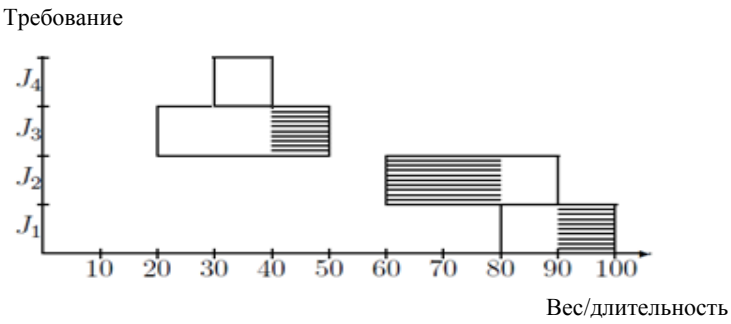


Рис. 1. Максимальные вариации отношений весов к длительностям обслуживания требований

Параллелепипед устойчивости перестановки $\pi_k = (J_1, J_2, J_3, J_4)$ изображен на рис. 2.

Размерность $|N_k|$ параллелепипеда устойчивости $SB(\pi_k, T)$ равна количеству требований J_{k_i}, для которых выполняется нестрогое неравенство $d_{k_i}^- \le d_{k_i}^+$. Обозначим через n^k число требований, для которых выполняются соотношения $d_{k_i}^- = d_{k_i}^+$ и $a_{k_i} < b_{k_i}$. Относительный объем $VolSB(\pi_t, T)$ многогранника устойчивости будет вычисляться как произведение величин

$(u_{k_i} - l_{k_i})/(b_{k_i} - a_{k_i})$ для всех требований J_{k_i} с отрезками $[l_{k_i}, u_{k_i}]$ максимальных вариаций длительностей, для которых выполняется строгое неравенство $l_{k_i} < u_{k_i}$.

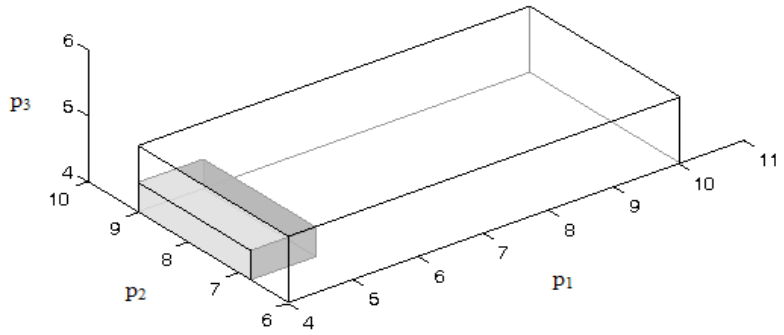


Рис. 2. Многогранник T, в котором выделен параллелепипед устойчивости перестановки $\pi_k = (J_1, J_2, J_3, J_4)$

Наибольший интерес для практического использования представляют перестановки $\pi_k \in S$, имеющие наибольшую размерность $|N_k|$ параллелепипеда устойчивости $SB(\pi_k, T)$. Если таких перестановок несколько, то следует выбирать перестановки с наибольшим относительным объемом многогранника устойчивости $SB(\pi_t, T)$ среди всех перестановок $\pi_k \in S$ с наибольшими размерностями многогранников устойчивости $|N_k| = |N_t|$ и минимальным количеством n^k максимальных вариаций с нулевой длиной, т. е. перестановку (перестановки) с наибольшим относительным объемом соответствующего многогранника размерности $|N_t| - n^k$. (Относительный объем $VolSB(\pi_t, T)$ вычисляется как произведение величин $(u_{k_i} - l_{k_i})/(b_{k_i} - a_{k_i})$ для всех требований J_{k_i} с отрезками $[l_{k_i}, u_{k_i}]$ максимальных вариаций длительностей, для которых выполняется строгое неравенство $l_{k_i} < u_{k_i}$.) Множество всех таких перестановок $\pi_t \in S$ обозначим $S^{\max}$.

Для построения перестановки $\pi_t \in S^{\max}$ разработан полиномиальный алгоритм. Рассмотрим свойства многогранника устойчивости, которые позволили разработать алгоритм минимальной асимптотической сложности $O(n \log n)$.

Свойство 1. *Для любых требований $J_i \in J$ и $J_v \in J$, $v \neq i$, в перестановке π_k выполняется равенство*

$$\left(\frac{w_i}{u_i}, \frac{w_i}{l_i}\right) \cap \left[\frac{w_v}{b_v}, \frac{w_v}{a_v}\right] = \varnothing .$$

Свойство 1 следует непосредственно из формул (1) и (2).

Используя свойство 1, покажем, как можно определить порядок требования $J_i \in J$ относительно требования $J_v \in J$ для любого $v \neq i$ в перестановке $\pi_t = \{J_{t_1}, \ldots, J_{t_n}\}$. Рассмотрим все три возможных случая пересечения открытого интервала $\left(\frac{w_i}{b_i}, \frac{w_i}{a_i}\right)$ с отрезком $\left[\frac{w_v}{b_v}, \frac{w_v}{a_v}\right]$:

$$\text{(I)} \quad \frac{w_v}{b_v} < \frac{w_i}{b_i}, \quad \frac{w_v}{a_v} < \frac{w_i}{a_i}; \tag{3}$$

$$\text{(II)}\ \frac{w_v}{b_v}=\frac{w_i}{b_i},\quad \frac{w_v}{a_v}=\frac{w_i}{a_i}; \tag{4}$$

$$\text{(III)}\ \frac{w_v}{b_v}\ge\frac{w_i}{b_i},\quad \frac{w_v}{a_v}\le\frac{w_i}{a_i}. \tag{5}$$

Предполагается, что хотя бы одно из неравенств (5) является строгим.

В случае (III) множество требований J_v, для которых выполняются неравенства (5) относительно требования J_i, будем обозначать $J(i)$.

Свойство 2. *Для случая (I) существует перестановка $\pi_t \in S^{\max}$, в которой требование J_i предшествует требованию J_v.*

Доказательство. В случае (I) искомый порядок требований J_v и J_i в перестановке $\pi_t \in S^{\max}$ может быть определен строгим неравенством из (3): требование J_i предшествует требованию J_v в перестановке π_t. Действительно, если требование J_v предшествует требованию J_i, то обе максимальные вариации $[l_i,u_i]$ и $[l_v,u_v]$ длительностей обслуживания требований p_i и p_v в перестановке $\pi_k \in S$ пусты (это следует из равенств (1) и (2)).

Свойство 3. *Для случая (II) существует перестановка $\pi_t \in S^{\max}$, в которой требования J_i и J_v расположены рядом (одно непосредственно за другим): $i=t_r$, $v=t_{r+1}$.*

Доказательство. Если требования J_i и J_v расположены рядом, то *максимальная вариация* $[l_u,u_u]$ длительностей обслуживания требования p_u для любого требования $J_u \in J\setminus\{J_i,J_v\}$ в перестановке π_k не меньше, чем в случае, когда некоторое требование $J_w \in J\setminus\{J_i,J_v\}$ расположено между требованиями J_i и J_v.

Если выполняются равенства (4), то можно ограничить поиск перестановки $\pi_t \in S^{\max}$ подмножеством перестановок множества S с расположенными рядом требованиями J_i и J_v (свойство 3). Кроме того, порядок таких требований $\{J_i,J_v\}$ не влияет на объем многогранника устойчивости и на его размер.

Замечание. Согласно свойству 3 при поиске перестановки $\pi_t \in S^{\max}$ пару требований $\{J_i,J_v\}$, для которых выполняется (4), можно рассматривать как одно требование.

Оценим количество *максимальных вариаций* $[l_i,u_i]$ длительностей обслуживания требования $J_i \in J$, которые могут возникнуть в перестановках из множества S. При формулировке и доказательстве свойств 4(*ii*) и 5 будем предполагать, что множества требований, попарно удовлетворяющих неравенствам (4), рассматриваются как одно требование (см. замечание).

Свойство 4. *(i) Для фиксированной перестановки $\pi_k \in S$ требование $J_i \in J$ может иметь не более одной максимальной вариации $[l_i,u_i]$ длительности $p_i \in [a_i,b_i]$ в перестановке π_k.*

(ii) Для всего множества перестановок S только в случае (III) требование $J_i \in J$ может иметь более чем одну ($|J(i)|+1>1$) максимальную вариацию $[l_i,u_i]$ длительности $p_i \in [a_i,b_i]$ для какой-либо перестановки из множества S.

Доказательство. Часть (*i*) свойства 4 следует из того факта, что непустая *максимальная вариация* $[l_i,u_i]$ длительностей обслуживания требования $J_i \in J$ (если она существует) однозначно определена подмножеством требований $J^-(i)$, расположенных перед требованием J_i в перестановке π_k, и подмножеством требований $J^+(i)$, расположенных после требования J_i. Подмножества $J^-(i)$ и $J^+(i)$ однозначно определены для фиксированной перестановки $\pi_k \in S$ и фиксированного требования $J_i \in J$.

Часть (*ii*) свойства 4 следует из следующих соображений. Если открытый интервал $\left(\frac{w_i}{b_i},\frac{w_i}{a_i}\right)$ не пересекается с отрезком $\left[\frac{w_v}{b_v},\frac{w_v}{a_v}\right]$ для каждого требования $J_v \in J$, то существует перестановка $\pi_t \in S^{\max}$ с максимальной вариацией $[l_i,u_i]=[a_i,b_i]$ длительности обслуживания требования $J_i \in J$.

Каждое требование $J_v \in J$ с $\left(\frac{w_i}{b_i},\frac{w_i}{a_i}\right)\bigcap\left[\frac{w_v}{b_v},\frac{w_v}{a_v}\right]\neq\varnothing$, удовлетворяющее неравенствам (3) (случай (I)), может уменьшить максимальную вариацию $[l_i,u_i]$ длительности обслуживания требования $J_i \in J$, но не может породить новых максимальных вариаций длительности обслуживания требования $J_i \in J$.

В случае (III) требование J_v, удовлетворяющее неравенствам (5), может породить новую максимальную вариацию длительности обслуживания только требования J_i, удовлетворяющего неравенствам (5). Таким образом, количество элементов $|L(i)|$ множества $L(i)$ таких отрезков $[l_i,u_i]$ не превышает числа $|J(i)|+1$.

Обозначим через L' упорядоченное множество $(i,[l_i,u_i])$, состоящее из всех отрезков $[l_i,u_i]$, каждый из которых является наибольшим из максимальных вариаций длительностей обслуживания p_i требования $J_i \in J$ по всем перестановкам из множества *S*. Пусть L – множество $(i,[l_i,u_i])$, состоящее из всех отрезков $[l_i,u_i]$, являющихся максимальными вариациями длительностей обслуживания p_i всех требований $J_i \in J$ в какой-либо перестановке из множества *S*.

Свойство 5. $|L'|\le n$.

Доказательство. Поскольку множество L' содержит не более одной максимальной вариации длительностей обслуживания p_i каждого требования $J_i \in J$, то для него также выполняется неравенство $|L'|\le n$.

Нетрудно убедиться в том, что доказанные утверждения позволяют обосновать следующий $O(n\log n)$-алгоритм построения перестановки, многогранник устойчивости которой имеет наибольшую размерность и наибольший относительный объем.

Алгоритм «Многогранник»

Вход: отрезки $[a_i;b_i]$, веса w_i, требования $J_i \in J$.

Выход: перестановка $\pi_t \in S^{\max}$, многогранник устойчивости $SB(\pi_t,T)$.

Шаг 1. Построить списки $M(U)=(J_{u_1},...,J_{u_n})$ и $w(U)=\left(\frac{w_{u_1}}{b_{u_1}},...,\frac{w_{u_n}}{b_{u_n}}\right)$ в порядке невозрастания значений w_{u_r}/b_{u_r}. Если на некотором подмножестве требований значения w_{u_r}/b_{u_r} совпадают, то требования этого подмножества сортируются в порядке неубывания значений w_{u_r}/a_{u_r}.

Шаг 2. Построить списки $M(L)=(J_{l_1},...,J_{l_n})$ и $w(L)=\left(\frac{w_{l_1}}{a_{l_1}},...,\frac{w_{l_n}}{a_{l_n}}\right)$ в порядке невозрастания значений w_{l_r}/a_{l_r}. Если на некотором подмножестве требований значения w_{l_r}/a_{l_r} совпадают, то требования этого подмножества сортируются в порядке неубывания значений w_{l_r}/b_{l_r}.

Шаг 3. Если существуют множества требований с одинаковыми отрезками $\left[\frac{w_v}{b_v}, \frac{w_v}{a_v}\right]$, то оставить для дальнейшего рассмотрения только одно из них (свойство 3). Положить n_1 равным количеству оставшихся требований.

Шаг 4. Положить $j = 1$.

Шаг 5. Если $j \le n_1$, то перейти к шагу 6, иначе перейти к шагу 12.

Шаг 6. Если $J_{u_j} = J_{l_j}$, то требование J_{l_j} должно быть расположено в позиции j в перестановке $\pi_t \in S^{\max}$, перейти к шагу 7. В противном случае требование $J_{l_j} = J_i$ удовлетворяет неравенствам (5), перейти к шагу 8.

Шаг 7. Положить $j := j + 1$. Перейти к шагу 5.

Шаг 8. Построить множество $J(i) = \{J_{u_j}, ..., J_{u_{k-1}}\}$ всех требований J_v, удовлетворяющих неравенствам (5), где $J_i = J_{l_j} = J_{u_k}$. Если в позициях $j, ..., k$ в списках $M(L)$ и $M(U)$ стоят разные наборы требований, то переместить требования множества $J(i)$ в позиции $j+1, ..., k$ в списках $M(L)$ и $w(L)$, сдвинув остальные требования соответственно вправо.

Шаг 9. Выбрать наибольший диапазон $[l_{l_j}, u_{l_j}]$ из всех диапазонов, построенных для требования $J_{l_j} = J_i$, и разбить множество $J(i)$ на подмножества $J^-(i)$ и $J^+(i)$, определяющие диапазон $[l_{l_j}, u_{l_j}]$ (см. алгоритм «Диапазон»).

Шаг 10. В перестановке $\pi_t \in S^{\max}$ поместить требования множеств $J^-(i)$, $\{J_i\}$, $J^+(i)$ в позиции $j, ..., k$. Положить $j = k + 1$. Перейти к шагу 5.

Шаг 11. Удаленные на шаге 3 требования разместить в перестановке $\pi_t \in S^{\max}$ рядом с требованием с таким же отрезком $\left[\frac{w_v}{b_v}, \frac{w_v}{a_v}\right]$.

Шаг 12. Построить многогранник устойчивости $SB(\pi_t, T)$ по алгоритму STABOX, предложенному в статье [5].

При представлении упорядоченных множеств $M(L)$ и $w(L)$ в виде списочных структур перемещение требований множества $J(i)$ в позиции $j+1, ..., k$ в списках $M(L)$ и $w(L)$ на шаге 8 со сдвигом остальных требований вправо можно реализовать за один перебор требований множества $J(i)$. Для этого перед выполнением шага 1 для элементов списка $M(U)$ необходимо сопоставить им адреса соответствующих элементов списка $M(L)$.

При выполнении шага 9 следует учитывать тот факт, что требования множества $J(i) = \{J_{u_j}, ..., J_{u_{k-1}}\}$ в списке $M(U)$ расположены в порядке невозрастания значений w_{u_r} / b_{u_r}, $j \le r \le k-1$, в силу чего наибольший диапазон $[l_{l_j}, u_{l_j}]$ можно найти с помощью алгоритма «Диапазон» за один перебор требований множества $J(i)$. Для этого нужно перебирать требования в порядке неубывания значений w_{u_r} / b_{u_r} и определять отрезки $[d^-_{l_j}, d^+_{l_j}]$ между требованиями, смежными в списке $M(U)$. Нижнюю границу $d^-_{l_j}$ отрезка $[d^-_{l_j}, d^+_{l_j}]$ между требованиями J_{u_t} и $J_{u_{t-1}}$ можно определить по формуле

$$d^-_{l_j} := \max\{d^-_{l_j}, w_{u_t} / a_{u_t}\}. \tag{6}$$

Соответствующую верхнюю границу можно получить по формуле

$$d^+_{l_j} = w_{u_{t-1}} / b_{u_{t-1}}. \tag{7}$$

Графически этот случай показан на рис. 3.

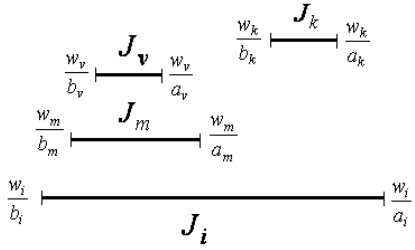


Рис. 3. Интервалы отношений весов к длительностям обслуживания требований

Построим списки $M(U)$ и $M(L)$ и множество требований $J(i)$:

$$M(U) = \underbrace{(J_k, J_v, J_m, \boldsymbol{J_i})}_{J(i)};$$

$$M(L) = (\boldsymbol{J_i}, J_k, J_m, J_v);$$

$$J(i) = \{J_k, J_v, J_m\}.$$

Требования множества $J(i) = \{J_k, J_v, J_m\}$ в списке $M(U)$ расположены в порядке невозрастания значений w_{u_r} / b_{u_r}. Перебирая требования множества $J(i)$ в порядке неубывания значений w_{u_r} / b_{u_r} (т. е. в обратном порядке: (J_m, J_v, J_k)), получаем следующие четыре вариации отношения весов к длительностям обслуживания требований:

1) $\left[\frac{w_i}{b_i}, \frac{w_m}{b_m}\right]$;

2) $\varnothing$ (поскольку $d_i^- = \frac{w_m}{a_m} > \frac{w_v}{b_v} = d_i^+$);

3) $\left[\max\left\{\frac{w_m}{a_m}, \frac{w_v}{a_v}\right\}, \frac{w_k}{b_k}\right] = \left[\frac{w_m}{a_m}, \frac{w_k}{b_k}\right]$;

4) $\left[\max\left\{\frac{w_m}{a_m}, \frac{w_k}{a_k}\right\}, \frac{w_i}{a_i}\right] = \left[\frac{w_k}{a_k}, \frac{w_i}{a_i}\right]$.

Из полученных интервалов выбирается тот, который дает самую большую максимальную вариацию длительностей обслуживания требований.

Следует отметить, что фиксированный порядок следования требований в перестановке требований множества $J \setminus (J(i) \cup J_{l_j})$ может привести к необходимости рассмотрения в алгоритме «Диапазон» сокращенных диапазонов $[\hat{a}_z, \hat{b}_z]$, $[\hat{a}_z, \hat{b}_z] \subseteq [a_z, b_z]$, $z \in \{u_j, ..., u_{k-1}, l_j\}$ вместо заданных отрезков $[a_z, b_z]$ возможных длительностей обслуживания требований $J_z \in J(i) \cup J_{l_j}$. Границы отношений весов к длительностям для сокращенных диапазонов можно определить по формулам

$$w_z / \hat{a}_z = \min\{w_z / a_z, w_{u_{j-1}} / b_{u_{j-1}}\},\ w_z / \hat{b}_z = \max\{w_z / b_z, w_{l_{k+1}} / a_{l_{k+1}}\}. \quad (8)$$

Алгоритм «Диапазон»

Вход: отрезки $[a_i, b_i]$, веса w_i, $J_i \in J$, множество $J(i) = \{J_{u_j}, ..., J_{u_{k-1}}\}$ для требования J_{l_j}, список $M(U) = (J_{u_1}, ..., J_{u_n})$ требований, расположенных в порядке невозрастания значений w_{u_r} / b_{u_r}, список $M(L) = (J_{l_1}, ..., J_{l_{j-1}}, J_{l_j}, ..., J_{l_{k-1}}, J_{l_k}, J_{l_{k+1}}, ..., J_{l_n}) = (J_{l_1}, ..., J_{l_{j-1}}, J_{u_k}, ..., J_{u_{k-1}}, J_{l_{k+1}}, ..., J_{l_n})$ требований, расположенных в порядке невозрастания значений w_{l_r} / a_{l_r} для требований $J_{l_1}, ..., J_{l_{j-1}}$ и $J_{l_{k+1}}, ..., J_{l_n}$.

Выход: наибольший диапазон $[l_{l_j}, u_{l_j}]$, подмножества $J^-(i)$ и $J^+(i)$.

Шаг 1. Найти сокращенный диапазон $[a, b] = [w_{l_j} / \hat{b}_{l_j}, w_{l_j} / \hat{a}_{l_j}]$ отношения веса к длительностям обслуживания требования J_{l_j} по формулам (8). Если $[a, b] \neq \varnothing$, то перейти к шагу 2, иначе положить $[l_{l_j}, u_{l_j}] = \varnothing$, $J^-(i) = J(i)$, $J^+(i) = \varnothing$, перейти к шагу 9.

Шаг 2. Положить $t = k - 1$, $t_1 = t + 1$, $d^-_{l_j} = w_{l_j} / b_{l_j}$, $d^+_{l_j} = w_{u_t} / b_{u_t}$, $[l_{l_j}, u_{l_j}] = [w_{l_j} / d^+_{l_j}, w_{l_j} / d^-_{l_j}]$, $J^-(i) = J(i)$, $J^+(i) = \varnothing$.

Шаг 3. Если $t < j$, то перейти к шагу 7.

Шаг 4. Положить $d^-_{l_j} = \max\{d^-_{l_j}, w_{u_t} / a_{u_t}\}$. Если $t > j$, то положить $d^+_{l_j} = w_{u_{t-1}} / b_{u_{t-1}}$, в противном случае положить $d^+_{l_j} = b$.

Шаг 5. Если диапазон $[d^-_{l_j}, d^+_{l_j}] \bigcap [a, b]$ не пуст, а длина соответствующего отрезка $[l'_{l_j}, u'_{l_j}]$ больше длины отрезка $[l_{l_j}, u_{l_j}]$, то положить $[l_{l_j}, u_{l_j}] = [l'_{l_j}, u'_{l_j}]$, $t_1 = t$.

Шаг 6. Положить $t := t - 1$. Перейти к шагу 3.

Шаг 7. Если $t_1 = j$, то положить $J^-(i) = \varnothing$, $J^+(i) = J(i)$.

Шаг 8. Если $j < t_1 < k$, то положить $J^-(i) = (J_{u_j}, ..., J_{u_{t_1}})$, $J^+(i) = (J_{u_{t_1+1}}, ..., J_{u_{k-1}})$.

Шаг 9. Конец алгоритма.

Шаг 3 алгоритма «Многогранник» основывается на свойстве 3 и приведенном замечании, шаг 6 – на свойстве 2, шаги 8, 9 – на свойстве 4 (часть (*ii*)). Шаг 10 основан на теореме, которая доказана ниже.

Свойство 6. *Существует перестановка* $\pi_t \in S^{\max}$ *с множеством пар* $(i, [l_i, u_i])$, *где* i – *номер требования,* $[l_i, u_i]$ – *максимальная вариация длительности* p_i, $J_i \in J$, *такой, что множество* $(i, [l_i, u_i])$ *совпадает с множеством* $L' \subseteq L$.

Доказательство. Для каждого требования J_i максимальные вариации $[l_i, u_i]$ можно найти как отрезки максимальной длины, для которых выполняется равенство $\left(\frac{w_i}{u_i}, \frac{w_i}{l_i}\right) \bigcap \left[\frac{w_v}{b_v}, \frac{w_v}{a_v}\right] = \varnothing$ для всех требований $J_v \in J \setminus J_i$. Поскольку для всех требований J_i имеет место включение $[l_i, u_i] \subseteq [a_i, b_i]$, то $\left(\frac{w_i}{u_i}, \frac{w_i}{l_i}\right) \bigcap \left(\frac{w_v}{u_v}, \frac{w_v}{l_v}\right) = \varnothing$. Для каждого требования можно выбрать наибольшую из максимальных вариаций. При этом для каждого из требований с непустой максимальной вариацией однозначно определяются подмножества предшествующих ему и следующих за ним требований. Поскольку интервалы $\left(\frac{w_i}{u_i}, \frac{w_i}{l_i}\right)$ не пересекаются, можно построить перестановку, удовлетворяющую условиям свойства 6. Такая перестановка

будет иметь многогранник устойчивости с наибольшей размерностью, наибольшим относительным объемом и минимальным количеством максимальных вариаций нулевой длины.

Для доказательства теоремы проанализируем алгоритм «Многогранник». На шагах 1–3 и 6 все требования $J^t = \{J_i \mid J_{u_j} = J_i = J_{l_j}\}$ расположены одинаково в обоих списках $M(U)$, $M(L)$ и имеют фиксированное расположение в перестановке $\pi_t \in S^{\max}$. Расположение остальных требований $J \setminus J^t$ в перестановке π_t определяется на шагах 8, 9. Фиксированный порядок следования требований J^t может сократить первоначальный диапазон $[a_i, b_i]$ требований $J_i \in J \setminus J^t$, обозначим его через $[\hat{a}_i, \hat{b}_i]$ Таким образом, на шагах 8, 9 для требований $J_i \in J \setminus J^t$ вместо диапазонов $[a_i, b_i]$ в алгоритме «Многогранник» рассматриваются, возможно, уменьшенные диапазоны $[\hat{a}_i, \hat{b}_i]$.

Теорема. *Алгоритм «Многогранник» строит перестановку* $\pi_t \in S^{\max}$*, такую, что размерность* $|N_t|$ *многогранника устойчивости* $SB(\pi_t, T) = \times_{t_i \in N_t}[l_{t_i}, u_{t_i}] \subseteq T$ *является наибольшей среди всех перестановок S, а относительный объем многогранника устойчивости* $SB(\pi_t, T)$ *является наибольшим среди всех перестановок* $\pi_k \in S$*, имеющих наибольшую размерность их многогранников устойчивости и минимальное количество* n^k *максимальных вариаций нулевой длины.*

Доказательство. Покажем, что построенная по алгоритму «Многогранник» перестановка удовлетворяет свойству 6. Согласно шагам 1–7 алгоритма «Многогранник», если для требования J_i выполняются условия шага 6, то для него и каждого из следующих за ним в перестановке π_t требований выполняются условия (3). В этом случае требование имеет не более одной максимальной вариации, и она может быть непустой только тогда, когда требование J_i расположено перед всеми последующими требованиями в перестановке π_t.

Обозначим через J^* множество всех требований J_i, для которых существуют требования J_v, такие, что требования J_i и J_v попарно удовлетворяют неравенствам (5), и которые сами не принадлежат ни одному из множеств $J(i)$. Тогда для всех требований $J_i \in J^*$, согласно свойству 4 (часть (*ii*)) и шагам 8–11 алгоритма «Многогранник», множества $J^-(i)$ и $J^+(i)$, соответствующие наибольшей максимальной вариации требования J_i, определяются перебором множеств $J(i)$. В этом случае максимальные вариации требования определяются с учетом предшествующих ему и следующих за ним требований. Поэтому требования J_v, принадлежащие одновременно нескольким множествам $J(i)$, могут рассматриваться в паре с любым из множеств $J_i \in J^*$. После выполнения шага 8 для требования J_i условия (3) должны выполняться для требования J_i и для каждого из требований J_v, включенных в перестановку π_t на более поздних итерациях.

3. Результаты вычислительных экспериментов

Проведенные вычислительные эксперименты на множестве случайно сгенерированных примеров показали высокую эффективность разработанного алгоритма как по времени решения задачи $1 \mid a_i \le p_i \le b_i \mid \sum w_i C_i$, так и по качеству получаемых расписаний (перестановок) относительно критерия $\sum w_i C_i$. Эксперименты проводились на персональном компьютере Pentium(R) 4, CPU 3 GHz, 1,00 GB RAM.

Построение примеров проводилось по следующей схеме. Для каждого требования случайным образом генерировалась середина C интервала длительностей его обслуживания из диапазона [1, 100]. Нижняя граница длительности обслуживания требования вычислялась по формуле $a_i = C \cdot (1 - L/100)$, верхняя – по формуле $b_i = C \cdot (1 + L/100)$, где L – погрешность длительности обслуживания требования в процентах. Вес каждого требования генерировался в диапазоне [1, 50].

Величина относительной погрешности целевой функции рассчитывалась исходя из значения целевой функции для эвристического расписания, построенного согласно предложенному алгоритму, и соответствующего оптимального значения целевой функции для случайно сгенерированной детерминированной задачи с длительностями из заданных интервалов возможных значений. Для каждой комбинации $n \in \{100,\ 200,\ ...,\ 1000\}\}$ и $L \in \{1,\ 5,\ 10,\ 15,\ 20,\ 25,\ 30,\ 40\}\}$ была решена серия из 100 тестовых примеров $1 \mid a_i \le p_i \le b_i \mid \sum w_i C_i$.

В проведенных экспериментах при $L = 40$ % средняя относительная погрешность Δ целевой функции для каждой серии не превосходила 3,4 % при любом количестве n требований. При $L = 20$ % средняя относительная погрешность Δ целевой функции для каждой серии не превосходила 0,86 % при любом количестве n требований. При $L = 5$ % средняя относительная погрешность Δ целевой функции для каждой серии не превосходила 0,075 % при любом количестве n требований.

Заключение

В статье введено и исследовано понятие параллелепипеда устойчивости оптимальной перестановки обслуживания требований $J = \{J_1, J_2, ..., J_n\}$ (определение 2) для задачи $1 \mid a_i \le p_i \le b_i \mid \sum w_i C_i$ минимизации суммы взвешенных моментов завершения обслуживания требований $J_i \in J$ при условии, что при решении задачи $1 \mid a_i \le p_i \le b_i \mid \sum w_i C_i$ известны только нижняя и верхняя границы возможных длительностей обслуживания требований. Параллелепипед устойчивости является подмножеством шара устойчивости оптимального решения, который исследовался ранее для различных оптимизационных задач (например, в [4, 7, 8]).

Параллелепипед устойчивости может быть построен за $O(n \log n)$ элементарных действий. В статье предложен алгоритм «Многогранник» сложности $O(n \log n)$ для построения перестановки требований $J = \{J_1, J_2, ..., J_n\}$ с наибольшей размерностью и наибольшим объемом параллелепипеда устойчивости.

Полученные результаты предполагается использовать в разрабатываемой новой версии комплекса программ «Расписание», предназначенного для планирования рабочего времени руководящего работника [6]. При разработке эвристических алгоритмов построения расписаний для руководящего работника необходимо снизить риски получения неоптимальных расписаний.

При построении расписания и управления реализацией построенного расписания будут использованы описанные выше алгоритмы «Многогранник» и «Диапазон». Это позволит повысить гарантии того, что полученное расписание с большой вероятностью будет близким к оптимальному расписанию и, следовательно, риски его использования руководящим работником будут минимальны. В конечном итоге это приведет к максимизации прибыли.

Некоторые из полученных в данной статье результатов были представлены в материалах двух международных научных конференций [9, 10].

Список литературы

1. Sequencing and scheduling: Algorithms and complexity / E.L. Lawler [et al.] // Handbooks in Operations Research and Management Science. Logistics of Production and Inventory. – New York, 1993. – P. 445–522.

2. Егорова, Н.Г. Минимизация суммы взвешенных моментов завершения обслуживания требований с интервальными длительностями / Н.Г. Егорова, Ю.Н. Сотсков // Информатика. – 2008. – № 3. – С. 5–16.

3. Smith, W.E. Various Optimizers for Single-Stage Production / W.E. Smith // Naval Research and Logistics Quarterly. – 1956. – Vol. 3, № 1. – P. 59–66.

4. Сотсков, Ю.Н. Теория расписаний. Системы с неопределенными числовыми параметрами / Ю.Н. Сотсков, Н.Ю. Сотскова. – Минск : ОИПИ НАН Беларуси, 2004. – 290 с.

5. Sotskov, Yu.N. Minimizing total weighted flow time under uncertainty using dominance and a stability box / Yu.N. Sotskov, T.-C. Lai // Computers & Operations Research. – 2012. – Vol. 39. – P. 1271–1289.

6. Модели и комплекс программ для планирования рабочего времени / Ю.Н. Сотсков [и др.] // Информатика. – 2007. – № 4. – С. 23–36.

7. Emelichev, V.A. Multicriterial investment problem in conditions of uncertainty and risk / V.A. Emelichev, V.V. Korotkov, K.G. Kuz'min // Journal of Computer and Systems Sciences International. – 2011. – Vol. 50, № 6. – P. 1011–1018.

8. Емеличев, В.А. О радиусе устойчивости эффективного решения векторной квадратичной булевой задачи на узкие места / В.А. Емеличев, В.В. Коротков // Дискретный анализ и исследование операций. – 2011. – Т. 18, № 6. – С. 3–16.

9. The stability box in interval data for minimizing the sum of weighted completion times / Yu.N. Sotskov [et al.]// SIMULTECH 2011: 1st International Conf. on Simulation and Modeling Methodologies. Technologies and Applications, Noordwijkerhout, The Netherlands, 29–31 July 2011. – Portugal : SciTePress – Science and Technology Publications, 2011. – P. 14–23.

10. Егорова, Н.Г. Перестановка с наибольшим параллелепипедом устойчивости для обслуживания требований с неопределенными длительностями операций / Н.Г. Егорова, Ю.Н. Сотсков // Пятая Междунар. науч. конф. «Танаевские чтения». – Минск : ОИПИ НАН Беларуси, 2012. – С. 24–29.

Поступила 11.06.12

Объединенный институт проблем
информатики НАН Беларуси,
Минск, Сурганова, 6
e-mail: NataMog@yandex.by
sotskov@newman.bas-net.by
aklotr809@gmail.com

N.G. Egorova, Yu.N. Sotskov, A.A. Kasiankou

PERMUTATIONS WITH LARGEST STABILITY BOX FOR PROCESSING JOBS WITH INTERVAL OPERATION TIMES

A single machine problem of minimizing the sum of job weighted completion times is considered. It is assumed that only lower and upper bounds for processing time of each job are known. Stability box of an optimal permutation is studied. An $O(n \log n)$ time algorithm is developed for constructing a permutation with stability box having largest dimension and largest volume.

УДК 519.8

В.М. Котов[1], Х. Келлерер[2], Н. Браунер[3], Г. Финке[3]

АЛГОРИТМ ДЛЯ ВЕРСИЙ ЗАДАЧИ $Pm||C_{\max}$ С НЕПОЛНОЙ ИНФОРМАЦИЕЙ

Исследуются две модели с неполной информацией для задач теории расписаний с идентичными процессорами. Предлагается общая параметрическая схема построения решений для таких задач. Достигаются рекордные гарантированные оценки точности для алгоритмов при соответствующих параметрах, которые доказывают принципиальное различие моделей.

Введение

Построение расписаний на m идентичных процессорах, когда требуется распределить работы таким образом, чтобы время завершения последней работы было минимально, является классической задачей теории расписаний [1]. Хорошо известно, что эта задача является NP-трудной. Для ее решения предложено множество приближенных алгоритмов.

В последнее время большой интерес вызывают версии задачи с неполной информацией, так называемые онлайн и semi онлайн. В онлайн-версии нет никакой информации о количестве работ, длительности их выполнения на процессоре. Работы поступают одна за другой, и, прежде чем поступит информация о длительности следующей работы или об ее отсутствии, текущая работа должна быть назначена на конкретный процессор, причем это назначение окончательное, т. е. не может быть изменено в дальнейшем. Для оценки качества работы алгоритмов решения версий задач с неполной информацией обычно используется сравнительный анализ. Сравниваются значения решений, получаемых алгоритмом для онлайн-версии, с оптимальными значениями соответствующей офлайн-версии (когда все данные заранее известны), и устанавливается гарантированная оценка алгоритма [2, 3]. Следует отметить, что при анализе алгоритмов для задач с неполной информацией используются так называемые нижние границы для гарантированных оценок онлайн- и semi онлайн-алгоритмов [4, 5]. Наличие нижней границы означает принципиальную невозможность построения алгоритма с гарантированной оценкой, лучшей, чем нижняя граница.

Semi онлайн-версия задач предполагает наличие дополнительной информации общего характера [6]. Для рассматриваемой задачи известно несколько semi онлайн-версий: работы поступают в отсортированном порядке, известны оптимальное значение решения (задача 1) [7] и общая сумма длительностей выполнения (задача 2) [8].

Общая схема для задач с известным оптимальным значением целевой функции или суммарной длительностью всех работ

Будем считать, что Z – оптимальное значение целевого функционала, а S – общая сумма длительностей выполнения работ для semi онлайн-версий задачи. Тогда нижняя граница LB для оптимального значения функционала в первом случае равна Z, а во втором – средней загрузке процессоров S/m.

Будем считать, что $LB = 1$. Этого можно добиться, пронормировав все величины (поделив на значение LB). Определим параметр $\alpha > 0$ и, используя этот параметр, опишем алгоритм, гарантированная оценка которого будет $(1+\alpha)$. В работе [8] было доказано, что нижняя граница для semi онлайн-версии, когда известна общая сумма процессорных времен, не меньше 1,5, поэтому $\alpha \geq 0{,}5$.

Разобьем поступающие работы на классы в соответствии с их процессорным временем.

В класс A определим работы, у которых процессорное время больше чем $(1+\alpha)/2$. Очевидно, что для обеспечения нужной гарантированной оценки нельзя назначать две работы из класса A на один процессор.

В класс *B* определим работы, у которых процессорное время больше α, но не больше чем (1+α)/2. Очевидно, что для обеспечения нужной гарантированной оценки на один процессор можно назначать две работы из класса *B*. При этом загрузка такого процессора будет больше 1.

В класс *C* определим работы, у которых процессорное время не больше чем α. Добавление таких работ на процессор с загрузкой меньше *LB* приведет к тому, что загрузка процессора может превысить 1, но не превысит (1 + α).

Понятно, что в любом решении невозможна ситуация, когда загрузка каждого процессора превосходит 1. Основная идея алгоритма состоит в специальным образом осуществляемой загрузке процессоров, чтобы для одного процессора или группы процессоров их средняя загрузка была не меньше 1, а максимальная загрузка любого из процессоров не превосходила более чем в (1 + α) раз величину оптимального значения целевой функции.

Ключевым моментом данной схемы является выделение специальных типов процессоров:

– процессор относится к типу *Close_A*, если на него назначена одна работа из класса *A* и несколько работ из класса *C*, причем общая загрузка процессора больше 1, но не превышает величины (1 + α);

– процессор относится к типу *Open_A*, если на него назначена одна работа из класса *A* и несколько работ из класса *C*, причем общая загрузка процессора не больше 1;

– процессор относится к типу *Close_B*, если на него назначены две работы из класса *B*;

– процессор относится к типу *Open_B*, если на него назначена одна работа из класса *B*.

Для работ из класса *C* определим группу из четырех процессоров (пусть это процессоры *CC*1, *CC*2, *CC*3, *CC*4), которую будем называть *Close_bunch_C*. Такую группу можно получить следующим образом. Пусть поступают работы из класса *C*. При поступлении очередной работы из класса *C* назначаем ее на процессор *CC*1 группы, если получаемая суммарная загрузка не превышает α. Если же суммарная загрузка процессора превысит α, то пытаемся назначить работу на процессор *CC2*, чтобы суммарная загрузка не превысила α. Если это возможно, то назначаем. В случае когда невозможно назначить работу ни на процессор *CC*1, ни на *CC*2 без превышения α, назначаем ее на процессор *CC*3, добавляем к группе пустой процессор *CC*4 и считаем, что группа сформирована.

Очевидно, что в сформированной группе суммарная загрузка на первых двух процессорах больше α. Это справедливо также для первого и третьего, а также для второго и третьего процессоров.

Свойство. Загрузка каждого из процессоров *CC*1, *CC*2, *CC*3 группы не превосходит α, процессор *CC*4 пуст, а суммарная загрузка процессоров *CC*1, *CC*2, *CC*3 больше 3α/2.

Может оказаться, что будет сформирована неполная группа четвертого типа, когда в ней менее четырех процессоров. Такую группу будем называть *Open_bunch_C*. Группа этого типа может содержать от одного до трех процессоров, причем три процессора в ней могут быть только в том случае, если не осталось ни одного пустого процессора для формирования группы *Close_bunch_C*.

Алгоритм построения расписания состоит из двух этапов. На первом этапе формируются процессоры описанных выше типов следующим образом.

Если поступает работа из класса *A*, то приоритетным процессором для нее является наиболее загруженный процессор из группы *Open_bunch_C*. В этом случае получается процессор либо типа *Close_A*, либо *Open_A*. Если группы *Open_bunch_C* нет, но есть группа *Close_bunch_C*, то приоритетным для нее является наиболее загруженный процессор из данной группы. В этом случае формируется процессор типа *Close_A* и группа *Open_bunch_C*, которая получается путем удаления из нее пустого процессора. Иначе работа назначается на пустой процессор. Следует отметить, что процессорное время работы может быть больше 1. Назначение такой работы на любой из перечисленных выше процессоров (в том числе и на пустой) приводит к получению процессора *Close_A*, причем его суммарная загрузка не превосходит (1+α)*LB*.

Если поступает работа из класса *B*, то приоритетным для нее является процессор *Open_B*. Иначе работа назначается на пустой процессор. В этом случае получается процессор либо типа *Close_B*, либо *Open_B*.

Если поступает работа из класса *C*, то приоритетным для нее является процессор *Open_A*. В этом случае получается процессор либо типа *Close_A*, либо *Open_A*. Если такого процессора

нет, то работа назначается на процессор из незавершенной группы *Open_bunch_C* с целью получения группы *Close_bunch_C*. Иначе работа назначается на пустой процессор и открывается новая группа *Open_bunch_C*.

Первый этап алгоритма заканчивается, когда для очередной работы нет пустого процессора (не считая пустых процессоров в группах типа *Close_bunch_C*, которые на первом этапе не заполняются).

Очевидно, что после первого этапа могут быть одновременно получены процессоры *Close_A*, *Open_A*, *Close_B*, *Open_B* и группы процессоров *Close_bunch_C* и *Open_bunch_C*. При этом количество процессоров *Open_B* не превышает 1, а если есть группы процессоров *Close_bunch_C* или *Open_bunch_C*, то количество процессоров *Open_A* не превышает 1.

В дальнейшем процессоры *Close_A* и *Close_B* редуцируются, так как их загрузка больше 1. Отметим, что при назначении поступившей работы всегда имеется процессор, загрузка которого меньше 1.

Рассмотрим два возможных случая структуры распределения процессоров по типам после завершения первого этапа алгоритма.

Случай 1. Имеется более одного процессора типа *Open_A*. Это означает, что групп процессоров типа *Close_bunch_C* и *Open_bunch_C* нет. Действительно, пусть имеются два процессора типа *Open_A*, *AA*1 и *AA*2. Группа процессоров типа *Close_bunch_C* или *Open_bunch_C* не может быть построена после получения *AA*1 и *AA*2, так как работы из класса *C* были бы назначены на эти процессоры. С другой стороны, только один процессор из группы *Open_bunch_C* может иметь загрузку меньше $\alpha/2$.

Таким образом, процессоры могут быть только типа *Open_A* и не более одного процессора *Open_B*, остальные процессоры будут *Close_A* и *Close_B*.

Если в дальнейшем поступают работы из класса *C*, то они назначаются на наиболее загруженный процессор типа *Open_A* для получения процессора *Close_A*.

Если поступила работа длительности *X* из класса *B*, то пытаемся назначить ее на процессор типа *Open_B*, если он есть. Предположим, что процессора типа *Open_B* нет. Это означает, что можно пересчитать нижнюю оценку оптимального решения.

Действительно, имеется, по крайней мере, $m+1$ работа из классов *A* или *B*, причем одна работа из класса *B* должна быть назначена на процессор вместе с работой из класса *A* либо на один процессор должны быть назначены две работы из класса *A* или три работы из класса *B*. Следовательно, $LB \geq \min\{3\alpha, \alpha+(1+\alpha)/2, 1+\alpha\} = \alpha+(1+\alpha)/2$ при $1/2 \leq \alpha < 1$.

Заметим, что при назначении работы длительности *X* на наименее загруженный процессор его загрузка не превысит $1+(1+\alpha)/2$. Отсюда

$$(1+(1+\alpha)/2)/LB \leq (1+(1+\alpha)/2)/(\alpha+(1+\alpha)/2) = 1+(1-\alpha)/(\alpha+(1+\alpha)/2) \leq 1{,}5.$$

Если поступила работа длительности *X* из класса *A*, то можно пересчитать *LB*, при этом ее процессорное время может быть больше 1. Для этого определим новое значение $LB = \max\{\alpha+(1+\alpha)/2, X\}$.

Тогда при назначении работы длительности *X* на наименее загруженный процессор его загрузка не превысит $1+X$. Из этого следует

$$(1+X)/LB \leq 1+1/\max\{\alpha+(1+\alpha)/2, X\}.$$

Поэтому $(1+X)/LB \leq 1+\alpha$ для $\alpha \geq 2/3$.

Лемма. При отсутствии групп процессоров типа Close_bunch_C и Open_bunch_C при $\alpha \geq 2/3$ для любой последовательности работ назначение работы из классов *A* и *B* на наименее загруженный процессор обеспечивает загрузку, которая не превышает $(1 + \alpha)LB$.

Замечание. Для semi онлайн-версии, когда известно оптимальное значение целевой функции, в случае 1 после первого этапа будут поступать только работы из класса *C*, так как общее количество работ из классов *A* и *B* не превышает *m* (длительность каждой такой работы больше чем $Z/2$).

Случай 2. Имеется не более одного процессора типа *Open_A* и имеются группы процессоров типа *Close_bunch_C* и/или *Open_bunch_C* .

На втором этапе группы *Close_bunch_C* больше не меняются, а могут только редуцироваться. Группа *Close_bunch_C* будет считаться редуцированной, если сумма длительностей работ на ее процессорах не меньше четырех. Редуцированная группа из дальнейшего рассмотрения исключается, на процессоры такой группы новые работы не назначаются.

Идея второго этапа алгоритма основывается на использовании резервного четвертого процессора группы *Close_bunch_C*, что позволяет гарантировать суммарную загрузку в каждой группе процессоров типа *Close_bunch_C* не меньше четырех.

Отметим, что при поступлении работ из класса *A* они сначала назначаются на процессоры из группы *Open_bunch_C* .

Предположим, что имеются, по крайней мере, две нередуцированные группы процессоров типа *Close_bunch_C*. Группу *Close_bunch_CA* будем использовать для работ из класса *A*, а группу *Close_bunch_CB* – для работ из классов *B* и *C*.

Пусть в группу *Close_bunch_CA* были добавлены четыре работы из класса *A*. Тогда для редуцирования этой группы достаточно, чтобы суммарная загрузка в группе была не меньше четырех. Это значит, что с учетом свойства 1 должно выполняться соотношение $3\alpha/2+4(1+\alpha)/2\geq 4$. Отсюда следует, что редуцирование группы возможно при $\alpha\geq 4/7$.

Рассмотрим редуцирование группы *Close_bunch_CB*.

Пусть $L(CC)$ соответствует загрузке процессора *CC*. Не умаляя общности, будем считать, что загрузки процессоров *CC*1, *CC*2, *CC*3 упорядочены в порядке $L(CC1)\leq L(CC2)\leq L(CC3)$.

Вначале покажем, что при добавлении в группу более пяти работ класса *B* она может быть редуцирована. Если добавлено не менее шести работ класса *B*, то суммарная загрузка будет, как минимум, $3\alpha/2+6\alpha\geq 4$ при $\alpha\geq 1/2$. Поэтому будем рассматривать случай, когда поступило последовательно до пяти работ класса *B*.

Первую поступившую работу длительности $X0$ назначаем на процессор *CC*1. Если его загрузка не превысила α, то снова имеем группу *Close_bunch_C*. При этом опять будем считать, что загрузки процессоров *CC*1, *CC*2, *CC*3 упорядочены в порядке $L(CC1)\leq L(CC2)\leq L(CC3)$.

Пусть загрузка процессора *CC*1 стала больше α. При поступлении второй работы поступаем следующим образом.

Вариант 1. Если поступила работа длительности $X1$ из класса *B*, то пытаемся назначить ее на процессор CC1. Если суммарная загрузка не превысила $1+\alpha$, то делаем назначение. При этом загрузка процессора CC1 стала больше $\alpha/2+(1+\alpha)/2>1$ при $\alpha\geq 1/2$. При поступлении работ из классов *B* и *C* на пустой процессор может быть назначено не менее двух работ. Поэтому если эти работы не могли быть назначены на процессоры *CC*2 и *CC*3, то суммарная загрузка процессоров *CC*2, *CC*3 и *CC*4 будет больше $2(1+\alpha)$. Суммарная загрузка в группе будет больше четырех при $\alpha\geq 1/2$.

Вариант 2. Пусть поступила работа длительности $X1$ из класса *B*, но $L(CC1)+X1>1+\alpha$. В этом случае назначаем работу длительности $X1$ на процессор *CC*3. При этом очевидно, что работа длительности $X0$ была из класса *B*. Учитывая свойство 1, имеем $L(CC1)+L(CC3)+X1>1+\alpha+\alpha/2$. Поэтому суммарная загрузка одного из процессоров будут больше чем $(1+\alpha+\alpha/2)/2$. Не умаляя общности, пусть это процессор *CC*1. Тогда опять для трех процессоров *CC*2, *CC*3 и *CC*4 можно обеспечить суммарную загрузку больше $2(1+\alpha)$. Поэтому суммарная загрузка в группе будет больше $2(1+\alpha)+(1+\alpha+\alpha/2)/2$. Для редуцирования требуется выполнение неравенства $2(1+\alpha)+ (1+\alpha+\alpha/2)/2\geq 4$, которое справедливо при $\alpha\geq 6/11$. Отсюда следует, что редуцирование группы возможно при $\alpha\geq 6/11$.

Вариант 3. Поступила работа длительности $X1$ из класса *C*. Назначаем ее на прибор *CC*2. Будем считать, что $L(CC2)>\alpha$, иначе опять процессоры *CC*2 и *CC*3 можно пересортировать. После этого все поступающие работы из класса *C* будем назначать на процессор *CC*1, пока его загрузка не превысит 1. После этого имеем ситуацию, описанную выше. Пусть пришла работа длительности $X2$ из класса *B*. Пробуем назначить ее на процессор *CC*2. Если это возможно, то назначаем ее, получаем процессор с загрузкой больше 1. Поэтому получили вариант, описанный выше. Предположим, что $L(CC2)+X2>1+\alpha$. Учитывая $L(CC3)\geq L(CC2)$ до назначения работы длительности $X1$ на процессор *CC*2 и $X1\leq\alpha$, получаем $1< L(CC3)+X2\leq 1+\alpha$. Поэтому существует процессор с загрузкой больше 1. Аналогичный вариант описан выше.

Рассмотрим случай, когда осталась одна нередуцированная группа. При этом всегда можно получить процессор, загрузка которого больше 1. Действительно, при поступлении работы из класса *A* используется процессор *CC*3, для работ из классов *B* и *C* используются процессоры *CC*1 и *CC*4. Как только получен процессор с загрузкой, большей 1, остается не более трех процессоров. Если среди них есть два процессора с суммарной загрузкой больше 3/2, то оставшаяся максимальная возможная загрузка четвертого процессора не превышает 3/2, поэтому возможные оставшиеся работы могут быть гарантированно назначены на него. Если такой пары нет, то загрузка каждого из двух не превышает α. Если невозможно назначить работу на один из процессоров, то она может быть назначена на другой.

Таким образом, при любой последовательности поступления работ происходит редуцирование процессоров с суммарной загрузкой больше 1 или редуцирование группы со средней загрузкой больше 1, что гарантирует корректную работу алгоритма. При этом редуцирование групп процессоров возможно при $\alpha=\max\{4/7,6/11\}=4/7$.

Из этих рассуждений вытекают следующие утверждения:

Теорема 1. *При наличии непустой группы процессоров типа Close_bunch_C для любой последовательности длительностей работ при $\alpha=4/7$ существует алгоритм, обеспечивающий редуцирование.*

Теорема 2. *Предложенная двухэтапная схема корректно решает задачу с известной общей суммой длительностей выполнения работ при $\alpha\geq 2/3$, а задачу с известным оптимальным значением целевой функции при $\alpha\geq 4/7$.*

Заключение

В работе [9] для задачи с известной общей суммой длительностей выполнения недавно доказана нижняя граница, которая лежит в пределах интервала [1,58504; 1,58505]. Предложенный в настоящей работе алгоритм для задачи с известным оптимальным значением целевой функции имеет гарантированную оценку, которая лучше нижней границы для задачи с известной суммой длительностей выполнения. Это указывает на принципиальное различие задач.

В приведенной схеме не учитывается специфика задач. Например, для задачи с известным оптимальным значением целевой функции суммарное количество работ из классов *A* и *B* не превосходит *m*.

Дальнейшим возможным направлением исследований представляется разработка модифицированных алгоритмов, которые будут учитывать специфику каждой из задач, что, возможно, позволит существенно улучшить гарантированные оценки приведенной общей схемы.

Работа выполнена при частичной финансовой поддержке БРФФИ (проект № Ф10ФП-001).

Список литературы

1. Graham, R.L. Bounds for certain multi-processing anomalies / R.L. Graham // Bell System Technical J. – 1966. – Vol. 45. – P. 1563–1581.

2. Galambos, G. An on-line scheduling heuristic with better worst case ratio than Graham's list scheduling / G. Galambos, G. Woeginger // SIAM J. on Computing. – 1993. – Vol. 22. – P. 349–355.

3. Fleischer, R. Online scheduling revisited / R. Fleischer, M. Wahl // J. of Scheduling. – 2000. – Vol. 3. – P. 343–353.

4. Faigle, U. On the performance of on-line algorithms for partition problems / U. Faigle, W. Kern, G. Turan // Acta Cybernetica. – 1989. – Vol. 9. – P. 107–119.

5. Rudin III, J.F. Improved bounds for the online scheduling problem / J.F. Rudin III, R. Chandrasekaran // SIAM J. on Computing. – 2003. – Vol. 32. – P. 717–735.

6. Semi on-line algorithms for the partition problem / H. Kellerer [et al.] // Operations Research Letters. – 1997. – Vol. 21. – P. 235–242.

7. Azar, Y. On-line bin-stretching / Y. Azar, O. Regev // Theoretical Computer Sci. – 2001. – Vol. 268. – P. 17–41.

8. Cheng, T.C.E. Semi-on-line multiprocessor scheduling with given total processing time / T.C.E. Cheng, H. Kellerer, V. Kotov // Theoretical Computer Sci. – 2005. – Vol. 337. – P. 134–146.

9. Albers, S. Semi-Online Scheduling Revisited / S. Albers, M. Hellwig // Theoretical Computer Sci. – 2012. – Vol. 443. – P. 1–9.

Поступила 22.06.2012

[1]*Белорусский государственный университет,*
Минск, пр. Независимости, 4
e-mail: kotovvm@bsu.by

[2]*Университет Граца, Университетштрассе 15,*
А-8010, Австрия
e-mail: hans.kellerer@uni-graz.at

[3]*Университет Гренобля,*
Феликс Виоле пр., Гренобль, Франция
e-mail: nadia.brauner@grenoble-inp.fr
gerd.finke@g-scop.inpg.fr

V.M. Kotov, H. Kellerer, N. Brauner, G. Finke

AN ALGORITHM FOR SEMI ONLINE VERSIONS OF THE PROBLEM $Pm||C_{\max}$

We propose a parametric scheme for two versions of semi online multiprocessor scheduling problem to minimize makespan, with a priori known total processing time and a priori known optimal value, respectively, which provides the best known worst-case approximation for these versions.

УДК 658.512.2

Г.М. Левин, Б.М. Розин

ОПТИМИЗАЦИЯ ДЛИТЕЛЬНОСТЕЙ ПОСЛЕДОВАТЕЛЬНО-ПАРАЛЛЕЛЬНОГО ВЫПОЛНЕНИЯ ПЕРЕСЕКАЮЩИХСЯ МНОЖЕСТВ ОПЕРАЦИЙ

Предлагаются математическая модель и декомпозиционный метод оптимизации длительностей последовательно-параллельного выполнения пересекающихся множеств операций. Метод основывается на сочетании идей параметрической декомпозиции и динамического программирования.

Введение

Значительное внимание в последние десятилетия в литературе уделялось различным аспектам планирования выполнения комплексов операций в организационных и производственных системах [1–5]. В данной работе рассматривается задача оптимизации длительностей последовательно-параллельного выполнения пересекающихся множеств операций.

Заданы множество J элементарных операций (в дальнейшем э-операций) и комплекс **I** составных операций (в дальнейшем с-операций). С-операция i комплекса включает все э-операции соответствующего подмножества $J_i \subseteq J$, причем в состав с-операции i может входить m_{ij} идентичных э-операций $j \in J$. Комплекс содержит n различных с-операций (образованных различными подмножествами множества J) и может включать n_i идентичных с-операций $i \in I = \{1, \ldots, n\}$, причем подмножества семейства $\{J_1, \ldots, J_i, \ldots, J_n\}$, образующие с-операции комплекса, могут пересекаться (рис. 1).

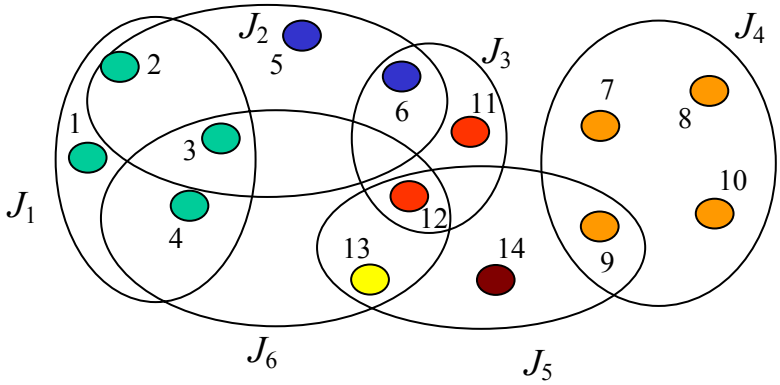


Рис. 1. Пример пересечения с-операций комплекса

Процесс выполнения комплекса **I** заключается в последовательном выполнении (однократном либо циклически повторяющемся) всех его с-операций. Все э-операции, входящие в состав очередной с-операции, выполняются одновременно, причем длительность с-операции равна наибольшей из длительностей входящих в нее э-операций. Для каждой э-операции $j \in J$ заданы диапазон $[t_{1j}, t_{2j}]$ возможных длительностей ее выполнения и определенная на этом отрезке убывающая функция $f_j(t)$ зависимости затрат на ее выполнение от принятой длительности t ее выполнения. Рассматривается случай, когда принятая длительность э-операции должна быть одинакова для всех с-операций, в которые эта э-операция входит. Стоимость выполнения каждой из с-операций в целом помимо суммарной стоимости выполнения составляющих ее э-операций включает также дополнительные затраты, пропорциональные длительности ее выполнения.

В данной работе ограничимся случаем, когда коэффициент пропорциональности $E > 0$ одинаков для всех с-операций комплекса и последовательность выполнения с-операций не влияет на длительность выполнения составляющих их э-операций.

Требуется найти значения длительностей $t_j \in [t_{1j}, t_{2j}]$ выполнения всех э-операций $j \in J$, минимизирующие суммарные затраты на выполнение всех с-операций комплекса.

Подобные задачи возникают, в частности, при проектировании групповых процессов многоинструментальной обработки деталей на многопозиционных производственных линиях конвейерного типа (рис. 2). В качестве примера рассмотрим процесс обработки на многопозиционном многоинструментальном оборудовании последовательности деталей, составленной из следующих друг за другом идентичных подпоследовательностей (групп), каждая из которых включает h деталей m различных наименований, $h \geq m$. В группе может быть несколько деталей одного наименования. Предполагается, что рабочие позиции линейно упорядочены и каждая деталь последовательно в этом порядке обрабатывается на каждой рабочей позиции соответствующим этой позиции и детали набором инструментов, причем в каждый момент времени на каждой позиции может обрабатываться лишь одна деталь. Один такт обработки состоит в одновременной обработке на каждой из рабочих позиций соответствующей (такту и позиции) детали, при этом все инструменты каждой позиции, выполняющие обработку соответствующей детали, также работают одновременно. После завершения любого такта обработки каждая обрабатываемая деталь со своей позиции синхронно перемещается на следующую позицию, деталь с последней позиции снимается, а на первую позицию устанавливается очередная деталь последовательности. Таким образом, цикл обработки группы деталей состоит из h тактов.

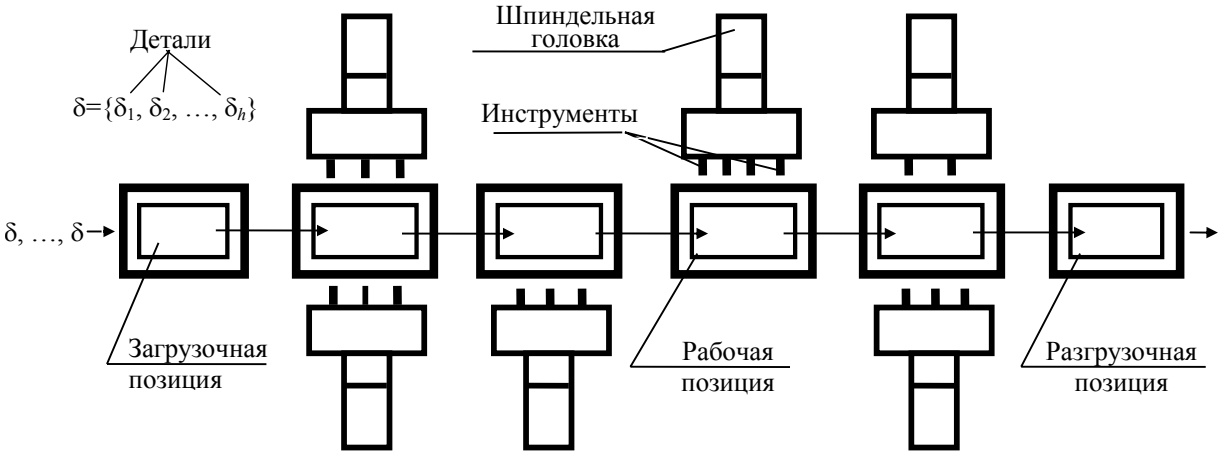


Рис. 2. Общий вид конвейерной линии для групповой обработки деталей

Инструменты каждой позиции сгруппированы в один или несколько блоков, каждый из которых расположен в отдельной шпиндельной головке со своим приводом подачи. Таким образом, все инструменты одного блока обрабатывают конкретную деталь на одной и той же минутной подаче. Затраты на обработку группы деталей складываются из затрат, пропорциональных суммарной продолжительности тактов, и доли затрат на смену инструментов, приходящейся на одну группу зависящих от режимов (длительностей) выполнения этими инструментами операций.

Множество элементарных операций, выполняемых инструментами для одного такта обработки, может пересекаться с множеством операций для другого такта вследствие наличия в группе повторяющихся деталей либо если детали различных наименований содержат одинаковые конструктивные элементы, обрабатываемые одним инструментом.

Требуется определить такие режимы (длительности) обработки, которые удовлетворяют заданным ограничениям и минимизируют затраты на обработку группы деталей.

1. Математическая постановка задачи и ее параметризация

Задача А. Для заданного комплекса **I** с-операций требуется найти такой вектор $t = (t_j | j \in J) \in \mathbf{t} = \prod_{j \in J} [t_{1j}, t_{2j}]$, которому соответствует наименьшее значение функции общих затрат

$$F(t) = \sum_{i \in I} n_i (E \max_{j \in J_i} t_j + \sum_{j \in J_i} m_{ij} f_j(t_j)) = E \sum_{i \in I} n_i \max_{j \in J_i} t_j + \sum_{j \in J} p_j f_j(t_j), \quad (1)$$

где $p_j = \sum_{i \in I_j} n_i m_{ij}$ и $I_j = \{I \in I | j \in J_i\}$.

Замечание. В рамках данной задачи без ограничения общности можно считать, что комплекс **I** с-операций не может быть разбит на несколько подмножеств таким образом, чтобы элементы из разных подмножеств не пересекались. В противном случае исходная задача разбивается на соответствующее число аналогичных независимых задач относительно каждого из таких подмножеств.

Введем вектор $T = (T_1, \ldots, T_i, \ldots, T_n)$, компонента T_i которого определяет длительность выполнения с-операции $i \in I$. Таким образом, $T_i \in \mathbf{T}_i = [\max_{j \in J_i} t_{1j}, \max_{j \in J_i} t_{2j}]$. Тогда наряду с задачей А можно рассматривать следующую параметризованную задачу В:

$$\Phi(T, t) = E \sum_{i \in I} n_i T_i + \sum_{j \in J} p_j f_j(t_j) \to \min_{T,t}; \quad (2)$$

$$t_j \in [t_{1j}, t_{2j}], \quad j \in J; \quad (3)$$

$$t_j \le T_i, \quad j \in J, \quad i \in I_j. \quad (4)$$

Задачи А и В эквивалентны в том смысле, что если t' – решение задачи А, то $(T(t'), t')$ – решение задачи В, где $T(t') = (T_1(t'), \ldots, T_n(t'))$ и $T_i(t') = \max_{j \in J_i} t'_j$. В свою очередь, если (T^*, t^*) – решение задачи В, то t^* – решение задачи А.

2. Решение задачи В

Как показано ниже, решение задачи В может быть сведено к отысканию кратчайшего пути из начальной вершины некоторого бесконтурного орграфа в одну из его конечных вершин. Каждой вершине орграфа сопоставлены некоторый набор с-операций комплекса, образующее их подмножество э-операций и принимаемая максимальная длительность этих операций, а дуге – затраты на выполнение подмножества операций (с-операций и э-операций), дополняющих множество операций начальной вершины дуги до множества операций концевой вершины дуги. При этом длительность э-операций дополняющего подмножества является максимально возможной, но не превышает длительности, сопоставленной концевой вершине дуги, а длительность на концевой вершине дуги не меньше длительности на ее начальной вершине. Начальной вершине орграфа сопоставлены пустые множества операций и нулевая длительность, а каждой из концевых вершин орграфа – множества операций комплекса в целом.

Для реализации такого подхода введем в рассмотрение ряд вспомогательных функций и задач, полагая $\Omega \subset J$:

– $t_j(\tau) = \min\{t_{2j}, \tau\}, \ \tau \ge t_{1j}$;

– $t_j(T) = \min\{t_{2j}, \min_{i \in I_j} T_i\}$;

– $\Psi(T) = E \sum_{i \in I} n_i T_i + \sum_{j \in J} p_j f_j(t_j(T)), \ T \in \prod_{i \in I} \mathbf{T}_i$;

– задачу C минимизации функции $\Psi(T)$ на множестве $\prod_{i\in I}\mathbf{T}_i$;

– $\mathbf{I}(\Omega) = \{J_i\backslash\Omega \mid J_i\backslash\Omega \neq \varnothing,\ i \in I\}$;

– семейство $\mathbf{\Gamma}^+(\Omega)$ минимальных по включению элементов из $\mathbf{I}(\Omega)$, т.е. $\Omega' \not\subset \Omega''$ для любой пары (Ω', Ω'') множеств из $\mathbf{\Gamma}^+(\Omega)$;

– $s_2(\Omega) = \sum_{i\in\bigcup_{j\in\Omega} I_j} n_i$;

– $I(\Omega_1,\Omega_2)=\{i\in I \mid \Omega_2 = J_i\backslash\Omega_1\}$ и $s_1(\Omega_1,\Omega_2)= \sum_{i\in I(\Omega_1,\Omega_2)} n_i$, где $\Omega_k \subseteq J$, $k = 1, 2$, $\Omega_1 \cap \Omega_2 = \varnothing$;

– $\Delta_1(\Omega) = \max_{j\in\Omega} t_{1j}$, $\Delta_2(\Omega) = \max_{j\in\Omega} t_{2j}$ и $\Delta(\Omega) = [\Delta_1(\Omega), \Delta_2(\Omega)]$;

– $\varphi(\Omega, r, \tau) = Er\tau + \sum_{j\in\Omega} p_j f_j(t_j(\tau))$, где $\tau \geq \Delta_1(\Omega)$, r – положительное целое;

– значение $\tau(\Omega, r, \tau')$ параметра $\tau \in [\max\{\tau', \Delta_1(\Omega)\}, \max\{\tau', \Delta_2(\Omega)\}]$, которому соответствует наименьшее значение функции $\varphi(\Omega, r, \tau)$, где $\tau' > 0$;

– задачу $C(\Omega)$, аналогичную задаче C, порождаемую семейством $\mathbf{I}(\Omega)$ подмножеств множества $J \setminus \Omega$ э-операций при тех же значениях n_i и m_{ij} для $J_i \setminus \Omega \neq \varnothing$ и $j \in J_i \setminus \Omega$.

Отметим некоторые свойства функции $\varphi(\Omega, r, \tau)$:

а) если функции $f_j(t_j)$ выпуклы на $[t_{1j}, t_{2j}]$ для всех $j \in J$, то функция $\varphi(\Omega, r, \tau)$ является выпуклой на множестве $\tau \geq \Delta_1(\Omega)$ для любых $\Omega \subseteq J$ и r;

б) если $0 < r_1 < r_2$, то $\varphi(\Omega, r_1, \tau) < \varphi(\Omega, r_2, \tau)$, что очевидно, и $\tau(\Omega, r_1, \tau') \geq \tau(\Omega, r_2, \tau')$ для всех $\tau \geq \geq \Delta_1(\Omega)$, $\tau' \geq 0$.

Без ограничения общности можно считать, что $T^*_1 = \min\{T^*_i \mid i \in I\}$ и не существует такого $i \in I$, что $J_1 \subset J_i$. Введенные определения позволяют выявить следующие свойства решения задачи B:

1. Очевидно, что $T^*_i = \max_{j\in J_i} t^*_j$ и $t^*_j = \min\{t_{2j}, \min_{i\in I_j} T^*_i\}$, поэтому $T^*_i = \max_{j\in J_i} t^*_j$ и $T^*_{i'} \leq T^*_{i''}$ для любых $i', i'' \in I$, таких, что $J_{i'} \subset J_{i''}$.

2. Вектор T^* является решением задачи C.

3. $t^*_j = \min\{t_{2j}, T^*_1\}$ для всех $j \in J_1$.

4. Значение $T^*_1 \in [\Delta_1(J_1),\ \tau(J_1, s_2(J_1), \Delta_1(J_1))]$.

5. Вектор $(T^*_2, \ldots, T^*_n)$ является решением задачи $C(J_1)$.

Введем в рассмотрение множество $\mathbf{\Pi}$ последовательностей $\pi = ((\Omega_1(\pi), \tau_1(\pi)), \ldots, (\Omega_v(\pi), \tau_v(\pi)), \ldots, (\Omega_{h_\pi}(\pi), \tau_{h_\pi}(\pi)))$, соответствующих путям бесконтурного орграфа из начальной вершины в одну из концевых, где $\Omega_v(\pi)$ – некоторое подмножество э-операций и $\tau_v(\pi)$ – их максимальная длительность, удовлетворяющие следующим условиям:

– $\Omega_v(\pi) \in \mathbf{\Gamma}^+(Z_{v-1}(\pi))$, где $Z_v(\pi) = Z_{v-1}(\pi) \cup \Omega_v(\pi)$ и $Z_0(\pi) = \varnothing$;

– $\tau_v(\pi) \in [\max\{\Delta_1(\Omega_v(\pi)), \tau_{v-1}(\pi)\}, \tau(\Omega_v(\pi), s_1(Z_{v-1}(\pi), \Omega_v(\pi)), \tau_{v-1}(\pi))]$, где $\tau_0(\pi) = 0$;

– $Z_{h_\pi}(\pi) = J$.

Очевидно, что $h_\pi \leq n$ и $\bigcup_{v=1}^{h_\pi} I(Z_{v-1}(\pi), \Omega_v(\pi)) = I$ для всех $\pi \in \mathbf{\Pi}$.

Пусть $Z \subseteq J$ и $\Theta(Z)$ – множество таких с-операций $i \in I$, что $J_i \subseteq Z$. Тогда $\mathbf{\Pi}$ – это множество таких последовательностей пар $(\Omega_v(\pi), \tau_v(\pi))$, что $\Omega_v(\pi)$ – минимальное (по включению) подмножество э-операций из $J \setminus Z_{v-1}(\pi)$, для которого выполняется $\Theta(Z_{v-1}(\pi)) \subset \Theta(Z_{v-1}(\pi) \cup \Omega_v(\pi))$, и $\tau_v(\pi) \geq \tau_{v-1}(\pi)$.

Для представленного на рис. 1 комплекса операций одна из возможных последовательностей $\Omega_1(\pi), \ldots, \Omega_v(\pi), \ldots, J$ подмножеств э-операций для построения $\pi \in \mathbf{\Pi}$ может быть следующей: $\Omega_1(\pi) = \{1, 2, 3, 4\}$, $\Omega_2(\pi) = \{5,6\}$, $\Omega_3(\pi) = \{7, 8, 9, 10\}$, $\Omega_4(\pi) = \{11, 12\}$, $\Omega_5(\pi) = \{13\}$, $\Omega_6(\pi) = \{14\}$.

При этом $Z_1(\pi) = J_1$, $Z_2(\pi) = J_1 \cup J_2$, $Z_3(\pi) = J_1 \cup J_2 \cup J_4$, $Z_4(\pi) = J_1 \cup J_2 \cup J_4 \cup J_3$, $Z_5(\pi)=J_1 \cup J_2 \cup J_4 \cup J_3 \cup J_6$, $Z_6(\pi) = J_1 \cup J_2 \cup J_4 \cup J_3 \cup J_6 \cup J_5 = J$. Отметим, в частности, что для $Z_2(\pi)$ множество $\mathbf{I}(Z_2(\pi)) = \{\{11, 12\}, \{12, 13\}, \{7, 8, 9, 10\}, \{9, 12, 13, 14\}\}$ и $\mathbf{I}^+(Z_2(\pi))=\{\{11, 12\}, \{12, 13\}, \{7, 8, 9, 10\}\}$, а для $Z_4(\pi)$ множество $\mathbf{I}(Z_4(\pi))=\{\{13\}, \{13, 14\}\}$ и $\mathbf{I}^+(Z_4(\pi))=\{13\}$.

Положим $G(\pi) = \sum_{v=1}^{h_\pi} \varphi(\Omega_v(\pi), s_1(Z_{v-1}(\pi), \Omega_v(\pi)), \tau_v(\pi))$ для $\pi \in \mathbf{\Pi}$. Тогда

6. Существует такое $\mathbf{\Pi}^* \subseteq \mathbf{\Pi}$, что $G(\pi) = \Psi(T^*)$ для всех $\pi \in \mathbf{\Pi}^*$.

7. Для всех $\pi \in \mathbf{\Pi}^*$ значение $\tau_v(\pi) = T^*_i$ для всех $i \in I(Z_{v-1}(\pi),\Omega_v(\pi))$ и существует такое $\pi \in \mathbf{\Pi}^*$, что $\Omega_1(\pi) = J_1$.

Таким образом, решение задачи B может быть сведено к решению задачи C, а последнее – к построению последовательности $\pi \in \mathbf{\Pi}^*$.

3. Построение последовательности $\pi \in \mathbf{\Pi}^*$

Для описания возможного подхода к построению некоторой $\pi \in \mathbf{\Pi}^*$ введем следующие обозначения:

$\mathbf{Z}_0 = \{Z_v(\pi) | \pi \in \mathbf{\Pi}, v = 0, \ldots, h_\pi\}$ и $\mathfrak{R} = \{\mathfrak{R}_v(\pi) = (Z_v(\pi), \tau_v(\pi)) | \pi \in \mathbf{\Pi}, v = 0, \ldots, h_\pi\}$;

$\mathbf{I}^-(\Omega)$ – семейство таких подмножеств Ω' множества Ω, что $\Omega \setminus \Omega' \in \mathbf{Z}_0$ и $\Omega' \in \mathbf{I}^+(\Omega \setminus \Omega')$;

$$H(Z, \tau) = \min\{\sum_{p=1}^{v} \varphi(\Omega_p(\pi), s_1(Z_{p-1}(\pi), \Omega_p(\pi)), \tau_p(\pi)) | \pi \in \mathbf{\Pi}, v \in \{1, \ldots, h_\pi\}, \mathfrak{R}_v(\pi) = (Z, \tau)\},$$

$(Z, \tau) \in \mathfrak{R}$.

Можно показать справедливость рекуррентных соотношений

$$\min\{G(\pi) | \pi \in \mathbf{\Pi}\} = \min\{H(J, \tau) | (J, \tau) \in \mathfrak{R}\}; \quad (5)$$

$$H(Z, \tau) = \min\{H(Z \setminus \Omega, \tau') + \varphi(\Omega, s_1(Z \setminus \Omega, \Omega), \tau) | \Omega \in \mathbf{I}^-(Z), \tau' \le \tau, (Z \setminus \Omega, \tau') \in \mathfrak{R}\}, (Z, \tau) \in \mathfrak{R}, \quad (6)$$

где $H(\varnothing, 0) = 0$.

Соотношения (5), (6) позволяют использовать для построения $\pi \in \mathbf{\Pi}^*$ традиционные вычислительные схемы динамического программирования, сводящие эту задачу к нахождению кратчайшего пути в орграфе $(\mathfrak{R}, \mathbf{D})$ из вершины $(\varnothing,0) \in \mathfrak{R}$ в одну из вершин вида $(J, \tau) \in \mathfrak{R}$. В этом графе пара $((Z_1, \tau_1), (Z_2, \tau_2)) \in \mathfrak{R} \times \mathfrak{R}$ принадлежит множеству дуг $\mathbf{D}$ тогда и только тогда, когда $Z_1 \subset Z_2$, $Z_2 \setminus Z_1 \in \mathbf{I}^+(Z_1)$ и $\tau_1 \le \tau_2$; длина дуги $((Z_1, \tau_1),(Z_2, \tau_2)) \in \mathbf{D}$ равна $\varphi(Z_2 \setminus Z_1, s_1(Z_1, Z_2 \setminus Z_1), \tau_2)$.

Заключение

В статье разработаны математическая модель и метод решения задачи оптимизации длительностей последовательно-параллельного выполнения пересекающихся множеств операций. Метод основан на сочетании идей параметрической декомпозиции и динамического программирования. Полученные результаты могут быть использованы, в частности, при проектировании и управлении функционированием многопозиционных производственных систем различного типа.

Работа была выполнена при поддержке Белорусского республиканского фонда фундаментальных исследований (проект Ф12ФП-001).

Список литературы

1. Boysen, N. A classification of assembly line balancing problems / N. Boysen, M. Fliedner, A. Scholl // European Journal of Operational Research. – 2007. – Vol. 183. – P. 674–693.

2. Bukchin, J. Design of flexible assembly line to minimize equipment cost / J. Bukchin, M.Tzur // IIE Transactions. – 2000. – Vol. 32. – P. 585–598.

3. Gupta, A.K. Optimization of due-date objectives in scheduling semiconductor batch manufacturing / A.K. Gupta, A.I. Sivakumar // International Journal of Machine Tools and Manufacture. – 2006. – Vol. 46. – P. 1671–1679.

4. Задачи распределения ресурсов в управлении проектами / П.С. Баркалов [и др.] – М. : ИПУ РАН, 2002. – 65 с.

5. Burkov,V.N. Models and methods of multiprojects' management / V.N. Burkov, D.A. Novikov // Systems Science. – 1999. – Vol. 256, № 2. – P. 5–14.

Поступила 19.06.2012

Объединенный институт проблем
информатики НАН Беларуси,
Минск, Сурганова, 6
e-mail: {levin; rozin}@newman.bas-net.by

G.M. Levin, B.M. Rozin

OPTIMIZATION OF DURATIONS OF SEQUENTIAL-PARALLEL EXECUTION OF INTERSECTING OPERATION SETS

Mathematical model and method for the problem of optimization of durations of sequential-parallel execution of intersecting operation sets are proposed. The proposed method is based on the combination of approaches of parametric decomposition and dynamic programming.

ИНФОРМАТИКА

2012 **октябрь-декабрь** **№ 4**

УДК 519.7

Ю.В. Поттосин

ИТЕРАТИВНЫЙ МЕТОД ЭНЕРГОСБЕРЕГАЮЩЕГО КОДИРОВАНИЯ СОСТОЯНИЙ ДИСКРЕТНОГО АВТОМАТА

Рассматривается задача кодирования состояний дискретного автомата с целью уменьшения интенсивности переключений элементов памяти в реализующей схеме. Определение значений внутренних переменных сводится к задаче нахождения максимального разреза во взвешенном графе.

Введение

В последнее время при проектировании дискретных устройств управления на основе сверхбольших интегральных схем большое внимание уделяется проблеме снижения энергопотребления проектируемой схемы. Это обусловлено, с одной стороны, стремлением увеличить время действия источника энергии в портативных приборах и, с другой стороны, стремлением снизить остроту проблемы отвода тепла при проектировании сверхбольших интегральных схем. Поэтому одним из основных критериев оптимизации при проектировании дискретных устройств является величина потребляемой схемой энергии.

Как отмечено в работах [1, 2], потребляемая мощность схемы, построенной на основе КМОП-технологии, пропорциональна интенсивности переключений логических элементов и элементов памяти. Это дает возможность частично решать данную проблему на уровне логического проектирования. В частности, снижения энергопотребления можно добиваться при кодировании состояний автомата [3–5]. Кодировать состояния при этом надо таким образом, чтобы при переходе автомата из одного состояния в другое меняли свое состояние как можно меньше элементов памяти.

Первоначально критерием оптимизации при кодировании состояний была простота описания булева автомата, представляющего собой систему выходных булевых функций и функций возбуждения элементов памяти автомата [6]. Существует два подхода к решению этой задачи. Один из них предложен в статье [7] и направлен на упрощение двухуровневого представления системы булевых функций, т. е. получение как можно меньшего общего числа различных элементарных конъюнкций в системе дизъюнктивных нормальных форм. Другой подход [8] использует структуру разбиений на множестве состояний автомата, и целью кодирования при этом подходе является ослабление зависимости функций от их аргументов, т. е. считается, что функция тем проще, чем от меньшего числа аргументов она зависит.

Первый подход [7] положен в основу методов, предлагаемых в работах [5, 9], где кодирование состояний автомата сводится к оптимальной укладке графа поведения автомата в булевом гиперкубе. Этот подход рассчитан на общую модель автомата. В настоящей работе для решения задачи кодирования состояний используется подход, предложенный в работе [10] и описанный в работе [11]. Он близок к подходу [8], используемому также в работах [3, 4]. Предлагаемый далее метод ориентирован на модель автомата с абстрактным состоянием [6]. Метод назван итеративным, поскольку его выполнение состоит в последовательности итераций, на каждой из которых вводится внутренняя переменная, являющаяся компонентой кода состояния автомата.

1. Описание метода

Данный метод выгодно применять, когда для задания автомата используется модель с абстрактным состоянием и число его состояний невелико по сравнению с числом выходных булевых переменных.

Примером задания автомата с абстрактным состоянием являются следующие две матрицы:

$$\boldsymbol{U}=\begin{array}{c}\begin{matrix}x_1 & x_2 & x_3 & q & \end{matrix}\\ \begin{bmatrix} - & - & - & 1 & 1\\ 0 & - & - & 2 & 2\\ 1 & 0 & - & 2 & 3\\ 1 & 1 & - & 2 & 4\\ - & - & 0 & 3 & 5\\ - & 0 & 1 & 3 & 6\\ - & 1 & 1 & 3 & 7\\ - & - & - & 4 & 8\\ - & 0 & - & 5 & 9\\ - & 1 & 0 & 5 & 10\\ - & 1 & 1 & 5 & 11 \end{bmatrix}\end{array},\quad \boldsymbol{V}=\begin{bmatrix} q^+ & y_1 & y_2 & y_3 & y_4\\ 2 & 0 & - & 1 & 0\\ 3 & 1 & 0 & - & 1\\ 3 & 0 & - & 1 & 0\\ 5 & - & 1 & 1 & -\\ - & 0 & 1 & - & 0\\ 5 & 0 & 0 & - & 0\\ 2 & 0 & - & 0 & 0\\ 1 & 1 & 0 & 0 & 0\\ - & 0 & 1 & 1 & -\\ - & 0 & 1 & 0 & -\\ 4 & - & 0 & 0 & 0 \end{bmatrix},$$

где, например, вторая сверху пара строк представляет сопровождаемый выходными сигналами $y_1 = 1$, $y_2 = 0$, $y_4 = 1$ при неопределенном значении y_3 переход автомата из состояния 2 в состояние 3 при $x_1 = 0$ и любых значениях x_2 и x_3.

Рассмотрим троичные векторы, являющиеся частями строк матриц $\boldsymbol{U}$ и $\boldsymbol{V}$. *Противоречием* в задании функции y_i в матрицах $\boldsymbol{U}$ и $\boldsymbol{V}$ назовем пару неортогональных троичных векторов $\boldsymbol{x}_s$ и $\boldsymbol{x}_t$, которым соответствуют противоположные значения y_i (0 и 1). Например, функция y_4 имеет противоречия в виде пар строк (1, 2), (2, 5), (2, 6), (2, 7), (2, 8) и (2, 11).

Процесс кодирования состояний заключается в устранении подобных противоречий путем введения внутренних булевых переменных $z_1, z_2, \ldots, z_k$ и соответственно функций $z_1^+, z_2^+, \ldots, z_k^+$. При введении новых функций вводятся, возможно, и новые противоречия, которые устраняются тем же путем. Значения переменных $z_1, z_2, \ldots, z_k$ и функций $z_1^+, z_2^+, \ldots, z_k^+$ должны быть согласованы со значениями соответствующих многозначных переменных q и q^+, т. е. одним и тем же значениям переменной q или q^+ должны соответствовать одни и те же значения булевых переменных.

Результат указанного процесса в виде системы булевых функций в значительной степени зависит от порядка выбора функций для устранения противоречий. С целью упрощения получаемых функций рекомендуется для такого выбора использовать следующие соображения.

Текущая ситуация в данном процессе характеризуется функциями $y_1, y_2, \ldots, y_m$, $z_1^+, z_2^+, \ldots, z_l^+$ с противоречиями относительно аргументов $x_1, x_2, \ldots, x_n, z_1, z_2, \ldots, z_l$ (в начальной ситуации имеются только $y_1, y_2, \ldots, y_m$ и $x_1, x_2, \ldots, x_n$). Каждый шаг процесса заключается во введении внутренней переменной и придании ей значений, устраняющих противоречия в выбранной функции. При этом в матрицы $\boldsymbol{U}$ и $\boldsymbol{V}$ добавляется новый столбец. Вектор, представляющий часть строки матрицы $\boldsymbol{U}$ без компоненты q, обозначим $\boldsymbol{x}'$. Компонентами этого вектора являются переменные $x_1, x_2, \ldots, x_n$ и $z_1, z_2, \ldots, z_l$.

Для всех функций y_i и z_j^+ построим *взвешенные графы противоречий*. Вершинам этих графов соответствуют состояния автомата, и две вершины q_g и q_h связаны ребром, если и только если q_g и q_h находятся в строках матрицы $\boldsymbol{U}$, троичные векторы которых $\boldsymbol{x}'_s$ и $\boldsymbol{x}'_t$ образуют противоречие для y_i. Вес w_{gh} ребра $q_g q_h$ определяется как $w_{gh} = m_{gh}/p_{gh}$, где m_{gh} – число пар строк матрицы $\boldsymbol{U}$, соответствующих противоречиям, в каждой из которых присутствуют q_g и q_h, а p_{gh} – увеличенное на единицу (чтобы не было деления на 0) число переходов между состояниями q_g и q_h заданного автомата не важно в какую сторону.

В работе [5] этот фактор выражается через вероятности переходов между состояниями, вычисляемые по методу Чэпмена – Колмогорова [12]. Однако этот метод пригоден только для полностью определенного автомата, который из любого состояния может перейти через какую-то последовательность переходов в любое его состояние. Здесь же рассматривается более общий случай не полностью определенного автомата.

Для устранения противоречий рассматриваем матрицы смежности $\boldsymbol{D}(y_i)$ взвешенных графов противоречий и в первую очередь выбираем ту функцию y_i, матрица $\boldsymbol{D}(y_i)$ которой обладает максимальной суммой значений элементов.

Рассмотрим, как вводится новая переменная z_{l+1} со значениями на некоторых состояниях автомата. Здесь уместно обратиться к задаче нахождения максимального разреза взвешенного графа, т. е. такого разбиения множества вершин V на два подмножества A и B, чтобы сумма весов ребер, соединяющих вершины из A с вершинами из B, была максимальной. На состояниях, соответствующих вершинам из множества A, переменной z_{l+1} приписывается значение 0 (или 1), а на состояниях, соответствующих вершинам из B, – значение 1 (или 0). Для разбиения множества V на подмножества A и B используем «жадный» алгоритм из работы [13], который представляет собой последовательность итераций, на каждой из которых в подмножестве B выбирается вершина v и переносится в подмножество A. Начальными значениями являются $A = \varnothing$ и $B = V$, а вершина v выбирается следующим образом.

Пусть d – сумма весов ребер, инцидентных вершине v, и d_A – сумма весов ребер, соединяющих вершину v с вершинами из A. Перенос вершины v из B в A сопровождается изменением суммы весов ребер, соединяющих вершины из A с вершинами из B, на величину $d - 2d_A$. На первом шаге эта величина равна степени переносимой вершины, а на последующих шагах она может быть отрицательной. Каждый раз выбирается та вершина, для которой величина $d - 2d_A$ максимальна, и процесс заканчивается, когда для всех вершин из подмножества B эта величина перестает быть положительной.

2. Пример

Для рассматриваемого примера величины p_{gh} представим в виде следующей матрицы (в силу симметричности подобных матриц здесь и в дальнейшем представляем только их верхнюю половину):

$$\begin{array}{c} \begin{array}{cccc} 2 & 3 & 4 & 5 \end{array} \\ \left[\begin{array}{cccc} 9 & 1 & 9 & 1 \\ & 9 & 1 & 3 \\ & & 1 & 3 \\ & & & 3 \end{array}\right] \end{array} \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \end{array}.$$

Получим матрицы смежности взвешенных графов противоречий, элементами которых являются веса соответствующих ребер:

$$\boldsymbol{D}(y_1) = \begin{array}{c} \begin{array}{cccc} 2 & 3 & 4 & 5 \end{array} \\ \left[\begin{array}{cccc} 1/9 & 0 & 1/9 & 0 \\ & 3/9 & 1 & 2/3 \\ & & 3 & 0 \\ & & & 2/3 \end{array}\right] \end{array} \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \end{array}, \quad \boldsymbol{D}(y_2) = \begin{array}{c} \begin{array}{cccc} 2 & 3 & 4 & 5 \end{array} \\ \left[\begin{array}{cccc} 0 & 0 & 0 & 0 \\ & 1/9 & 1 & 1 \\ & & 1 & 1/3 \\ & & & 2/3 \end{array}\right] \end{array} \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \end{array},$$

$$\boldsymbol{D}(y_3) = \begin{array}{c} \begin{array}{cccc} 2 & 3 & 4 & 5 \end{array} \\ \left[\begin{array}{cccc} 0 & 1 & 1/9 & 2 \\ & 1/9 & 2 & 2/3 \\ & & 1 & 1/3 \\ & & & 1/3 \end{array}\right] \end{array} \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \end{array}, \quad \boldsymbol{D}(y_4) = \begin{array}{c} \begin{array}{cccc} 2 & 3 & 4 & 5 \end{array} \\ \left[\begin{array}{cccc} 1/9 & 0 & 0 & 0 \\ & 3/9 & 1 & 1/3 \\ & & 0 & 0 \\ & & & 0 \end{array}\right] \end{array} \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \end{array}.$$

Для функций y_1, y_2, y_3 и y_4 суммы значений равны соответственно $5\frac{8}{9}$, $4\frac{1}{9}$, $7\frac{5}{9}$ и $1\frac{7}{9}$. Делаем попытку устранить противоречия функции y_3, имеющей наибольший для этого показатель. Некоторые противоречия удается устранить введением внутренней переменной z_1.

Последовательность итераций при поиске максимального разреза графа, соответствующего функции y_3, отражена в табл. 1, где пустая клетка в столбце B говорит о том, что соответствующая вершина перешла в множество A. Наибольшие степень вершины и величины $d - 2d_A$ выделены. В результате получены $A = \{1, 4\}$ и $B = \{2, 3, 5\}$.

Таблица 1

Начальная ситуация		Шаг 1		Шаг 2	
B	d	B	$d - 2d_A$	B	$d - 2d_A$
1	$3\frac{1}{9}$	1	$\mathbf{2}\frac{7}{9}$		
2	$2\frac{7}{9}$	2	$-2\frac{7}{9}$	2	$-2\frac{7}{9}$
3	$2\frac{1}{9}$	3	$\frac{1}{9}$	3	$-1\frac{8}{9}$
4	$\mathbf{3}\frac{4}{9}$				
5	3	5	$2\frac{1}{3}$	5	$-1\frac{2}{3}$

Состояниям 1, 4 припишем $z_1 = 0$, а состояниям 2, 3, 5 – $z_1 = 1$. Появилась новая функция z_1^+, которая также имеет противоречия в задании. Ребра, соответствующие устраненным противоречиям, удаляются из всех графов. Получим следующие матрицы:

$$\boldsymbol{U} = \begin{array}{c} \begin{matrix} x_1 & x_2 & x_3 & z_1 & q \end{matrix} \\ \begin{bmatrix} - & - & - & 0 & 1 \\ 0 & - & - & 1 & 2 \\ 1 & 0 & - & 1 & 2 \\ 1 & 1 & - & 1 & 2 \\ - & - & 0 & 1 & 3 \\ - & 0 & 1 & 1 & 3 \\ - & 1 & 1 & 1 & 3 \\ - & - & - & 0 & 4 \\ - & 0 & - & 1 & 5 \\ - & 1 & 0 & 1 & 5 \\ - & 1 & 1 & 1 & 5 \end{bmatrix} \end{array}, \quad \boldsymbol{V} = \begin{array}{c} \begin{matrix} z_1^+ & q^+ & y_1 & y_2 & y_3 & y_4 \end{matrix} \\ \begin{bmatrix} 1 & 2 & 0 & - & 1 & 0 \\ 1 & 3 & 1 & 0 & - & 1 \\ 1 & 3 & 0 & - & 1 & 0 \\ 1 & 5 & - & 1 & 1 & - \\ - & - & 0 & 1 & - & 0 \\ 1 & 5 & 0 & 0 & - & 0 \\ 1 & 2 & 0 & - & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ - & - & 0 & 1 & 1 & - \\ - & - & 0 & 1 & 0 & - \\ 0 & 4 & - & 0 & 0 & 0 \end{bmatrix} \end{array},$$

$$\boldsymbol{D}(y_1) = \begin{array}{c} \begin{matrix} 2 & 3 & 4 & 5 \end{matrix} \\ \begin{bmatrix} 0 & 0 & 1/9 & 0 \\ & 3/9 & 0 & 2/3 \\ & & 0 & 0 \\ & & & 0 \end{bmatrix} \end{array} \begin{matrix} 1 \\ 2, \\ 3 \\ 4 \end{matrix}, \quad \boldsymbol{D}(y_2) = \begin{array}{c} \begin{matrix} 2 & 3 & 4 & 5 \end{matrix} \\ \begin{bmatrix} 0 & 0 & 0 & 0 \\ & 1/9 & 0 & 2/3 \\ & & 0 & 1/3 \\ & & & 0 \end{bmatrix} \end{array} \begin{matrix} 1 \\ 2, \\ 3 \\ 4 \end{matrix}, \quad \boldsymbol{D}(y_3) = \begin{array}{c} \begin{matrix} 2 & 3 & 4 & 5 \end{matrix} \\ \begin{bmatrix} 0 & 0 & 1/9 & 0 \\ & 1/9 & 0 & 2/3 \\ & & 0 & 1/3 \\ & & & 0 \end{bmatrix} \end{array} \begin{matrix} 1 \\ 2, \\ 3 \\ 4 \end{matrix},$$

$$\boldsymbol{D}(y_4) = \begin{array}{c} \begin{matrix} 2 & 3 & 4 & 5 \end{matrix} \\ \begin{bmatrix} 0 & 0 & 0 & 0 \\ & 3/9 & 0 & 1/3 \\ & & 0 & 0 \\ & & & 0 \end{bmatrix} \end{array} \begin{matrix} 1 \\ 2, \\ 3 \\ 4 \end{matrix}, \quad \boldsymbol{D}(z_1^+) = \begin{array}{c} \begin{matrix} 2 & 3 & 4 & 5 \end{matrix} \\ \begin{bmatrix} 0 & 0 & 1/9 & 0 \\ & 0 & 0 & 2/3 \\ & & 0 & 1/3 \\ & & & \end{bmatrix} \end{array} \begin{matrix} 1 \\ 2. \\ 3 \\ 4 \end{matrix}$$

Суммы элементов матриц $\boldsymbol{D}(y_1)$, $\boldsymbol{D}(y_2)$, $\boldsymbol{D}(y_3)$, $\boldsymbol{D}(y_4)$ и $\boldsymbol{D}(z_1^+)$ равны соответственно $1\frac{1}{9}$, $1\frac{1}{9}$, $1\frac{2}{9}$ 1 и $1\frac{1}{9}$. Видно, что наибольшим числом противоречий опять обладает функция y_3. Последо-

вательность итераций при поиске максимального разреза графа, соответствующего функции y_3, отражена в табл. 2.

Таблица 2

Начальная ситуация		Шаг 1		Шаг 2	
B	d	B	$d-2d_A$	B	$d-2d_A$
1	$\frac{1}{9}$	1	$\frac{1}{9}$		
2	$\frac{7}{9}$	2	$-\frac{5}{9}$	2	$-\frac{5}{9}$
3	$\frac{4}{9}$	3	$-\frac{2}{9}$	3	$-\frac{2}{9}$
4	$\frac{1}{9}$	4	$\frac{1}{9}$	4	$-\frac{1}{9}$
5	**1**				

Введем переменную z_2, придав ей значение 0 для состояний 1, 5 и значение 1 для состояний 2, 3, 4. Получим матрицы

$$\boldsymbol{U}=\begin{array}{c}\begin{matrix}x_1 & x_2 & x_3 & z_1 & z_2 & q\end{matrix}\\ \begin{bmatrix} - & - & - & 0 & 0 & 1\\ 0 & - & - & 1 & 1 & 2\\ 1 & 0 & - & 1 & 1 & 2\\ 1 & 1 & - & 1 & 1 & 2\\ - & - & 0 & 1 & 1 & 3\\ - & 0 & 1 & 1 & 1 & 3\\ - & 1 & 1 & 1 & 1 & 3\\ - & - & - & 0 & 1 & 4\\ - & 0 & - & 1 & 0 & 5\\ - & 1 & 0 & 1 & 0 & 5\\ - & 1 & 1 & 1 & 0 & 5 \end{bmatrix}\end{array},\quad \boldsymbol{V}=\begin{array}{c}\begin{matrix}z_1^+ & z_2^+ & q^+ & y_1 & y_2 & y_3 & y_4\end{matrix}\\ \begin{bmatrix} 1 & 1 & 2 & 0 & - & 1 & 0\\ 1 & 1 & 3 & 1 & 0 & - & 1\\ 1 & 1 & 3 & 0 & - & 1 & 0\\ 1 & 0 & 5 & - & 1 & 1 & -\\ - & - & - & 0 & 1 & - & 0\\ 1 & 0 & 5 & 0 & 0 & - & 0\\ 1 & 1 & 2 & 0 & - & 0 & 0\\ 0 & 0 & 1 & 1 & 0 & 0 & 0\\ - & - & - & 0 & 1 & 1 & -\\ - & - & - & 0 & 1 & 0 & -\\ 0 & 1 & 4 & - & 0 & 0 & 0 \end{bmatrix}\end{array},$$

$$\boldsymbol{D}(y_1)=\begin{array}{c}\begin{matrix}2 & 3 & 4 & 5\end{matrix}\\ \begin{bmatrix} 0 & 0 & 0 & 0\\ & 3/9 & 0 & 0\\ & & 0 & 0\\ & & & 0 \end{bmatrix}\end{array}\begin{matrix}1\\2\\3\\4\end{matrix},\quad \boldsymbol{D}(y_2)=\begin{array}{c}\begin{matrix}2 & 3 & 4 & 5\end{matrix}\\ \begin{bmatrix} 0 & 0 & 0 & 0\\ & 1/9 & 0 & 0\\ & & 0 & 0\\ & & & 0 \end{bmatrix}\end{array}\begin{matrix}1\\2\\3\\4\end{matrix},\quad \boldsymbol{D}(y_3)=\begin{array}{c}\begin{matrix}2 & 3 & 4 & 5\end{matrix}\\ \begin{bmatrix} 0 & 0 & 0 & 0\\ & 1/9 & 0 & 0\\ & & 0 & 0\\ & & & 0 \end{bmatrix}\end{array}\begin{matrix}1\\2\\3\\4\end{matrix},$$

$$\boldsymbol{D}(y_4)=\begin{array}{c}\begin{matrix}2 & 3 & 4 & 5\end{matrix}\\ \begin{bmatrix} 0 & 0 & 0 & 0\\ & 1/3 & 0 & 0\\ & & 0 & 0\\ & & & 0 \end{bmatrix}\end{array}\begin{matrix}1\\2\\3\\4\end{matrix},\quad \boldsymbol{D}(z_1^+)=\begin{array}{c}\begin{matrix}2 & 3 & 4 & 5\end{matrix}\\ \begin{bmatrix} 0 & 0 & 0 & 0\\ & 0 & 0 & 0\\ & & 0 & 0\\ & & & 0 \end{bmatrix}\end{array}\begin{matrix}1\\2\\3\\4\end{matrix},\quad \boldsymbol{D}(z_2^+)=\begin{array}{c}\begin{matrix}2 & 3 & 4 & 5\end{matrix}\\ \begin{bmatrix} 0 & 0 & 0 & 0\\ & 1/3 & 0 & 0\\ & & 0 & 0\\ & & & 0 \end{bmatrix}\end{array}\begin{matrix}1\\2\\3\\4\end{matrix}.$$

Для устранения оставшихся противоречий достаточно ввести переменную z_3 со значением 0 для состояния 2 и со значением 1 для состояния 3. Окончательно получим следующие матрицы:

$$\boldsymbol{U}=\begin{array}{c}\begin{matrix}x_1 & x_2 & x_3 & z_1 & z_2 & z_3\end{matrix}\\ \begin{bmatrix}- & - & - & 0 & 0 & -\\ 0 & - & - & 1 & 1 & 0\\ 1 & 0 & - & 1 & 1 & 0\\ 1 & 1 & - & 1 & 1 & 0\\ - & - & 0 & 1 & 1 & 1\\ - & 0 & 1 & 1 & 1 & 1\\ - & 1 & 1 & 1 & 1 & 1\\ - & - & - & 0 & 1 & -\\ - & 0 & - & 1 & 0 & -\\ - & 1 & 0 & 1 & 0 & -\\ - & 1 & 1 & 1 & 0 & -\end{bmatrix}\end{array},\quad \boldsymbol{V}=\begin{array}{c}\begin{matrix}z_1^{+} & z_2^{+} & z_3^{+} & y_1 & y_2 & y_3 & y_4\end{matrix}\\ \begin{bmatrix}1 & 1 & 0 & 0 & - & 1 & 0\\ 1 & 1 & 1 & 1 & 0 & - & 1\\ 1 & 1 & 1 & 0 & - & 1 & 0\\ 1 & 0 & - & - & 1 & 1 & -\\ - & - & - & 0 & 1 & - & 0\\ 1 & 0 & - & 0 & 0 & - & 0\\ 1 & 1 & 0 & 0 & - & 0 & 0\\ 0 & 0 & - & 1 & 0 & 0 & 0\\ - & - & - & 0 & 1 & 1 & -\\ - & - & - & 0 & 1 & 0 & -\\ 0 & 1 & - & - & 0 & 0 & 0\end{bmatrix}\end{array}.$$

Для оценки качества полученного результата введем величину $S=\sum_{i,j}P_{ij}s_{ij}$, где P_{ij} – число переходов между состояниями q_i и q_j в любую сторону, s_{ij} – число переключаемых элементов при переходе между состояниями q_i и q_j. Суммирование ведется по всем переходам между состояниями в заданном автомате. Будем считать, что чем меньше эта величина, тем лучше результат кодирования. В рассмотренном примере $S = 40$. Если кодировать состояния произвольно, например использовать двоичное представление последовательности натуральных чисел с нулем, то получим результат в виде матриц

$$\boldsymbol{U}=\begin{array}{c}\begin{matrix}x_1 & x_2 & x_3 & z_1 & z_2 & z_3\end{matrix}\\ \begin{bmatrix}- & - & - & 0 & 0 & 0\\ 0 & - & - & 0 & 0 & 1\\ 1 & 0 & - & 0 & 0 & 1\\ 1 & 1 & - & 0 & 0 & 1\\ - & - & 0 & 0 & 1 & 0\\ - & 0 & 1 & 0 & 1 & 0\\ - & 1 & 1 & 0 & 1 & 0\\ - & - & - & 0 & 1 & 1\\ - & 0 & - & 1 & - & -\\ - & 1 & 0 & 1 & - & -\\ - & 1 & 1 & 1 & - & -\end{bmatrix}\end{array},\quad \boldsymbol{V}=\begin{array}{c}\begin{matrix}z_1^{+} & z_2^{+} & z_3^{+} & y_1 & y_2 & y_3 & y_4\end{matrix}\\ \begin{bmatrix}0 & 0 & 1 & 0 & - & 1 & 0\\ 0 & 1 & 0 & 1 & 0 & - & 1\\ 0 & 1 & 0 & 0 & - & 1 & 0\\ 1 & - & - & - & 1 & 1 & -\\ - & - & - & 0 & 1 & - & 0\\ 1 & - & - & 0 & 0 & - & 0\\ 0 & 0 & 1 & 0 & - & 0 & 0\\ 0 & 0 & 0 & 1 & 0 & 0 & 0\\ - & - & - & 0 & 1 & 1 & -\\ - & - & - & 0 & 1 & 0 & -\\ 0 & 1 & 1 & - & 0 & 0 & 0\end{bmatrix}\end{array}.$$

Для этого варианта кодирования $S = 44$.

Заключение

Предлагаемый метод кодирования состояний автомата рассчитан на использование его в автоматизированной системе логического проектирования. Сравнение результатов кодирования состояний изложенным методом и методом произвольного кодирования состояний показывает, что применение первого метода позволяет снизить интенсивность переключений элементов памяти при переходах между состояниями в автомате. Минимизация интенсивности переключений не противоречит минимизации количества элементов в схеме, что также ведет к снижению энергопотребления. Предлагаемый метод допускает частичное совместное решение этих двух задач на этапе кодирования состояний. При этом, как уже было сказано, существует два критерия минимизации: число различных элементарных конъюнкций в дизъюнктивных нормальных формах функций и количество аргументов, от которых зависит отдельная функция. Выбор их определяется используемой элементной базой. Этим двум критериям

соответствуют два подхода к решению рассматриваемой задачи, один из которых, соответствующий второму упомянутому критерию, использован в данной работе.

Список литературы

1. Мурога, С. Системное проектирование сверхбольших интегральных схем. В 2-х кн. / С. Мурога. – М. : Мир, 1985. – Кн. 1. – 288 с.

2. Pedram, M. Power minimization in IC design: Principles and applications / M. Pedram // ACM Trans. Design Automat. Electron. Syst. – 1996. – Vol. 1. – P. 3–56.

3. Kashirova, L. State assignment of finite state machine for decrease of power dissipation / L. Kashirova, A. Keevallik, M. Meshkov // Second International Conference Computer-Aided Design of Discrete Devices, CAD DD'97, Minsk, Republic of Belarus, November 12–14, 1997. – Minsk : Institute of Engineering Cybernetics NASB, 1997. – Vol. 1. – P. 60–67.

4. Sudnitson, A. Partition search for FSM low power synthesis / A. Sudnitson // Fourth International Conference Computer-Aided Design of Discrete Devices, CAD DD'2001, Minsk, November 14–16, 2001. – Minsk : Institute of Engineering Cybernetics NASB, 2001. – Vol. 1. – P. 44–49.

5. Закревский, А.Д. Алгоритмы энергосберегающего кодирования состояний автомата / А.Д. Закревский // Информатика. – 2011. – № 1(29). – С. 68–78.

6. Закревский, А.Д. Логические основы проектирования дискретных устройств / А.Д. Закревский, Ю.В. Поттосин, Л.Д. Черемисинова. – М. : Физматлит, 2007. – 592 с.

7. Armstrong, D.B. A programmed algorithm for assigning internal codes for sequential machines / D.B. Armstrong // IRE Trans., EC-11. – 1962. – № 4. – P. 466–472.

8. Hartmanis, J. Algebraic Structure Theory of Sequential Machines / J. Hartmanis, R.E. Stearns. – N.Y. : Prentis-Hall Inc., 1966. – 208 p.

9. Поттосин, Ю.В. Кодирование состояний дискретного автомата, ориентированное на уменьшение энергопотребления реализующей схемы / Ю.В. Поттосин // Прикладная дискретная математика. – 2011. – № 4(14). – С. 62–71.

10. Поттосин, Ю.В. Итеративный способ кодирования состояний дискретного автомата / Ю.В. Поттосин // Автоматизация логического проектирования дискретных устройств : сб. науч.тр. – Минск : Ин-т техн. кибернетики АН БССР, 1980. – Вып. 2. – С. 16–26.

11. Поттосин, Ю.В. Основы теории проектирования цифровых устройств / Ю.В. Поттосин. – Saarbrücken : LAP LAMBERT Academic Publishing, 2011. – 336 с.

12. Macii, E. High-level power modeling, estimation and optimization / E. Macii, M. Pedram, F. Somenzi // IEEE Trans. on Comp.-Aided Design of IC and Systems. – 1998. – Vol. 17, № 11. – P. 1061–1079.

13. Закревский, А.Д. Раскраска графов при декомпозиции булевых функций / А.Д. Закревский // Логическое проектирование : сб. науч. тр. – Минск : Ин-т техн. кибернетики НАН Беларуси, 2000. – Вып. 5. – С. 83–97.

Поступила 20.06.2012

Объединенный институт проблем
информатики НАН Беларуси,
Минск, Сурганова, 6
e-mail: pott@newman.bas-net.by

Yu.V. Pottosin

ITERATIVE LOW POWER STATE ASSIGNMENT OF A DISCRETE AUTOMATON

The problem of state assignment of a discrete automaton with the purpose of decreasing the switching activity of memory elements in the implementing circuit is considered. The determination of values of internal variables is reduced to finding a maximal cut in a weighted graph.

УДК 658.512:519.87

В.И. Романов

ИЕРАРХИЧЕСКИЙ ПОДХОД К ТОПОЛОГИЧЕСКОМУ ПРОЕКТИРОВАНИЮ МИКРОСХЕМ

Предлагается иерархический подход к построению топологических эскизов кристаллов микросхем. Данный подход основывается на фрагментации эскиза схемы, при которой топология отдельных фрагментов, образующих иерархию, может быть описана автоматически в соответствии с предлагаемыми алгоритмами размещения и трассировки, что существенно сокращает общий объем проектных работ. На самом нижнем уровне иерархии используется параметрически настраиваемая топологическая библиотека макроэлементов, основанных на применении регулярных структур. На последующих уровнях иерархии предлагается использовать стандартизованное группирование компонентов нижних уровней.

Введение

Микроэлектроника поставляет элементную базу для средств вычислительной техники, управления, связи и т. д. Производство схем большой степени интеграции (СБИС), ориентированных на специальные применения, в настоящее время немыслимо без сложных комплексных систем автоматизированного проектирования (САПР).

Свобода в выборе элементного состава и топологии кристалла, характерная для заказных СБИС, позволяет добиваться высоких технологических характеристик реализуемого на кристалле устройства управления, но одновременно значительно усложняет процесс проектирования [1].

Процесс проектирования СБИС имеет многоэтапный характер. Проектная информация, представляющая собой в начале проектирования алгоритмическое либо функционально-логическое описание будущей схемы на языке регистровых передач (RTL-описание), проходит через ряд этапов проектирования, пока не будет преобразована в геометрическое описание слоев будущего кристалла (например, GDSII-описание), пригодное для технологического производства.

Разрабатываемый программный комплекс проектирования топологии иерархически организованных сетей макроэлементов заказных цифровых СБИС предназначен для автоматизированного построения топологии функциональных блоков управляющей логики цифровых микросхем, описание которых носит иерархический характер и основано на использовании как отдельных макроэлементов, так и сетей макроэлементов, выполненных в виде регулярных МОП (металл-оксид-полупроводник)-структур.

Необходимость развития именно этапа топологического проектирования определяется прежде всего тем, что использование эффективных методов размещения и трассировки элементов СБИС позволяет уменьшить площадь кристалла. Однако при создании таких САПР появляются значительные технологические трудности. Выбор подходящего метода оптимизации площади, оценка его возможностей, областей применимости по различным критериям требуют привлечения весьма квалифицированных экспертов-разработчиков топологии.

Для данной области характерна также проблема все более увеличивающейся размерности решаемых задач, которая порождает новые технологические трудности на этапе формирования топологического эскиза разрабатываемой микросхемы.

Под проектным эскизом понимается изображение, отображающее физическое размещение на кристалле элементов микросхемы, их информационных связей, силовых линий земли (Gnd) и питания (Vcc).

1. Базовые требования

Ключевым элементом проекта заказной микросхемы является топологический эскиз кристалла, передаваемый на производство. Однако для его получения необходимо провести боль-

шую подготовительную работу, обеспечивающую трансформацию исходных спецификаций микросхемы, выполненных в некотором специализированном языке (например, в языке VHDL [2]), к описанию формата эскиза топологии, на основании которого уже может быть получена производственная спецификация проекта кристалла в некотором другом языке, например GDSII. Отсюда следует, что *создаваемая САПР должна быть сквозной и обеспечивать преобразования описания проектируемой схемы от исходного, выполненного на языке спецификации (VHDL), до итогового топологического описания (GDSII)*.

Известно, что, как и любое другое производство, изготовление кристаллов основывается на применяемой технологии, которая устанавливает группу технологических ограничений, продиктованных в основном использованием конкретного оборудования. Такие характеристики указывают минимальные контролируемые размеры топологии (контактных окон в оксиде кремния, затворов в транзисторах и т. д.). При проектировании микросхем на кристаллах часть норм связана с различными геометрическими характеристиками, главная из которых – размер точки – представляет собой единицу измерений, отражаемую на эскизе (изображении) кристалла. Все прочие количественные характеристики технологии часто определяются относительно этой базовой величины.

Таким образом, второе базовое требование к разрабатываемому программному комплексу состоит в том, что *создаваемая САПР должна поддерживать параметрическую настройку заказываемых технологических ограничений и обеспечивать комфортную работу с проектом при их применении*.

В настоящее время вопросы собственно синтеза микросхемы решаются с применением достаточно универсальных сквозных промышленных (коммерческих) систем проектирования заказных СБИС (например, LeonardoSpectrum [3]). Кроме того, существует целый ряд готовых программных систем, поддерживающих решение задач синтеза [1]. В этой связи основное внимание при разработке программного комплекса должно быть уделено этапу построения топологического представления проектируемой микросхемы. Программный комплекс проектирования топологии иерархически организованных сетей макроэлементов заказных цифровых СБИС является развитием САПР CLTT [4].

Основная идея, реализованная в рамках САПР CLTT, основана на описании микросхемы в виде совокупности макроэлементов, определяемых параметрически и отражающих заранее топологически определенные структуры. Разработка элементов топологической библиотеки макроструктур является отдельной задачей и в рамках САПР не рассматривается – фрагменты эскиза топологии проектируемой схемы строятся на основе параметрически настраиваемых заготовок макроэлементов.

2. Проблемы визуализации и масштабирования

В рамках программного комплекса в качестве компонентов проекта микросхемы могут выступать параметризуемые по числу входных, выходных переменных и числу промежуточных шин программируемые структуры типа программируемых логических матриц (ПЛМ), использующих параллельные соединения транзисторов, регулярные схемы на основе последовательно соединенных транзисторов (РМОП-схемы), постоянные запоминающие устройства (ПЗУ), IP-блоки с уже разработанной топологией и другие библиотечные макроэлементы СБИС. Иерархическая организация сетей макроэлементов предполагает, что некоторые из подсетей представляются в виде сетей из других макроэлементов или сетей библиотечных элементов.

Прежде всего будем предполагать, что технология производства кристаллов основана на применении нескольких проводящих слоев: как минимум, пары слоев металлизации и одного слоя поликремния, причем с целью оптимизации функционирования проектируемой схемы передача сигналов и электроэнергии между элементами будет осуществляться по одному из металлических слоев (рис. 1).

Исходя из технологии производства кристаллов (напыление, травление, фотолитография и проч.) можно сказать, что описывающее их изображение в каждом слое носит бинарный характер. Система проектирования должна обеспечивать возможность наблюдения технологического эскиза как по каждому из слоев металла, так и одновременно по всем слоям.

Предположим, что размер кристалла соответствует квадрату со стороной 15 мм. В современных условиях размер отдельной точки можно условно принять равным норме проектирования (например, 60 нм) и для определения реального топологического изображения в таком

квадрате только для одного слоя необходимо задать каждую из 62 500 000 000 точек (15 мм = = 15 000 мкм = 15 000 000 нм / 60 нм =250 000 точек).

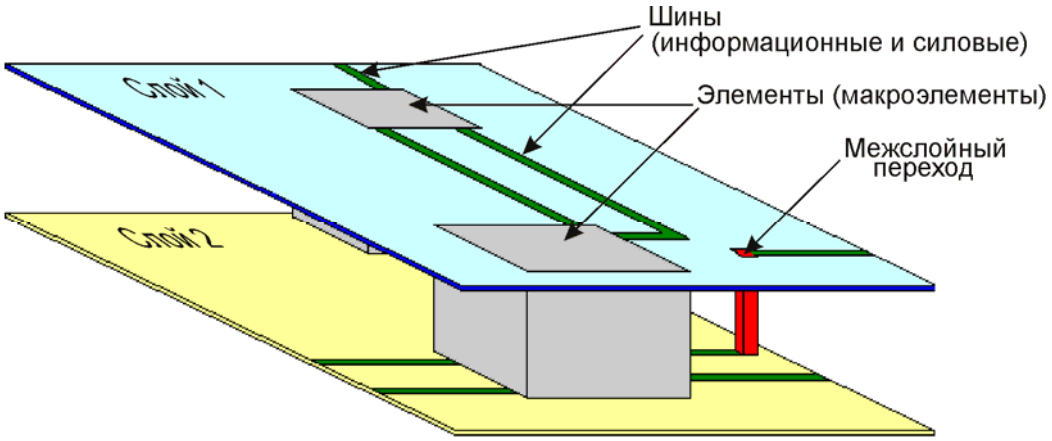


Рис. 1. Многослойная организация кристалла при проектировании

Отсюда можно сделать несколько выводов:

1. Итоговое изображение при проектировании, несмотря даже на свою бинарность, физически не может быть определено в растровой форме. Для его определения применим только векторный способ описания. Напомним, что в растровых изображениях определяется каждая точка отображаемой поверхности, а в векторных определяются функции построения, описывающие размещаемые на поверхности фигуры. При построении эскизов микросхем изображение может быть определено через множество отрезков прямых линий, привязанных к поверхности своими концевыми точками.

2. Линейного масштабирования всего изображения недостаточно для обеспечения комфортной работы по его построению: реализация деталей требует большего увеличения, при этом теряется обозримость эскиза и становится невозможно оперировать с «размытыми» на поверхности объектами, например информационными или силовыми шинами.

3. При разработке в САПР средств отображения проектируемых микросхем одним из критериев качества следует установить минимизацию объема обрабатываемой информации.

Одним из способов сокращения объемов перерабатываемой информации является декомпозиция изображения на линейно масштабируемые фрагменты – топологические композиты. Каждый из композитов представляется прямоугольником, на сторонах которого определены все его внешние связи. Фиксация этих позиций в масштабах всего изображения позволяет выполнить проектирование всей схемы без потери информации. Допустив повторную применимость проводимой структурной декомпозиции к отдельным фрагментам, в итоге получаем иерархически описываемый топологический эскиз. На каждом уровне иерархии проектировщик имеет дело только с ограниченным множеством структурных элементов и их связей, что способствует проведению более качественного проектирования.

Другим способом сокращения объемов перерабатываемой информации является группирование отдельных элементов в параметризуемые структуры – макроэлементы, для которых разрабатываются «стандартные» способы размещения на топологическом эскизе. В результате их использования задачи разводки связей отдельных элементов существенно теряют в своей размерности.

В рамках создаваемого программного комплекса предполагается совместить оба способа. На рис. 2 показана иерархическая организация технологического эскиза топологии микросхемы.

Естественно, сам подход к проектированию топологии эскиза на основе иерархии появился давно. Его различные исходные варианты [5, 6] со временем уточнялись с учетом дополнительных условий многослойности проектируемых кристаллов [7, 8]. Однако в большинстве работ собственно иерархия строилась по принципу сверху вниз, когда на общей «площадке» топо-

логии выбирался фрагмент (по месту в общем рисунке или по составу «участников») и таким образом осуществлялось разложение общей схемы на компоненты [9–11].

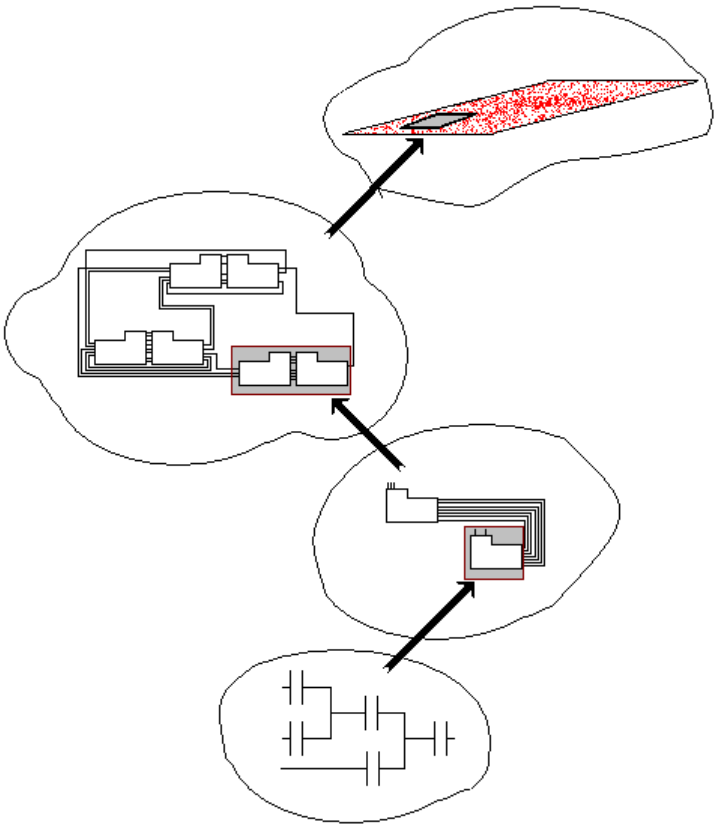


Рис. 2. Уровни иерархии в топологическом эскизе

Если рассматривать отдельно только собственно топологический редактор, служащий для подготовки результирующего эскиза, то следует отметить, что существующие аналоги предназначены для проектирования схемы «с нуля», когда содержание эскиза определяется проектировщиком последовательно путем внесения в эскиз отдельных элементов и их связей.

В рассматриваемом варианте топологический редактор оперирует с заранее заданной структурой элементов схемы и их связей – проектировщик в этом случае лишен возможности определения схемы «с лету» и не может ни добавить, ни удалить никаких элементов изображения, вопрос сводится только к выбору их расположения на плоскости эскиза.

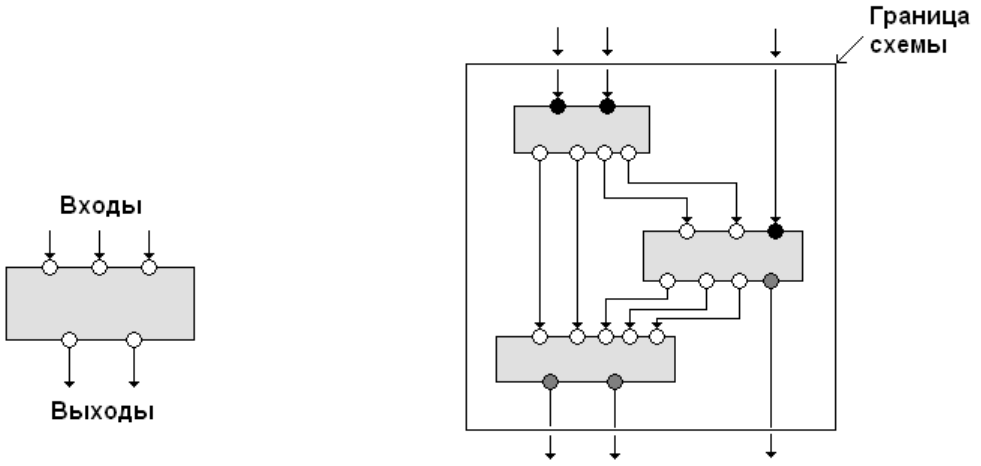
3. Математическая модель

Результатом выполненного логического синтеза является описание проектируемой схемы в виде сети взаимодействующих элементов.

Каждый из элементов сети характеризуется уникальным идентификатором; типом, определяющим реализуемую им логическую функцию, и наборами входных и выходных портов, через которые на элемент подаются и исходят сигналы (рис. 3). Информационная связь пары элементов однозначно идентифицируется кортежем из четырех составляющих: имя первого элемента – имя его выходного порта – имя второго элемента – имя его входного порта. Для всей сети определены подмножества внешних входных и выходных портов, сигналы на которых появляются извне описывающей проектируемую схему сети элементов.

Естественно, что изображенная на рис. 3 ситуация, когда все входные порты размещены на одной стороне элемента, а все выходные напротив, – это только частный случай размещения

портов. В общем случае их расположение на прямоугольном отображении элемента является произвольным и определяется технологией его создания.



а) *б)*

Рис. 3. Позиционирование портов: *а)* для отдельного элемента; *б)* для внешних входов и выходов схемы

Построенная сеть моделируется размеченным графом $G=(V, E)$, в котором множество вершин $V = (v_1, v_2, \ldots, v_n)$ соответствует множеству элементов схемы, а множество дуг $E =(e_1, e_2, \ldots, e_m)$ задает множество информационных связей элементов из E. Отдельно определяется множество входных $I(G)$ и выходных $O(G)$ портов схемы. Элементы этих множеств описываются возникающими информационными соединениями, характеризуемыми связанными с «внешностью» элементами схемы – парами вида <имя элемента – имя его порта>.

В рамках вводимой системы обозначений каждый элемент схемы v_i характеризуется своим именем $N(v_i)$, типом $T(v_i)$ и двумя множествами портов, соответствующих входным $I(v_i)$ и выходным $O(v_i)$ сигналам. Каждая внутренняя связь k-го и j-го элементов – дуга графа G – описывается четверкой $N(v_k)$, $i_q(v_k)$, $N(v_j)$, $o_p(v_j)$, в которой $i_q(v_k) \in I(v_k)$, $o_p(v_j) \in O(v_j)$.

Используемый при описании связи порт, не принадлежащий множеству внешних портов схемы $I(G) \cup O(G)$, определяется следующим образом:

– принадлежит какому-либо элементу схемы v_i и идентифицируется его именем $N(v_i)$;

– обладает собственным идентификационным именем;

– характеризуется направленностью (вход или выход);

– характеризуется назначением (информационный, синхронизации, питания, земли);

– характеризуется стороной размещения на габарите элемента (сверху, снизу, справа, слева);

– характеризуется значением позиционного смещения по выбранной стороне элемента (смещение рассматривается в реальных единицах размера, отмеряемых по ходу движения часовой стрелки при обходе прямоугольного габарита элемента).

Элементы ранее упомянутых множеств внешних портов $I(G)$ и $O(G)$ – входов и выходов схемы – описываются парами <элемент-порт> : $(N(v_k), i_q(v_k)) \in I(G)$ и $(N(v_j), o_p(v_j)) \in O(G)$ соответственно. Расположение самих внешних портов на границах схемы доопределяется в процессе топологического проектирования.

Задача построения топологического эскиза схемы состоит в координатной привязке к плоскости всех элементов схемы и соединяющих их информационных и силовых шин. Перечислим ряд присутствующих при этом дополнений и ограничений:

1. При размещении схемы на плоскости предполагается, что все ее компоненты будут укладываться в пределах некоторого прямоугольного фрагмента плоскости, имеющего обычно примерно равные длину и ширину и в дальнейшем называемого *площадкой*.

2. Площадка может содержать как всю схему, так и ее отдельный фрагмент – *композит*, включающий, по крайней мере, один ее элемент.

3. Каждый из композитов проектируется отдельно. При проектировании определяется как внутреннее устройство композита, так и его внешнее представление. Все композиты стро-

ятся по принципу отображения отдельного элемента: внутри «черный ящик» невидимого снаружи содержимого, на границах – порты, определенные через пару имен <имя композита, имя одного из его портов> и месторасположение на одной из сторон композита. Такой способ определения композита дает возможности абстрагироваться от его внутреннего устройства при проектировании более высокого уровня иерархии отображения схемы.

4. Кроме элементов на топологическом эскизе должны быть представлены силовые и информационные шины. Все шины обладают шириной, которая определяется используемыми нормами проектирования и другими технологическими ограничениями. Любая шина описывается множеством своих осевых отрезков, размещенных на плоскости либо горизонтально, либо вертикально. В корректно описанной шине ее горизонтальные и вертикальные отрезки чередуются.

5. В отличие от информационных шин силовые шины, к которым относятся линии земли и питания, не описываются на этапе логического синтеза. Соответствующие им элементы в представляемой графовой модели отображаются неориентированными ребрами, а не дугами, определяющими информационные шины.

6. Добавление силовых составляющих к результатам логического синтеза осуществляется в соответствии с общим правилом: силовые шины должны достигать каждого элемента схемы. Их ширина в общем случае может быть рассчитана в соответствии с конкретными технологическими условиями.

7. Внешние порты схемы будут определены точками, лежащими на границе – пересечении осевой линии шины с границей площадки.

8. Разрабатываемая модель топологического эскиза опирается на предположение наличия двух слоев металлизации. Фактически это означает, что для размещенной на эскизе шины можно указать последовательность сегментов, состоящих из отрезков шины, размещенных на одном слое металлизации. Переходы с одного слоя на другой могут быть выполнены при помощи размещения на эскизе специально описываемых межслойных контактов. По сравнению с точками ветвления реализация межслойных контактов должна удовлетворять ряду дополнительных технологических ограничений.

Таким образом, в целом на входе этапа топологического проектирования схема описывается:

– смешанным графом G, вершины которого соответствуют элементам схемы, дуги – информационным связям, ребра – силовым шинам, обеспечивающим линии земли и питания;

– множествами входов ($I(G)$) и выходов ($O(G)$), которые задают приемники и источники информационных сигналов, связанных с функционированием схемы;

– множествами внешних силовых контактов линий земли и питания, в которых каждый отдельный элемент задается точкой пересечения осевой линии силовой шины с границами площадки;

– набором частично формальных, а частично и неформальных технологических требований и ограничений на создаваемый топологический эскиз и процесс его построения.

4. Оформление иерархической структуры

Предлагаемый подход к построению иерархии основывается на группировании элементов по композитам с целью их укрупнения и сведения задачи построения топологического эскиза схемы к его представлению на основании построенных композитов.

На самом нижнем уровне иерархии в рамках разрабатываемого программного комплекса, занимаясь логическим проектированием, пользователь может определить множество подграфов $M = (M_1, M_2, \ldots, M_k)$, каждый из которых в некотором смысле выгодно реализовать при помощи параметрически определяемых макроэлементов – специальных структур, ориентированных на реализацию сложных логических функций. Каждый из подграфов M_j «забирает в себя» некоторое множество исходных элементов и обеспечивающих их информационных линий связи. После этого в общей модели схемы элемент M_j находит свою топологическую реализацию в виде отдельного композита, т. е. отображается прямоугольником с определенными на его границах контактами, информационными и силовыми.

Использование подобных регулярных схем облегчает решение задач диагностики и логического проектирования и позволяет проводить автоматическую генерацию топологии по структурному либо функциональному описанию.

Основной процесс проектирования топологии схемы на основе макроэлементов в форме матричных регулярных структур включает следующие этапы:

1) по функциональному описанию схемы генерируется структурное описание регулярных блоков;

2) для каждого макроэлемента согласно его структурному описанию определяется символьное представление его топологии;

3) формирование послойной топологии композитов, соответствующих макроэлементам, осуществляется путем компиляции из параметризированных послойных топологических фрагментов (элементов топологической библиотеки базовой регулярной структуры).

В результате задача размещения графа G трансформируется в задачу размещения графа $G^{(1)}(V^{(1)}, E^{(1)})$ значительно меньшей размерности, т. е. такого, что $|V^{(1)}| << |V|$, $|E^{(1)}| << |E|$.

Одним из приемов, обеспечивающих укрупнение компонентов эскиза, является объединение в одном композите нескольких «мелких» элементов, например транзисторов, не вписывающихся по каким-то соображениям в состав выделенных на графе макроэлементов. Такой композит будем называть *блоком нерегулярной логики*. В рамках одного проекта таких композитов может быть несколько, каждый из них проектируется отдельно. Принадлежность того или иного элемента тому или иному композиту диктуется соображениями сокращения длины информационных и силовых шин.

Следующим видом «стандартных» композитов является так называемый *расширенный макроэлемент,* представляющий собой совокупность некоторого макроэлемента и блока нерегулярной логики, в составе которого присутствуют логические элементы, информационно связанные по входам или выходам только с выбранным макроэлементом. Очевидно, что совместное расположение элементов-участников композита позволяет существенно снизить длины информационных соединений.

Еще одним видом «стандартных» композитов являются *парные макроэлементы* или *парные расширенные макроэлементы*: здесь речь идет о группировании пары однотипных макроэлементов примерно равного размера (только в этом случае можно рассчитывать на существенный эффект группирования).

Важно заметить, что проектирование исходного представления любого вида из описанных композитов может быть выполнено автоматически – построен фрагмент топологического эскиза, на котором размещены элементы и все виды связей: информационные и силовые.

В любом случае в качестве результата построения множества подграфов $F = (F_1, F_2, \ldots, F_p)$, каждый из которых проектируется в виде отдельного композита, возникает новый уровень иерархии как граф $G^{(2)}(V^{(2)},E^{(2)})$ размерности, меньшей размерности графа $G^{(1)}$.

В дальнейшем предполагается реализация процедуры построения однородных (по составу представленных в них типов макроэлементов) сетей, размещаемых в композитах следующего уровня иерархии.

Заключение

Описанный подход предлагается в качестве формальной основы для разрабатываемого программного комплекса проектирования топологии иерархически организованных сетей макроэлементов заказных цифровых СБИС. Этот комплекс является дальнейшим развитием разработанной ранее САПР CLTT [4], и в настоящее время осуществляется подготовка его рабочей версии. Указанная САПР CLTT выступает в роли испытательного полигона при создании нового программного обеспечения, что создает хорошие предпосылки как по срокам проведения разработки, так и по качеству реализуемых алгоритмов, поскольку поддерживается возможность их практического исследования на самых ранних стадиях проектирования. Ряд представленных в подходе идей, в частности организация топологии отдельных макроэлементов, уже получил апробацию в рамках системы CLTT.

Список литературы

1. Рабаи, Ж.М. Цифровые интегральные схемы / Ж.М. Рабаи, А. Чандракасан, Б. Николич. – 2-е изд.; пер. с англ. – М. : Изд. дом «Вильямс», 2007. – 912 с.

2. Суворова, Е.А. Проектирование цифровых систем на VHDL / Е.А. Суворова, Ю.Е. Шейнин. – СПб. : БХВ-Петербург, 2003. – 576 с.

3. Бибило, П.Н. Системы проектирования интегральных схем на основе языка VHDL, StateCAD, ModelSim, LeonardoSpectrum / П.Н. Бибило. – М. : СОЛОН-Пресс, 2005. – 384 с.

4. Система CLTT проектирования топологии функциональных блоков заказных цифровых СБИС / П.Н. Бибило [и др.] // Информационные технологии. – 2011. – № 1. – С. 8–14.

5. Lengauer, T. Exploiting hierarchy in VLSI design / T. Lengauer // VLSI Algorithms and Architectures Lecture Notes in Computer Science. – 1986. – Vol. 227. – P. 180–193.

6. Mlynek, D. Design of the VLSI systems / D. Mlynek, J. Leblebisi [Электронный ресурс]. – Режим доступа : http://lsmwww.epfl.ch/Education /former/2002-2003/VLSIDesign/. – Дата доступа : 23.02.2012.

7. Hierarchical BSG Floorplan for Hierarchical VLSI Circuit Design / Z.L. Wu [et al.] // IEICE Transactions. – 2000. – Vol. J83-A, № 10. – P. 1161–1168.

8. Adja, S.N. Fixed-Outline Floorplanning: Enabling Hierarchical Design / S.N. Adja, I.L. Markov // IEEE Transactions on very large scale Integration (VLSI) systems. – 2003. – Vol. 11, № 6. – P. 1120–1135.

9. Гаврилов, С.В. Средства проектирования полузаказных микросхем / С.В. Гаврилов, А.Н. Денисов, В.В. Коняхин [Электронный ресурс]. – Режим доступа : http:// http://www.asic.ru/publ.html#9. – Дата доступа : 22.06.2012.

10. Rubin, S.M. Computer Aids for VLSI Design / S.M. Rubin [Электронный ресурс]. – Режим доступа : http://www.rulabinsky.com/cavd/. – Дата доступа : 22.06.2012.

11. LayoutEditor – редактор топологий интегральных схем [Электронный ресурс]. – Режим доступа : http://www.eurointech.ru/layedit. – Дата доступа : 22.06.2012.

Поступила 28.06.2012

Объединенный институт проблем
информатики НАН Беларуси,
Минск, Сурганова, 6
e-mail: rom@newman.bas-net.by

V.I. Romanov

HIERARCHICAL APPROACH TO TOPOLOGICAL DESIGN OF INTEGRATED CIRCUITS

A hierarchical approach to topological design of integrated circuits drafts is proposed. The approach is based on the draft fragmentation, where the layout of different hierarchical blocks can be automatically described in accordance with the proposed placing and wiring algorithms aiming at reducing the design work. A customizable topological library of macro elements based on regular structures is used on the lowest hierarchical level. Grouping of standardized components of lower levels is proposed to apply at higher hierarchy levels.

УДК 681.325

Л.Д. Черемисинова, И.П. Логинова

СВЕРТКА РЕГУЛЯРНЫХ МАТРИЧНЫХ СТРУКТУР ЗАКАЗНЫХ СБИС МЕТОДОМ МОДЕЛИРОВАНИЯ ОТЖИГА

Рассматривается задача топологической оптимизации программируемых логических матриц методом свертки. Предлагаются алгоритмы многократной и простой сверток регулярных структур СБИС на основе моделирования отжига, позволяющие находить оптимальное или близкое к нему решение задачи свертки. Приводятся результаты исследования предложенных алгоритмов свертки.

Введение

При проектировании управляющей логики широко используются двухмерные матричные структуры, на пересечении строк и столбцов которых могут находиться транзисторы. Такие структуры характеризуются высокой степенью регулярности топологии и внутренних связей, что делает возможным автоматическую генерацию топологии по их функциональному описанию, а также облегчает решение задач диагностики и логического проектирования. Широко используемыми регулярными структурами [1] при проектировании СБИС являются программируемые логические матрицы (ПЛМ), матричные структуры на базе последовательно соединенных МОП-транзисторов (РМОП-структуры), матрицы Вайнбергера. Существенным недостатком регулярных структур на этапе топологического проектирования является то, что они, имея регулярную организацию, проигрывают многоуровневым реализациям на основе произвольной логики по площади, занимаемой на кристалле, за счет неэффективного ее использования. Последнее выражается в сильной разреженности матричных структур: в среднем около половины транзисторов в матрицах программирования больших структур в реальных схемах не используется.

Одним из наиболее эффективных методов сокращения площади регулярных структур является широко применяемый метод топологической оптимизации матричных структур, известный как свертка [1–4]. Свертка основана на разрыве шин матричной структуры и реализации на одной вертикальной (и/или горизонтальной) шине двух или более ее столбцов (и/или строк). В первом случае свертка называется простой, во втором многократной. Алгоритмы свертки, не изменяя функциональности матричной структуры, основаны на поиске оптимального переупорядочения и совмещения ее столбцов и строк.

Центральной частью любой регулярной структуры (ее ядром) является двухмерная матрица, столбцы которой представляют собой линии подвода входных сигналов, а в строках реализуются конъюнкции входных переменных. Конъюнкцию образуют те переменные, для которых на пересечении соответствующих столбцов со строкой находятся транзисторы. В настоящей работе рассматривается задача сокращения площади такой двухмерной матрицы путем ее свертки. Предлагается метод многократной столбцовой свертки регулярных структур на основе процедуры моделирования отжига и обобщение этого метода на случай простой свертки. В статье рассматривается также задача ограниченной свертки (наличие ограничений влияет на ее вид) на примере свертки регулярной структуры типа ПЛМ, которая предназначена для реализации системы дизъюнктивных нормальных форм булевых функций. Особенностью ПЛМ является то, что она состоит из двух транзисторных матриц И и ИЛИ, столбцам которых соответствуют входные и выходные переменные. Для этого типа регулярных структур приводятся результаты экспериментального исследования методов свертки и сравнительная оценка степени сокращения площади при использовании многократной и простой сверток.

1. Оптимизация площади матричной структуры методом свертки

Формальной моделью матричной структуры при решении задачи свертки является булева матрица $\boldsymbol{B}$, называемая структурной и имеющая множества $C(\boldsymbol{B}) = \{c_1,c_2,\ldots,c_n\}$ столбцов и $R(\boldsymbol{B}) = \{r_1,r_2,\ldots,r_m\}$ строк. Единичный элемент матрицы $b_i^j \in \boldsymbol{B}$ говорит о наличии транзистора на пересечении строки r_i и столбца c_j матричной структуры. Каждый столбец $c_j \in C(\boldsymbol{B})$ порождает множество $R(c_j)$ строк, которые он пересекает: $r_i \in R(c_j) \leftrightarrow b_i^j = 1$.

Например, матрице И для ПЛМ (рис. 1) соответствует структурная матрица $\boldsymbol{B}$, имеющая 8 строк и 14 столбцов, соответствующих входным сигналам в прямом и инверсном виде (рис. 2).

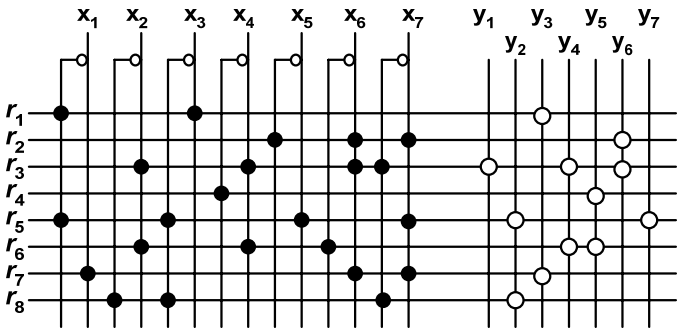


Рис. 1. Пример матричной структуры ПЛМ

	$\bar{x}_1$	x_1	$\bar{x}_2$	x_2	$\bar{x}_3$	x_3	$\bar{x}_4$	x_4	$\bar{x}_5$	x_5	$\bar{x}_6$	x_6	$\bar{x}_7$
	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_1	c_1	c_1	c_1
r_1	1	0	0	0	0	1	0	0	0	0	0	0	0
r_2	0	0	0	0	0	0	0	0	1	0	0	1	0
r_3	0	0	0	1	0	0	0	1	0	0	0	1	1
r_4	0	0	0	0	0	0	1	0	0	0	0	0	0
r_5	1	0	0	0	1	0	0	0	0	1	0	0	0
r_6	0	0	0	1	0	0	0	1	0	0	1	0	0
r_7	0	1	0	0	0	0	0	0	0	0	0	1	0
r_8	0	0	1	0	1	0	0	0	0	0	0	0	1

Рис. 2. Булева матрица $\boldsymbol{B}$, являющаяся моделью структуры И ПЛМ

При проведении свертки матричной структуры ищутся столбцы/строки матрицы $\boldsymbol{B}$, которые могут быть размещены на одной вертикальной/горизонтальной шине при ее топологической реализации. Непересекающиеся столбцы c_k и c_l называются совместимыми, если $R(c_k) \cap R(c_l) = \varnothing$. Набором столбцовой свертки $l^c_k = <c_{k1}, c_{k2},\ldots, c_{ku}>$ $(1 < u \le n)$ называется упорядоченное множество попарно совместимых столбцов c_{ij}. Набор свертки $l^c_k = <c_{k1}, c_{k2}>$ мощности два называется парой свертки. Столбцы, входящие в любой набор свертки l^c_k, не пересекаются и могут быть реализованы на одной вертикальной шине матричной структуры, располагаясь на ней следующим образом: столбцы c_{k1} выше c_{k2}, c_{k2} выше c_{k3} и т. д., $c_{k,u-1}$ выше c_{ku}. Реализация набора свертки l^c_k приводит к перестановке на множестве строк матрицы $\boldsymbol{B}$: $R(c_{k1}) > R(c_{k2})$ – строки из $R(c_{k1})$ располагаются выше строк из $R(c_{k2})$; $R(c_{k2}) > R(c_{k3})$ – строки из $R(c_{k2})$ располагаются выше строк из $R(c_{k3})$; $R(c_{k,u-1}) > R(c_{ku})$ – строки из $R(c_{k,u-1})$ располагаются выше строк из $R(c_{ku})$, порождая отношение на множестве строк $R(\boldsymbol{B})$:

$$P^r(l^c_k) = \bigcup_{i<j} (R(c_{ki}) \times R(c_{kj})),$$

т. е. $P^r(l^c_k) = \{ r_p \times r_q \,|\, r_p \in R(c_{ki}),\ r_q \in R(c_{kj}),\ i < j \}$.

Данное отношение является отношением частичного порядка, так как оно иррефлексивно, асимметрично и транзитивно по определению. Множеством столбцовой свертки (МСС) $L^c = \{l^c_1, l^c_2,\ldots, l^c_k\}$ называется множество непересекающихся наборов свертки. Число столбцов, входящих во все наборы $l^c_i \in L^c$, называется размером МСС L^c, а число k наборов – мощностью МСС L^c. МСС L^c порождает множество отношений порядка на множестве строк, которое задается объединением отношений порядка $P^r(l^c)$, порождаемых наборами $l^c_k \in L^c$:

$$P^r(L^c) = \bigcup_{i=1}^{k} (P^r(l^c_i)).$$

Отношение $P^r(L^c)$ является иррефлексивным, асимметричным, но в общем случае нетранзитивным. Транзитивное замыкание $T(P^r)$ отношения $P^r(L^c)$ является иррефлексивным, транзитивным, но может быть неасимметричным. Доказано в [3], что МСС L^c является реализуемым топологически (соответствующей свернутой матричной структурой), если транзитивное замы-

кание $T(P^r)$ отношения $P^r(L^c)$ порождает отношение частичного порядка на $R(\boldsymbol{B})$, т. е. $T(P^r)$ является асимметричным. Другими словами, МСС L^c является реализуемым топологически, если существует линейный порядок на множестве строк $R(\boldsymbol{B})$, задаваемый отношением $T(P^r)$. Реализуемое множество свертки L^c точно определяет структуру свернутой матрицы, а ее размер определяется размером свертки: число наборов свертки в L^c соответствует числу шин, которые заменяют $\sum_{i=1}^{k} |l^c_i|$ столбцов исходной матричной структуры.

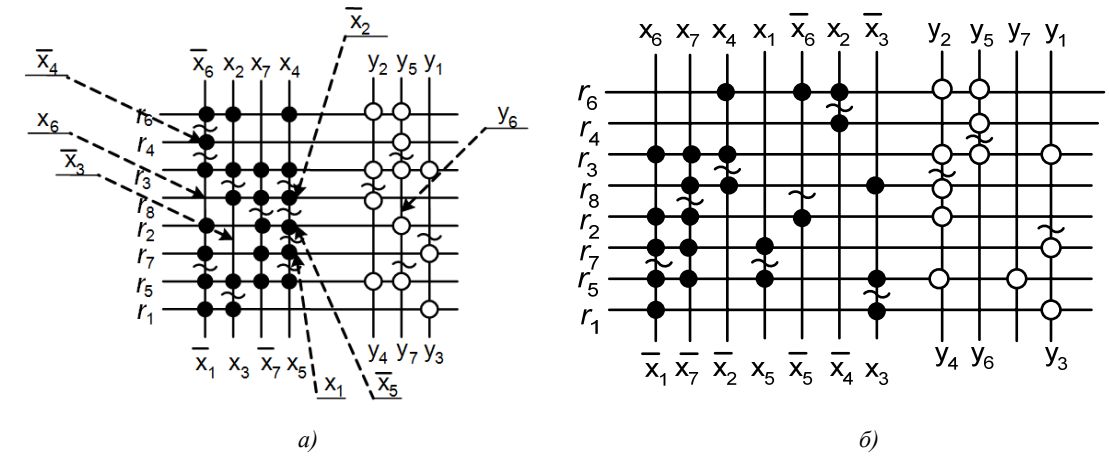
На рис. 3, *а* отображен результат ограниченной многократной столбцовой свертки, представленный множеством свертки с мощностью 7, для приведенной ранее структуры ПЛМ:

$$L^c = \{< \bar{x}_6,\ \bar{x}_4, x_6,\ \bar{x}_1>, <x_2,\ \bar{x}_3, x_3>, <x_7,\ \bar{x}_7>, <x_4,\ \bar{x}_2,\ \bar{x}_5, x_1, x_5>, <y_2, y_4>, <y_5, y_6, y_7>, <y_1, y_3>\}.$$

На этом рисунке «разрыв» шины, на которой располагаются свернутые столбцы, обозначается символом «~». Семь столбцов свернутой структуры замещает 21 столбец первоначальной структуры. На рис. 3, *б* отображен результат ограниченной простой столбцовой свертки этой же структуры ПЛМ, соответствующий множеству свертки

$$L^c = \{<x_6,\ \bar{x}_1>, <x_7,\ \bar{x}_7>, <x_4,\ \bar{x}_2>, <x_1, x_5>, <\bar{x}_6,\ \bar{x}_5>, <x_2,\ \bar{x}_4>, <\bar{x}_3, x_3>, <y_2, y_4>, <y_5, y_6>, <y_7>, <y_1, y_3>\}.$$

Итак, формальная постановка задачи столбцовой свертки состоит в следующем: дана булева матрица, соответствующая матричной структуре, необходимо найти реализуемое упорядоченное множество свертки, обеспечивающее максимальное сокращение числа столбцов.



а) *б)*

Рис. 3. Свертки ПЛМ: *а)* многократная; *б)* простая

2. Реализация многократной свертки регулярной структуры

Столбцовая свертка в соответствии с реализуемым множеством свертки L^c накладывает, как было установлено в разд. 1, ограничение на порядок строк матрицы $\boldsymbol{B}$ и сводится к перестановке ее строк. Соответственно решение задачи свертки может быть получено путем поиска такого упорядочения строк, которое позволило бы разбить множество столбцов на минимальное число неконфликтующих наборов свертки, т. е. наборов свертки, образующих реализуемое множество свертки. Бесконфликтный порядок строк является оптимальным, если он вызывает максимальную свертку столбцов или минимальное число физических вертикальных шин в структурной матрице.

Для того чтобы получить условия, которым должна удовлетворять перестановка строк, обеспечивающая максимальное по мощности реализуемое множество свертки, рассмотрим результат многократной свертки регулярной структуры (см. рис. 3, *а*). Эта свертка соответствует некоторой перестановке π на множестве строк исходной матрицы $\boldsymbol{B}$ (см. рис. 2):

$\{\pi(r_1), \pi(r_2), ..., \pi(r_u)\} \rightarrow \{r_6, r_4, r_3, r_8, r_2, r_7, r_5, r_1\}$. Перестановка π на множестве строк булевой матрицы $\boldsymbol{B}$ трансформирует ее в матрицу $\boldsymbol{B}^{\pi}$ (рис. 4), которая функционально эквивалентна матрице $\boldsymbol{B}$, но соответствует другой структурной реализации. Для заданной перестановки π строк столбцовая свертка реализуется путем расположения на каждой вертикальной шине частей совместимых (относительно перестановки π) столбцов матрицы $\boldsymbol{B}^{\pi}$.

Столбцовая свертка мощности k существует, если существует такое разбиение $C^{\pi} = \{C_1, C_2,..., C_k\}$ на множестве $C(\boldsymbol{B}^{\pi})$ столбцов, что каждое из множеств C_i состоит из взаимно совместимых столбцов. Соответственно множества C_i являются наборами свертки, а C^{π} – реализуемым множеством свертки. Для рассматриваемого примера булева матрица $\boldsymbol{B}^{\pi}$ допускает следующее разбиение C^{π} на множестве столбцов, которое удовлетворяет этим условиям и является реализуемым:

$$C^{\pi} = \{\{c_{11}, c_7, c_{12}, c_1\}, \{c_4, c_5, c_6\}, \{c_{13}, c_{14}\}, \{c_8, c_3, c_9, c_2, c_{10}\}\}.$$

Введем некоторые обозначения для формулировки критерия оценки перестановки π на множестве строк. Матрица $\boldsymbol{B}^{\pi}$ определяет граф $G = (V, E)$, вершины множества V которого соответствуют столбцам $c_i^{\pi} \in C(\boldsymbol{B}^{\pi})$ и пара вершин v_i и v_j связана ребром $e_{ij} \in E$, если соответствующие столбцы c_i^{π} и c_j^{π} несовместимы, т. е. пересекаются: $c_i^{\pi} \cap c_j^{\pi} \neq \varnothing$.

	$\bar{x}_1$	x_1	$\bar{x}_2$	x_2	$\bar{x}_3$	x_3	$\bar{x}_4$	x_4	$\bar{x}_5$	x_5	$\bar{x}_6$	x_6	$\bar{x}_7$	x_7
	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}	c_{11}	c_{12}	c_{13}	c_{14}
r_6	0	0	0	1	0	0	0	1	0	0	1	0	0	0
r_4	0	0	0	0	0	0	1	0	0	0	0	0	0	0
r_3	0	0	0	1	0	0	0	1	0	0	0	1	1	0
r_8	0	0	1	0	1	0	0	0	0	0	0	0	1	0
r_2	0	0	0	0	0	0	0	0	1	0	0	1	0	1
r_7	0	1	0	0	0	0	0	0	0	0	0	1	0	1
r_5	1	0	0	0	1	0	0	0	0	1	0	0	0	1
r_1	1	0	0	0	0	1	0	0	0	0	0	0	0	0

Рис. 4. Булева матрица $\boldsymbol{B}^{\pi}$, порождаемая перестановкой строк $\{r_6, r_4, r_3, r_8, r_2, r_7, r_5, r_1\}$

	$\bar{x}_1$	x_1	$\bar{x}_2$	x_2	$\bar{x}_3$	x_3	$\bar{x}_4$	x_4	$\bar{x}_5$	x_5	$\bar{x}_6$	x_6	$\bar{x}_7$	x_7
	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}	c_{11}	c_{12}	c_{13}	c_{14}
r_1^{π}	0	0	0	**1**	0	0	0	**1**	0	0	**1**	0	0	0
r_2^{π}	0	0	0	1	0	0	**1**	1	0	0	0	0	0	0
r_3^{π}	0	0	0	**1**	0	0	0	**1**	0	0	0	**1**	**1**	0
r_4^{π}	0	0	**1**	0	**1**	0	0	0	0	0	0	1	**1**	0
r_5^{π}	0	0	0	0	1	0	0	0	**1**	0	0	**1**	0	**1**
r_6^{π}	0	**1**	0	0	1	0	0	0	0	0	0	**1**	0	**1**
r_7^{π}	**1**	0	0	0	**1**	0	0	0	0	**1**	0	0	0	**1**
r_8^{π}	**1**	0	0	0	0	**1**	0	0	0	0	0	0	0	0

Рис. 5. Матрица интервалов $\boldsymbol{D}^{\pi}$, соответствующая матрице $\boldsymbol{B}^{\pi}$

Для данной перестановки π на множестве строк и разбиения $C^{\pi} = \{C_1, C_2,..., C_k\}$ на множестве столбцов свертка регулярной структуры реализуется назначением на каждую i-ю вертикальную шину частей $\boldsymbol{d}_j^p$ столбцов c_j из блока C_p, которые соответствуют интервалам (введенным в [5]), порождаемым перестановкой π на столбцах: $\boldsymbol{d}_j^p = [\min_i b_i^j, \max_i b_i^j,]$ (от первой сверху до последней единичной компоненты), где b_i^j – i-я компонента столбца c_j, имеющая значение 1. Таким образом, для заданной перестановки π матрица $\boldsymbol{B}^{\pi}$ порождает матрицу интервалов $\boldsymbol{D}^{\pi}$: при переходе от матрицы $\boldsymbol{B}^{\pi}$ к матрице интервалов $\boldsymbol{D}^{\pi}$ нули между $\min_i b_i^j$ и $\min_i b_i^j$ (в каждом столбце) заменяются единицами. На рис. 5 приведена матрица интервалов $\boldsymbol{D}^{\pi}$, соответствующая матрице И ПЛМ (см. рис. 1) и перестановке строк $\pi = \{r_6, r_4, r_3, r_8, r_2, r_7, r_5, r_1\}$. Здесь единицы исходной матрицы $\boldsymbol{B}$ (являющейся моделью матрицы И ПЛМ) отмечены жирным шрифтом. Матрица $\boldsymbol{D}^{\pi}$ задает следующие 14 интервалов: {[7,8]}, {[6,6]}, {[4,4]}, {[1,3]}, {[4,7]}, {[8,8]}, {[2,2]}, {[1,3]}, {[5,5]}, {[7,7]}, {[1,1]}, {[3,6]}, {[3,4]}, {[5,7]}.

При переходе от $\boldsymbol{B}^{\pi}$ к $\boldsymbol{D}^{\pi}$ в общем случае увеличиваются значения плотностей строк, оцениваемых числом единиц в них. Величина их прироста существенно зависит от перестановки π. В то же время чем меньше значения плотностей строк, тем большее число столбцов может быть свернуто. Более желателен выбор такой перестановки на множестве строк, которая в меньшей степени увеличивает плотности строк при переходе от $\boldsymbol{B}^{\pi}$ к $\boldsymbol{D}^{\pi}$.

Чтобы оценить прогнозируемый размер свертки, соответствующей перестановке π, рассмотрим отношение совместимости между интервалами: интервалы $\boldsymbol{d}_i$ и $\boldsymbol{d}_j$ называются совмес-

тимыми, если они не пересекаются: $\boldsymbol{d}_i \cap \boldsymbol{d}_j = \varnothing$. При свертке совместимые (непересекающиеся) интервалы могут быть назначены на одну и ту же вертикальную шину свернутой структуры. Зададим обратное отношение (отношение несовместимости на множестве интервалов) в виде графа $G = (V, E)$ (рис. 6), в котором вершины $v_i \in V$ соответствуют столбцам $c_i \in C(\boldsymbol{B}^\pi)$ и $(v_i, v_j) \in E$, если интервалы $\boldsymbol{d}_i$ и $\boldsymbol{d}_j$ пересекаются ($\boldsymbol{d}_i \cap \boldsymbol{d}_j \neq \varnothing$). Прогнозируемое число наборов свертки (определяющее число вертикальных шин свернутой матрицы) для перестановки π на множестве строк равно хроматическому числу графа G, а сами наборы образуются из столбцов матрицы $\boldsymbol{B}$, соответствующих одноцветным вершинам графа G.

Граф G по определению представляет собой интервальный граф, который является частным видом хордального графа [6], обладающего некоторыми замечательными свойствами, позволяющими решить задачу поиска реализуемого множеств свертки за полиномиальное время. Установлено, что хроматическое число хордального графа равно размеру его наибольшей клики. Размер наибольшей клики графа G равен максимальной из плотностей строк матрицы $\boldsymbol{D}^\pi$ (обозначаемой далее через p^π), что следует из определения графа $G = (V, E)$ отношения несовместимости на множестве интервалов. Другими словами, прогнозируемый размер свертки, соответствующий данной перестановке π, может быть найден прямо из соответствующей матрицы интервалов $\boldsymbol{D}^\pi$.

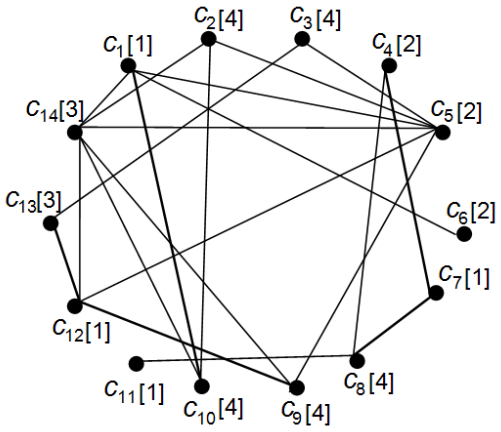


Рис. 6. Интервальный граф $G = (V, E)$

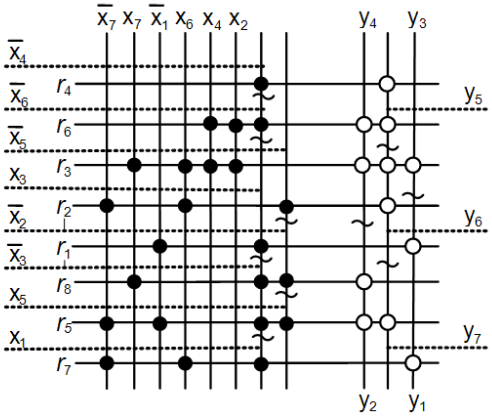


Рис. 7. Многократно свернутая ПЛМ с указанием линий подвода сигналов

Таким образом, численной оценкой качества перестановки строк π может служить максимальная плотность строк p^π. После выбора перестановки, лучшей по этому критерию, находится разбиение на множестве интервалов и соответственно разбиение на множестве столбцов матрицы $\boldsymbol{B}^\pi$ путем решения одной из задач:

1) раскраски графа $G = (V, E)$: для раскраски интервального графа, каковым является $G = (V, E)$, существует простой алгоритм раскраски линейной сложности [7];

2) упорядочения интервалов, заданных матрицей $\boldsymbol{D}^\pi$: множество интервалов может быть упорядочено с помощью алгоритма Left Edge [8] линейной сложности.

Хроматическое число интервального графа G (рис. 6) равно четырем для перестановки $\pi = \{r_6, r_4, r_3, r_8, r_2, r_7, r_5, r_1\}$ на множестве строк. Цвета вершин графа приведены в квадратных скобках. Соответственно получаются следующие четыре набора свертки:

$$<c_1, c_7, c_{11}, c_{12}>, <c_4, c_5, c_6>, <c_{13}, c_{14}>, <c_2, c_3, c_8, c_9, c_{10}>.$$

Множество свертки $L^c = \{<c_{11}, c_7, c_{12}, c_1>, <c_4, c_5, c_6>, <c_{14}, c_{13}>, <c_8, c_3, x_9, c_2, c_{10}>\}$ по построению является реализуемым, соответствующая свернутая структура изображена на рис. 3, *а*.

Таким образом, для любой перестановки π существует просто вычисляемая оценка ее эффективности: значение максимальной плотности p^π строк матрицы интервалов. После выбора наилучшей перестановки максимальное (для этой перестановки) реализуемое множество свертки находится с помощью алгоритма линейной сложности.

3. Метод моделирования отжига

Метод моделирования отжига (simulated annealing) основан на имитации физического процесса, который происходит при кристаллизации вещества из жидкого состояния в твердое. История метода восходит к пионерским работам Н. Метрополиса [9], использовавшего метод отжига для моделирования физических процессов, и С. Киркпатрика [10], впервые предложившего использовать его для решения оптимизационных задач. С тех пор моделирование отжига является широко используемым методом решения комбинаторных задач оптимизации для широкого круга приложений, где необходим поиск оптимального или близкого к нему решения. С формальной точки зрения этот метод оптимизации основан на управляемом случайном поиске. Его преимуществом является способность сбрасывать решение из локальных минимумов оптимизируемой функции с тем, чтобы продолжить поиск глобального минимума.

В общем случае задачи, для которых целесообразно применение метода отжига, могут быть сформулированы следующим образом [11, 12]: имеется конечное множество S состояний $s_i(x) = (x_1, x_2,\ldots, x_n)$ и целевая (стоимостная) функция $Cost(s_i(x))$. Требуется найти состояние $s_i(x) \in S$ с минимумом (максимумом) значения $Cost(s_i(x))$.

Подавляющее большинство подобных задач являются NP-трудными, на практике же для задач большой размерности достаточно получить решение, близкое к оптимальному. Одной из эвристик решения подобных задач за полиномиальное время является жадная эвристика, но она дает всего лишь локальный оптимум, не всегда доставляющий хорошее решение. Моделирование отжига дополняет жадный алгоритм случайным отклонением от локального минимума, преобразуя его в конечном счете в алгоритм случайного спуска. Величина случайного отклонения контролируется посредством параметра, называемого температурой. Более высокая температура делает алгоритм более чувствительным к увеличению величины целевой функции при выборе очередного состояния. Алгоритм моделирования отжига состоит из следующих шагов:

1. Начальные установки параметров задачи.
2. Изменение решения случайным образом.
3. Оценка нового решения.
4. Проверка условия принятия нового решения.
5. Уменьшение температуры.

На первом шаге процедуры моделирования отжига устанавливаются значение начальной температуры, начальное состояние процесса и его оценка (значение целевой функции). Шаги 2–5 выполняются в цикле до тех пор, пока не выполнится условие остановки, например, заданное значением температуры или числом итераций. После выбора решения $s_i(x)$ и вычисления его целевой функции $Cost(s_i(x))$ находится величина $\triangle = Cost(s_i(x)) - Cost^*$ изменения значения целевой функции по сравнению с найденным глобальным минимумом. Новое состояние $s_i(x)$ принимается с вероятностью 1, если $\triangle < 0$ (если новое состояние дает лучшее значение целевой функции, то оно будет всегда принято), или с вероятностью $P(\triangle,t) = e^{-\triangle/t}$ (вероятность принятия «худшего» состояния уменьшается экспоненциально с увеличением его стоимости и уменьшением температуры t).

Выбираемые значения температуры образуют некую убывающую, сходящуюся к нулю последовательность, она строится на основе распределений Больцмана (как, например, в [10]) или Коши, сверхбыстрого отжига [13] и др. Для задач больших размерностей, ориентированных на получение не обязательно оптимального решения, применяются последовательности с быстрым затуханием температуры, например с законом изменения температуры $t_i = c\, t_{i-1}$, где значение константы c выбирается из интервала 0,7 – 0,99.

Таким образом, схема метода моделирования отжига для конкретной оптимизационной задачи определяется:

– правилом выбора очередного состояния $s_i(x) \in S$;
– законом изменения температуры $t(i)$, где i – номер итерации;
– функцией вероятности принятия решения $P(\triangle,t)$.

4. Многократная свертка матричной структуры на основе моделирования отжига

В процессе решения задачи свертки методом моделирования отжига строки структурной матрицы $\boldsymbol{B}$ переставляются таким образом, чтобы минимизировать мощность порождаемого множества свертки. Состоянием задачи в процессе моделирования отжига является текущий порядок строк π в матрицах $\boldsymbol{B}^{\pi}$ и $\boldsymbol{D}^{\pi}$. За начальное состояние π принимается первоначально заданный порядок $\pi = \{r_1, r_2, \ldots, r_m\}$ на множестве строк. Итерацией (движением) является перестановка местами двух строк, меняющая текущее состояние на некоторое соседнее.

Температурный режим представляется геометрической прогрессией со знаменателем прогрессии 0,85 и начальным значением $t_0 = \lambda_t n$, где n – число столбцов матрицы $\boldsymbol{B}$, а значение $\lambda_t = 0{,}8$ выбрано экспериментальным путем. Таким образом, каждое последующее значение температуры получается умножением предыдущего на одну и ту же константу, меньшую единицы, и температура в пределе стремится к нулю. Число итераций при каждой температуре принимается равным $r_0 = 2\lambda_r C_m^2$, где m – число строк, $C_m^2 = m(m-1)/2$ – число всевозможных итераций, а значение $\lambda_r < 1$ принято равным 0,5 (как, например, в [12]).

Для каждого промежуточного состояния, характеризующегося перестановкой π, вычисляется значение целевой (стоимостной) функции Q^{π}. Целью отжига является нахождение матрицы интервалов $\boldsymbol{D}^{\pi}$, имеющей минимум максимальной плотности p^{π} ее строк: $\max p_i^{\pi} \to \min$.

Принимая во внимание, что перестановка строк и соответственно характеризующая ее максимальная плотность p^{π} являются случайными величинами, выбрано следующее выражение целевой функции Q^{π}:

$$Q^{\pi}(\pi) = (p^{\pi})^2 + \frac{\lambda_{\pi}}{m}\sum_i (p_i^{\pi})^2 ,$$

где значение λ принято равным 0,5 для того, чтобы уменьшить влияние второго члена $\frac{1}{m}\sum_i (p_i^{\pi})^2$ (средней плотности строк матрицы $\boldsymbol{D}^{\pi}$) на значение Q^{π} для текущего состояния π.

Итерация считается успешной (и состояние принимается в качестве текущего), если значение целевой функции $Q^{\pi}(\pi)$ для текущего состояния π меньше, чем полученное минимальное значение этой функции на предшествующих итерациях. Процедура отжига заканчивается, когда число успешных итераций при текущей температуре не превышает 5 % от числа всех итераций или температура становится достаточно низкой. Алгоритм многократной свертки методом моделирования отжига представляется следующим псевдокодом:

begin Folding

$\boldsymbol{B} \leftarrow \boldsymbol{B}_0$ — $\boldsymbol{B}_0$ – булева матрица, представляющая свертываемую структуру

$\pi \leftarrow \pi_0$ — π_0 – начальное состояние – начальный порядок строк: $\pi_0 = \{r_1, r_2, \ldots, r_m\}$

$\boldsymbol{D}_{\pi} \leftarrow \boldsymbol{D}^{\pi}{}_0$ — $\boldsymbol{D}^{\pi}{}_0$ – матрица интервалов, соответствующая перестановке строк π_0

$t \leftarrow t_0$ — t_0 – начальная температура: $t_0 = 0{,}8\,n$

$r_t \leftarrow r_0$ — r_0 – число итераций при начальной температуре $r_0 = m(m-1)/2$

$Q^{\pi} \leftarrow Q^{\pi}{}_0$ — $Q^{\pi}{}_0$ – значение целевой функции для перестановки строк π_0

while $((\delta_t \ge 0{,}05\, r_t) \vee (t \ne 0))$ **do** — условия остановки не выполняются

  $i \leftarrow 0$ — i – порядковый номер итерации при текущей температуре t

  $\delta_t \leftarrow 0$ — δ_t – число успешных итераций при текущей температуре t

  while $(i < r_t)$ **do** — для $t = t_{\text{const}}$

    $\pi^* \leftarrow$ generate(π) — генерация случайного состояния, соседнего для π

    $\boldsymbol{B}^* \leftarrow \boldsymbol{B}^{\pi}{}^*$ — построение матрицы $\boldsymbol{B}^{\pi}{}^*$ для π^*

    $\boldsymbol{D}^{\pi}{}^* \leftarrow \boldsymbol{D}^{\pi}(\pi^*)$ — построение матрицы интервалов $\boldsymbol{D}^{\pi}$ для π^*

    $Q^{\pi}{}^* \leftarrow Q^{\pi}(\pi^*)$ — оценка $\boldsymbol{D}^{\pi}{}^*$

    $\triangle \leftarrow Q^{\pi}{}^* - Q^{\pi}$

    if $((\triangle \le 0) \vee (\text{random}(0,1) < e^{-\triangle/t}))$ **then**

      $\pi \leftarrow \pi^*$; $\boldsymbol{B} \leftarrow \boldsymbol{B}^{\pi}{}^*$; $Q^{\pi} \leftarrow Q^{\pi}{}^*$;

      $\delta_t \leftarrow \delta_t + 1$ — увеличение числа успешных итераций

```
        end if
      end while
      t ← reduce temperature(t);        значение температуры умножается на 0,85
    end while                           лучшее решение найдено
end Folding
```

5. Ограниченная многократная свертка

Рассматриваемая процедура многократной свертки модифицируется в том случае, когда вводятся ограничения на вид свернутой структуры, которые можно разделить на следующие основные группы:

1. Ограничения структуры. Этот класс ограничений определяется используемым типом регулярной структуры. Например, при использовании ПЛМ необходимо принимать во внимание, что ее топология состоит из пары связанных матриц И и ИЛИ.

2. Ограничения входа/выхода. Этот класс ограничений зависит от среды, в которой свернутая структура реализована. Например, могут быть наложены ограничения на порядок и сторону подвода входных/выходных шин, группирование входных/выходных шин.

3. Электрические ограничения. Накладываются ограничения на тип свертки. Например, это может быть только простая или двухдольная [14] свертка.

4. Физические ограничения. Этот класс ограничений касается топологической реализации прямого и инверсного литералов. Например, входной сигнал в прямом и инверсном виде не может подаваться с разных сторон структуры или линии их подвода должны располагаться рядом. В связи с этим соответствующие столбцы не могут располагаться на одной шине и поэтому не могут быть свернуты.

Далее покажем, как учитываются ограничения структуры, накладываемые на свертку, продемонстрировав это на примере простой свертки и свертки ПЛМ.

6. Свертка ПЛМ структуры на основе моделирования отжига

ПЛМ состоит из двух транзисторных матриц И и ИЛИ, столбцам которых соответствуют входные и выходные переменные соответственно. В строках матрицы И реализуются элементарные конъюнкции, значения которых подаются на строки матрицы ИЛИ, в столбцах которой реализуются дизъюнкции (или их инверсии в зависимости от программирования выходов) этих конъюнкций. Свертка ПЛМ позволяет некоторые ее столбцы совмещать на одной вертикальной шине (а строкам разделять горизонтальную шину). Топология ПЛМ накладывает структурные ограничения на вид свертки, которые не позволяют реализовать на одной шине столбец матрицы И и столбец матрицы ИЛИ. Многократная и простая свертки ПЛМ-структуры показаны на рис. 3.

При формулировке процедуры моделирования отжига для случая многократной свертки ПЛМ требуется учитывать упомянутые ограничения. Для этого предлагается:

1) представлять матрицу интервалов $\boldsymbol{D}^{\pi}$ в виде двух подматриц $\boldsymbol{D}^{\pi}_{in}$ и $\boldsymbol{D}^{\pi}_{out}$, соответствующих матрицам И и ИЛИ;

2) различать входную p^{π}_{ini} и выходную p^{π}_{outi} плотности строк подматриц $\boldsymbol{D}^{\pi}_{in}$ и $\boldsymbol{D}^{\pi}_{out}$;

3) строить два интервальных графа $G^{in} = (V^{in}, E^{in})$ и $G^{out} = (V^{out}, E^{out})$ соответственно для подматриц $\boldsymbol{D}^{\pi}_{in}$ и $\boldsymbol{D}^{\pi}_{out}$.

Прогнозируемый размер свертки, порождаемой текущей перестановкой π, равен сумме хроматических чисел интервальных графов G^{in} и G^{out} или максимальной плотности p^{π}, вычисляемой как сумма максимальных плотностей p^{π}_{in} и p^{π}_{out}. Таким образом, процедура моделирования отжига для решения задачи многократной свертки с такими ограничениями отличается от сформулированной выше только способом вычисления максимальной плотности p^{π} и соответственно целевой функции Q^{π}.

Таким же образом могут быть приняты во внимание ограничения второго типа (касающиеся подведения входных и выходных сигналов). Например, на рис. 7 отображена многократ-

но свернутая ПЛМ, в которой для подведения входных сигналов к сегментам вертикальных шин выделяются отдельные горизонтальные шины. Учет ограничений, регламентирующих сторону подведения сигналов на свернутую структуру, может производиться посредством дополнения интервалов (добавления новых строк) $\boldsymbol{d}_i \in \boldsymbol{D}^{\pi}$ при переходе от $\boldsymbol{B}^{\pi}$ к $\boldsymbol{D}^{\pi}$. При этом нужно принимать во внимание увеличение длины интервалов на каждом шаге преобразования матрицы $\boldsymbol{B}^{\pi}$ в интервальную матрицу $\boldsymbol{D}^{\pi}$.

7. Простая свертка матричной структуры на основе моделирования отжига

Рассмотрим, как при свертке методом отжига можно учитывать более сложные ограничения. В частности, электрические ограничения приводят к тому, что допускается только простая свертка. Напомним, что свертка называется простой, если на шине возможен только один разрыв. В таком случае, очевидно, на одной физической шине может быть размещено только два столбца (или две строки). Простая столбцовая свертка имеет очевидное преимущество перед многократной сверткой, потому что внешние сигналы могут быть подведены с двух сторон матрицы свернутой структуры. В противном случае нужно выполнять подвод внутри структуры, что требует введения дополнительных слоев металлизации или горизонтальных линий (см. рис. 7). Это упрощает подвод сигналов и увеличивает площадь кристалла СБИС, выделенную под трассировку соединений, несмотря на то, что многократная свертка более результативна в смысле сокращения площади функциональной части регулярной структуры.

При поиске простой свертки методом моделирования отжига в работе [5] предложено использовать процедуру отжига, аналогичную процедуре для многократной свертки, но с более сложной целевой функцией Q^{π}, которая требует больших вычислительных затрат. Вычисление значения функции связано с оценкой мощности наибольшего паросочетания неориентированного графа, построенного на отношениях попарной совместимости столбцов матричной структуры. Таким образом, на каждой итерации процедуры отжига выполняется трудоемкая процедура построения графа совместимости и нахождения наибольшего паросочетания.

В статье предлагается процедура сокращения трудоемкости решения задачи простой свертки на основе метода отжига за счет однократной реализации процедуры построения графа отношений совместимости столбцов структуры и нахождения его наибольшего паросочетания. Это в некоторых случаях приводит к потере оптимальности простой свертки, но, принимая во внимание, что метод моделирования отжига сам по себе эвристический, не гарантирующий оптимального решения задачи свертки, предложенная идея является достаточно неплохой на практике при оптимизации больших регулярных структур.

Основное предположение базируется на эвристике, которая позволяет получить хорошую простую свертку на основе хорошей многократной свертки. Сначала методом отжига находится наилучшая перестановка π строк структурной матрицы для случая многократной свертки с использованием вышеизложенной процедуры. Затем на основе найденной перестановки π строится реализуемое множество многократной свертки L^c (состоящее из наборов свертки), на основе которого формируется множество простой свертки L^c_s (состоящее из пар свертки).

Итак, имеем перестановку строк π и соответствующую матрицу интервалов $\boldsymbol{D}^{\pi}$ (см. рис. 5). Пара столбцов c_i, c_j, соответствующих совместимым интервалам $\boldsymbol{d}_i$, $\boldsymbol{d}_j$, называется парой свертки и может быть назначена на одну вертикальную шину в свернутой структуре. Пусть $H = (V, E)$ – неориентированный граф, в котором вершина V соответствует интервалам $\boldsymbol{d}_i$ и $(v_i, v_j) \in E$, если и только если $\boldsymbol{d}_i$ и $\boldsymbol{d}_j$ не пересекаются. Размер простой свертки для перестановки строк π равен мощности наибольшего паросочетания графа $H = (V, E)$. Паросочетанием M в графе H называется множество попарно несмежных ребер, т. е. не существует двух ребер, инцидентных одной вершине [15].

Каждое ребро паросочетания M порождает пару свертки, состоящую из столбцов, которые соответствуют вершинам графа H, инцидентным ребру. Соответственно паросочетание M определяет множество простой свертки, а максимальное паросочетание – множество свертки максимальной мощности. Число вертикальных шин свернутой таким образом регулярной структуры равно мощности наибольшего паросочетания плюс число вершин графа H, не вклю-

ченных в это паросочетание.

Рассмотрим граф $H = (V, E)$ (рис. 8) для матрицы интервалов $\boldsymbol{D}^{\pi}$, показанной на рис. 5. На нем ребра графа, входящие в наибольшее паросочетание, изображены жирными линиями. Вид простой свертки ПЛМ для приведенного варианта паросочетания показан на рис. 3, *б*.

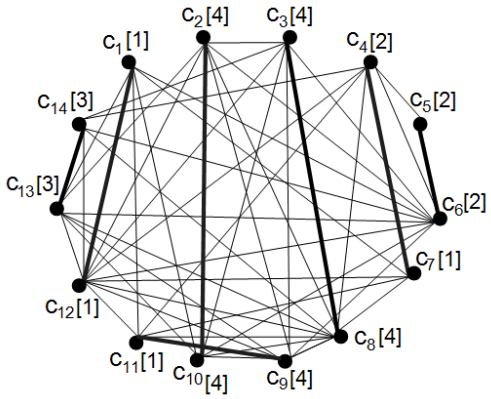


Рис. 8. Граф совместимости $H = (V, E)$

8. Результаты экспериментального исследования

Предложенные алгоритмы многократной и простой сверток были переформулированы для случая ПЛМ-структур и реализованы на языке C++. Было проведено экспериментальное исследование алгоритмов на примерах из набора benchmark [16], которое преследовало следующие цели:

1) установить степень сокращения площади ПЛМ при использовании многократной и простой сверток ПЛМ;

2) определить зависимость степени сокращения площади ПЛМ от степени разреженности ее структурных матриц И и ИЛИ;

3) провести сравнительное исследование двух типов свертки ПЛМ: многократной и простой.

Степень возможного сжатия ПЛМ-структуры существенно зависит от плотности матриц И и ИЛИ. Под плотностью матрицы понимается отношение числа активных транзисторов к числу всех транзисторов, (т. е. отношение числа единичных элементов булевой матрицы $\boldsymbol{B}$ к числу всех ее элементов), выраженное в процентах.

Каждой точке на графике (рис. 9) соответствует один пример из набора benchmark, который решался методами многократной и простой сверток. Снизу по оси X приведены названия тестовых ПЛМ, упорядоченные в соответствии с возрастанием плотности матриц И и ИЛИ (суммарные значения плотности показаны в скобках). По оси Y указана степень сжатия ПЛМ в результате свертки, выраженная в процентах. Степень сжатия измеряется отношением

$(1 - \frac{k}{n})$ %, где n и k – числа столбцов ПЛМ до и после свертки.

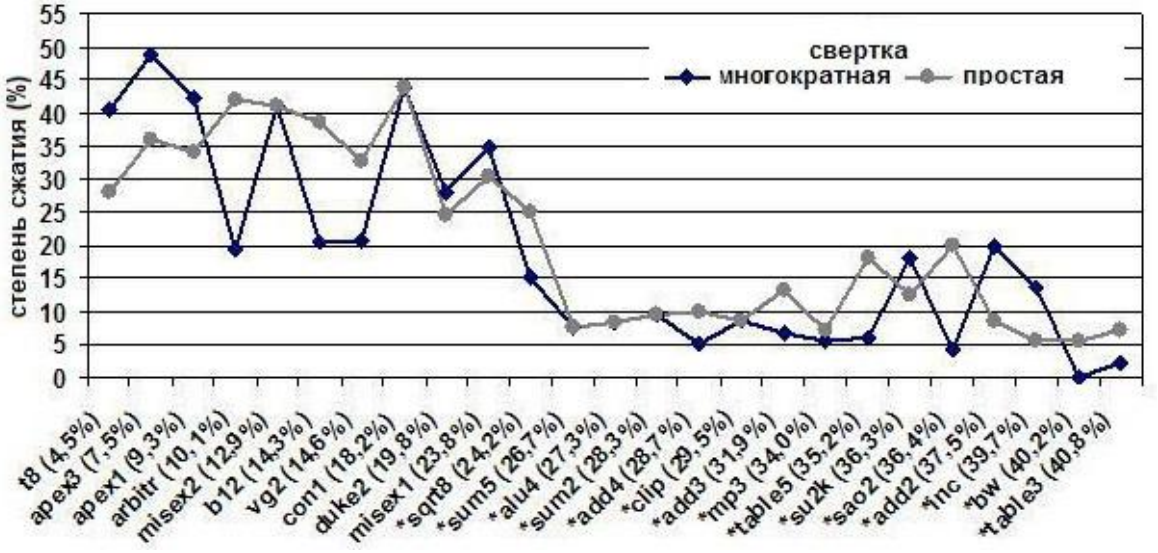


Рис. 9. Результаты экспериментальных исследований многократной и простой сверток с использованием алгоритма моделирования отжига

Результаты эксперимента позволили сделать следующие выводы, касающиеся области предпочтительного использования простой и многократной сверток:

1) многократная свертка дает лучшие результаты для случая сильно разреженных ПЛМ (менее 20 % площади занято транзисторами);

2) простая свертка дает лучшие результаты для случаев более плотных ПЛМ (более 25 % площади занято транзисторами);

3) в остальных случаях (20–25 % площади занято транзисторами) невозможно сказать, какой из методов свертки эффективнее – результаты сильно зависят от конкретных примеров.

Заключение

Исследовалась задача сокращения площади двухмерных матричных регулярных структур заказных СБИС. Рассматривались эвристические алгоритмы на основе процедуры моделирования отжига, которые дают хорошие результаты для случая уплотнения разреженных структур, имеющих небольшой процент активных элементов (транзисторов). Алгоритмы свертки на основе моделирования отжига исследовались для случаев многократной и простой столбцовой свертки ПЛМ. Результаты исследования показали:

– свертка на основе моделирования отжига может дать достаточно хорошие решения в смысле сокращения площади;

– простая свертка дает не худшие результаты по сравнению с многократной сверткой, позволяя часто находить даже лучшие решения.

Список литературы

1. Ульман, Дж. Вычислительные аспекты СБИС / Дж. Ульман. – М. : Радио и связь, 1990. – 480 с.

2. Hachtel, G.D. An Algorithm for optimal PLA Folding / G.D. Hachtel, A.R. Newton, A. L. Sangiovanni-Vincentelli // IEEE Trans. Computer-Aided Design of Integrated Circuit Syst. – 1982. – Vol. CAD-1, № 2. – P. 63–77.

3. DeMicheli, G.A. Multiple Constrained Folding of Programmable Logic Arrays: Theory and Applications / G.A. DeMicheli, Sangiovanni-Vincentelli // IEEE Trans. Computer-Aided Design. – 1983. – Vol. CAD-2, № 3. – P. 151–167.

4. Macii, E. Graph models for PLA folding problems / E. Macii , T. Wolf // International Journal of Systems Science. – 1996. – Vol. 26, № 7. – P. 1439–1445.

5. Wong, D.F. Simulated Annealing for VLSI Design / D.F. Wong, H.W. Leong, C.L. Liu // Boston : Kluwer Academic Publ. – 1988. – 220 p.

6. Hsu, W.-L. Fast and simple algorithms for recognizing chordal comparability graphs and interval graphs / W.-L. Hsu, T.-H. Ma // SIAM J. Comput. – 1999. – Vol. 28, № 3. – P. 1004–1020.

7. Golumbic, M.C. The complexity of comparability graph recognition and coloring / M.C. Golumbic // Computing. – 1977. – Vol. 18. – P. 199–208.

8. Hashimoto, A. Wire Routing by Optimizing Channel Assignment Within Larger Apertures / A. Hashimoto, J. Stevens // Proc. of 8th Design Automation Workshop, DAC '71. – Atlantic City, NJ, USA. – 1971.

9. Equation of State Calculations by Fast Computer Machines / N. Metropolis [et al.] // J. Chemical Physics. – 1953. – Vol. 21, № 6. – P. 1087–1092.

10. Kirkpatrick, S. Optimization by Simulated Annealing / S. Kirkpatrick, Jr.C.D. Gelatt, M.P. Vecchi // Science. – 1983. – Vol. 220(4598). – P. 671–680.

11. Greening, D.R. Simulated Annealing with Errors / D.R. Greening // Ph.D. dissertation, University OF CA Los Angeles. – 1995 [Electronic resourse]. – Mode of access : http: // dan.greening.org/publications/phd.pdf. – Date of access : 28.03.2004.

12. Van Laarhoven, P.J.M. Simulated Annealing: Theory and Applications / P.J.M. Van Laarhoven, E.H.L. Aarts. – Dordrecht : Reidel, 1987.

13. Ingber, L. Optimization of Trading Physics Models of Markets / L. Ingber, R.P. Mondescu // IEEE Trans. Neural Networks. – 2001. – № 12(4). – P. 776–790.

14. Liu, Chun-Yeh. An Efficient Algorithm for Bipartite PLA Folding / Chun-Yeh Liu, Kewal K. Saluja // IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems. – 1993. – Vol. 12, №. 12. – P. 1839–1847.

15. Лекции по теории графов / В.А. Емеличев [и др.]. – М. : Наука, 1990. – 384 с.

16. Berkeley PLA test set [Electronic resource]. – Mode of access : http:// www1.cs.columbia.edu/ ~cs4861/sis/ espresso-examples/. – Date of access : 03.05.2006.

Поступила 19.06.12

Объединенный институт проблем
информатики НАН Беларуси,
Минск, Сурганова, 6
e-mail: cld@newman.bas-net.by
log@newman.bas-net.by

L.D. Cheremisinova, I.P. Loginova

FOLDING OF CUSTOM VLSI REGULAR STRUCTURES BY SIMULATED ANNEALING

The problem of topological optimization of programmable logic arrays using folding technique is studied. The algorithms of multiple and simple folding of VLSI regular structures are presented, which are based on the method of simulated annealing and allow to find optimal or near optimal solutions of the folding problem. Results of computer experiments with the suggested algorithms are given.

УДК 519.8

Я.М. Шафранский[1], Д.С. Следнев[2]

МИНИМИЗАЦИЯ МАКСИМАЛЬНОГО ВРЕМЕННОГО СМЕЩЕНИЯ ДЛЯ ОДНОГО ПРИБОРА В УСЛОВИЯХ НЕОПРЕДЕЛЕННОСТИ ДИРЕКТИВНЫХ СРОКОВ

Рассматривается задача минимизации максимального временного смещения в условиях неопределенности директивных сроков при наличии ограничений предшествования и обслуживании требований одним прибором $1|prec; d_j \in [d_j^{\min}, d_j^{\max}]|L_{\max}$. *Формулируются необходимые и достаточные условия оптимальности расписания в детерминированном случае, а также необходимые и достаточные условия глобальной оптимальности расписания в случае с неопределенными директивными сроками, предлагается алгоритм построения глобально оптимального расписания.*

Введение

Рассматривается задача построения оптимального расписания работы обслуживающей системы, состоящей из одного прибора, который должен обслужить n требований. Прибор может обслуживать не более одного требования одновременно. Прерывания в обслуживании требования не допускаются. Все требования поступают в обслуживающую систему одновременно. Длительность обслуживания требования j составляет $p_j \geq 0$ единиц времени, $j = 1,\ldots, n$. Каждому требованию j поставлен в соответствие директивный срок $d_j \geq 0$ завершения его обслуживания. На множестве требований задано отношение предшествования, представленное ориентированным бесконтурным графом $G = (V, E)$. Если требование j_1 предшествует требованию j_2, т. е. $(j_1, j_2) \in E$, то обслуживание требования j_2 не может быть начато до завершения обслуживания j_1.

Расписание в такой системе может быть представлено в виде допустимой относительно заданных ограничений предшествования перестановки $\pi = (\pi(1),\ldots, \pi(n))$, определяющей последовательность обслуживания требований. Здесь $\pi(k)$ – требование, расположенное в перестановке π на позиции k.

Обозначим через $C_j(\pi) = p_{\pi(1)} + \ldots + p_{\pi(k)}$ момент завершения обслуживания требования j, расположенного в перестановке π на месте k. Тогда $L_j(\pi) = C_j(\pi) - d_j$ – временное смещение момента завершения обслуживания требования j относительно его директивного срока.

Необходимо найти допустимую перестановку π, для которой максимальное временное смещение

$$L_{\max}(\pi) = \max\{L_1(\pi),\ldots, L_n(\pi)\}$$

является наименьшим. Такая перестановка называется *оптимальной*.

Сформулированная задача рассматривается в условиях неопределенности, когда директивные сроки требований не заданы, а для каждого требования j известны лишь нижняя $d_j^{\min}$ и верхняя $d_j^{\max}$ границы возможных значений величины d_j.

Одним из примеров появления неопределенных директивных сроков является ситуация, когда приходится планировать сроки производства новой продукции, не зная точного момента наступления максимального спроса на нее, а имея лишь прогнозные оценки интервала, в котором лежит эта величина.

Следуя принятой в теории расписаний системе обозначений, представим детерминированный вариант задачи как $1|prec|L_{\max}$, а вариант в условиях неопределенности обозначим через $1|prec; d_j \in [d_j^{\min}, d_j^{\max}]|L_{\max}$.

Если в графе G существует путь из вершины v в вершину u, то v называется предшественником u, а u – потомком v. Обозначим через $A_G(v)$ и $B_G(v)$ соответственно множество всех потомков и предшественников вершины v в графе G.

1. Детерминированный вариант задачи

В детерминированном случае для решения задачи $1|prec|L_{\max}$ можно воспользоваться известным алгоритмом Лоулера [1].

***Алгоритм* 1**

1. Полагаем $k = n$.
2. Находим множество T висячих вершин графа $G = (V, E)$, т. е. таких вершин v, что $A_G(v) = \emptyset$.
3. Находим в T такую вершину u, что $d_u = \max\{d_v | v \in T\}$.
4. Полагаем $\pi(k) = u$, $k = k - 1$ и модифицируем граф G, полагая $V = V \setminus \{u\}$.
5. Если $k > 0$, переходим к выполнению шага 2. Если $k = 0$, то оптимальная перестановка построена и алгоритм завершает работу.

Если граф представлен в виде матрицы смежности, то временная сложность алгоритма составляет $O(n^2)$. Если для каждой вершины заданы список входящих дуг и полустепень исхода вершины, алгоритм можно реализовать таким образом, что его временная сложность составит $O(\max\{m, n \log n\})$, где $m = |E|$.

В работе [2] установлены необходимые и достаточные условия оптимальности расписания для задачи $1||L_{\max}$, т. е. для детерминированного варианта рассматриваемой задачи при отсутствии ограничений предшествования. Приведенная ниже теорема усиливает этот результат, описывая необходимые и достаточные условия оптимальности расписания для задачи $1|prec|L_{\max}$. Предварительно введем понятие локального улучшения.

Будем говорить, что требование $\pi(k)$ имеет локальное улучшение в перестановке π, если существует такой индекс $i < k$, что $p_{\pi(i)} > 0$, $\pi(i) \notin B_G(\pi(k))$ и $d_{\pi(i)} > d_{\pi(k)}$, причем $\pi(i) \notin B_G(\pi(j))$ для тех $j \le k$, что $d_{\pi(j)} \le d_{\pi(k)}$.

Из определения локального улучшения следует, что в перестановке π между требованиями $\pi(i)$ и $\pi(k)$ могут находиться потомки требования $\pi(i)$, но для любого такого потомка $\pi(j)$ должно выполняться соотношение $d_{\pi(j)} > d_{\pi(k)}$. Переместим требование $\pi(i)$ и всех его потомков, находящихся между $\pi(i)$ и $\pi(k)$ в π, на позиции, расположенные непосредственно после $\pi(k)$. Нетрудно заметить, что в полученной допустимой перестановке π' выполняется соотношение $L_{\pi(k)}(\pi) < L_{\pi(k)}(\pi')$; для всех требований $\pi(j)$, перемещенных вправо, выполняется $L_{\pi(j)}(\pi) < L_{\pi(k)}(\pi')$, а для всех остальных требований $\pi(l)$ справедливо $L_{\pi(l)}(\pi) \le L_{\pi(l)}(\pi')$.

Теорема 1. *Допустимая перестановка π является оптимальной для задачи $1|prec|L_{\max}$ тогда и только тогда, когда существует такой индекс k, что $L_{\max}(\pi) = L_{\pi(k)}(\pi)$ и требование $\pi(k)$ не имеет локальных улучшений в π.*

Доказательство. Необходимость. Будем называть требование $\pi(k)$ *критическим*, если $L_{\max}(\pi) = L_{\pi(k)}(\pi)$.

Пусть π – оптимальная перестановка. Докажем необходимость, используя индукцию по количеству критических требований.

Пусть $\pi(k)$ – единственное критическое требование. Если условие теоремы не выполнено, то $\pi(k)$ имеет локальное улучшение в π, т. е. существует такое $i < k$, что $d_{\pi(i)} > d_{\pi(k)}$, причем $p_{\pi(i)} > 0$ и $\pi(i) \notin B_G(\pi(j))$ для тех $j = 1,\ldots, k$, что $d_{\pi(j)} \le d_{\pi(k)}$. Пусть $A_G(\pi(i)) \cap \{\pi(i+1),\ldots, \pi(k-1)\} = \{\pi(i_1),\ldots, \pi(i_a)\}$. Перестановка π имеет вид $\pi = (\ldots, \pi(i),\ldots, \pi(i_1),\ldots, \pi(i_2),\ldots, \pi(i_a),\ldots, \pi(k),\ldots)$. Из определения требования $\pi(i)$ следует, что $d_{\pi(j)} > d_{\pi(k)}$ для любого $j \in \{i_1,\ldots, i_a\}$. Рассмотрим допустимую перестановку π', которая получена из π путем переноса требований $\pi(i), \pi(i_1),\ldots, \pi(i_a)$ на позиции, следующие непосредственно за позицией требования $\pi(k)$, т. е. $\pi' = (\ldots, \pi(k), \pi(i), \pi(i_1), \pi(i_2),\ldots, \pi(i_a),\ldots)$. Требование $\pi(i_a)$ расположено на позиции k в перестановке π', поэтому $C_{\pi(i_a)}(\pi') = C_{\pi(k)}(\pi)$. Из $d_{\pi(i_a)} > d_{\pi(k)}$ следует $L_{\pi(i_a)}(\pi') < L_{\pi(k)}(\pi)$. Кроме то-

го, $L_{\pi(i_{a-1})}(\pi') < L_{\pi(k)}(\pi),..., L_{\pi(i_1)}(\pi') < L_{\pi(k)}(\pi)$, $L_{\pi(i)}(\pi') < L_{\pi(k)}(\pi)$. Из $p_{\pi(i)} > 0$ следует $L_{\pi(k)}(\pi') < L_{\pi(k)}(\pi)$, а временные смещения всех остальных требований не увеличились. Таким образом, $L_{\max}(\pi') < L_{\max}(\pi)$, что противоречит оптимальности π.

Пусть условия теоремы являются необходимыми для оптимальности всех перестановок, содержащих менее $l \geq 2$ критических требований.

Рассмотрим случай с l критическими требованиями. Если условие теоремы не выполнено ни для одного критического требования, то для самого первого (левого) критического требования $\pi(k)$ существует такое $i < k$, что $d_{\pi(i)} > d_{\pi(k)}$, $p_{\pi(i)} > 0$ и $\pi(i) \notin B_G(\pi(j))$ при $d_{\pi(j)} \leq d_{\pi(k)}$, $j = 1,\ldots, k$. Действуя аналогично случаю с единственным критическим требованием, можно переместить требование $\pi(i)$ и его потомков, расположенных на позициях с номерами, меньшими k, на позиции справа от требования $\pi(k)$. В результате получим допустимую перестановку π', в которой требование $\pi(k)$ уже не является критическим, а общее число критических требований равно $l - 1$. Все эти требования будут расположены на тех же позициях, на которых они находились в перестановке π. Поскольку в перестановке π для них условия теоремы не выполнялись, то и в перестановке π' условия теоремы для них выполняться не будут, что противоречит индуктивному предположению.

Достаточность. Пусть для допустимой перестановки π выполняются условия теоремы. Из $L_{\pi(k)}(\pi) = L_{\max}(\pi)$ имеем $L_{\pi(k)}(\pi) = C_{\pi(k)}(\pi) - d_{\pi(k)} \geq L_{\pi(i)}(\pi) = C_{\pi(i)}(\pi) - d_{\pi(i)}$, $i > k$, следовательно, $d_{\pi(k)} \leq d_{\pi(i)}$, $i > k$, причем $d_{\pi(k)} = d_{\pi(i)}$ возможно только при $p_{\pi(i)} = 0$ и $p_{\pi(l)} = 0$ для всех $k < l \leq i$. Положим $D(\pi(k)) = \{\pi(l) | \, l > k, p_{\pi(l)} > 0\}$.

Рассмотрим оптимальную перестановку π', полученную в соответствии с алгоритмом 1.

Пусть k' – позиция, на которой в π' расположено требование $\pi(k)$. Из описания алгоритма 1 и допустимости перестановки π следует, во-первых, что все требования из множества $D(\pi(k))$ расположены в π' на позициях справа от k', и, во-вторых, что справа от k' нет таких требований $\pi(l)$, что $p_{\pi(l)} > 0$ и $d_{\pi(l)} < d_{\pi(k)}$ (исходя из описания алгоритма 1, последнее было бы возможно только в случае $\pi(l) \in A_G(\pi(k))$, но это противоречило бы допустимости перестановки π).

Если в π' справа от k' нет других требований с ненулевой длительностью обслуживания, то $C_{\pi(k)}(\pi') \geq C_{\pi(k)}(\pi)$ и $L_{\pi(k)}(\pi') \geq L_{\pi(k)}(\pi)$. Следовательно, π – оптимальная перестановка.

Предположим, что в π' справа от k' есть хотя бы одно требование $\pi(l)$, $l < k$, $p_{\pi(l)} > 0$ и $d_{\pi(l)} \geq d_{\pi(k)}$. Очевидно, $C_{\pi(l)}(\pi') \geq C_{\pi(k)}(\pi)$. Если $d_{\pi(l)} = d_{\pi(k)}$, то $L_{\pi(l)}(\pi') \geq L_{\pi(k)}(\pi)$. Предположим, что $d_{\pi(l)} > d_{\pi(k)}$. Тогда из условия теоремы следует существование такого $j < k$, что $\pi(l) \in B_G(\pi(j))$ и $d_{\pi(j)} \leq d_{\pi(k)}$. Из допустимости π' следует, что требование $\pi(j)$ расположено в π' справа от $\pi(l)$. Отсюда получаем $L_{\pi(j)}(\pi') \geq L_{\pi(k)}(\pi)$.

Итак, в любом случае $L_{\max}(\pi') \geq L_{\max}(\pi)$, т. е. π – оптимальная перестановка. ■

Полученные необходимые и достаточные условия оптимальности расписания для детерминированного варианта задачи используются далее для построения необходимых и достаточных условий глобальной оптимальности расписания для задачи в условиях неопределенности.

2. Задача в условиях неопределенности

В задаче $1|prec; d_j \in [d_j^{\min}, d_j^{\max}]|L_{\max}$ в условиях неопределенности директивные сроки завершения обслуживания требований не заданы, известны лишь интервалы их возможных значений: $d_j \in [d_j^{\min}, d_j^{\max}]$, и директивный срок d_j может принимать любое значение из указанного интервала.

Перестановка называется *глобально оптимальной* для задачи $1|prec; d_j \in [d_j^{\min}, d_j^{\max}]|L_{\max}$, если при любых допустимых значениях неопределенных параметров d_j (т. е. при любых $d_j \in [d_j^{\min}, d_j^{\max}]$, $j = 1,..., n$) эта перестановка является оптимальной для соответствующей детерминированной задачи.

Пусть дана перестановка $\pi = (\pi(1),..., \pi(n))$. Для $j = 1,..., n$ положим $L_j^{\min}(\pi) = C_j(\pi) - d_j^{\max}$, $L_j^{\max}(\pi) = C_j(\pi) - d_j^{\min}$. При любом выборе неопределенных параметров величина $L_j(\pi)$ для требования j принимает значения из интервала $[L_j^{\min}(\pi), L_j^{\max}(\pi)]$.

Для допустимой перестановки π введем множества A_1 и A_2 индексов требований:

$$A_1 = \{1 \le k \le n \mid L_{\pi(k)}^{\max}(\pi) > \max_{1 \le j \le n} L_{\pi(j)}^{\min}(\pi)\}, \varnothing;$$

$$A_2 = \begin{cases} \{1 \le k \le n \mid L_{\pi(k)}^{\min}(\pi) = L_{\pi(k)}^{\max}(\pi) = \max_{1 \le j \le n} L_{\pi(j)}^{\min}(\pi) > \max_{j \in A_1} L_{\pi(j)}^{\min}(\pi)\} \text{ при } A_1 \ne \varnothing; \\ \{1 \le k \le n \mid L_{\pi(k)}^{\min}(\pi) = L_{\pi(k)}^{\max}(\pi) = \max_{1 \le j \le n} L_{\pi(j)}^{\min}(\pi)\} \text{ при } A_1 = \varnothing. \end{cases}$$

Ниже схематично представлено взаимное расположение интервалов $[L_j^{\min}(\pi), L_j^{\max}(\pi)]$, определяющих состав множеств A_1 и A_2:

(а) L_j: [ ]
L_i: [ ]
L_l: |

(б) L_j: [ ]
L_i: [ ]
L_l: |
L_r: |

В случае (а) $A_1=\{j, i\}$, $A_2=\{l\}$, где l – детерминированное требование (т. е. $d_l^{\max} = d_l^{\min}$). В случае (б) $A_1=\varnothing$ и $A_2=\{l, r\}$, где l и r – детерминированные требования. Множество A_2, если оно не пусто, состоит только из детерминированных требований.

По аналогии с детерминированным случаем введем понятие требования, не имеющего локальных улучшений в перестановке.

Будем говорить, что требование $\pi(k)$ не имеет локальных улучшений в перестановке π, если для всех тех $i < k$, что $p_{\pi(i)} > 0$ и $\pi(i) \notin B_G(\pi(k))$, либо выполняется неравенство $d_{\pi(i)}^{\max} \le d_{\pi(k)}^{\min}$, либо существует такой индекс $j < k$, что $\pi(i) \in B_G(\pi(j))$ и $d_{\pi(j)}^{\max} \le d_{\pi(k)}^{\min}$.

Следующая теорема описывает необходимые и достаточные условия глобальной оптимальности перестановки.

Теорема 2. *Допустимая для задачи* $1|prec; d_j \in [d_j^{\min}, d_j^{\max}]|L_{\max}$ *перестановка* π *является глобально оптимальной тогда и только тогда, когда ни для одного индекса* $k \in A_1$ *требование* $\pi(k)$ *не имеет локальных улучшений в перестановке* π, *и если* $A_2 \ne \varnothing$, *то существует такой индекс* $l \in A_2$, *что требование* $\pi(l)$ *не имеет локальных улучшений в перестановке* π.

Доказательство. Необходимость. Пусть $|A_1| = a > 0$ и $A_1 = \{i_1,\ldots, i_a\}$. Поскольку π является глобально оптимальной перестановкой, то при любом выборе $d_j \in [d_j^{\min}, d_j^{\max}]$, $j = 1,\ldots, n$, перестановка π должна оставаться оптимальной и для нее должны выполняться условия теоремы 1. В частности, условия теоремы 1 должны выполняться для $d_{\pi(i)} = d_{\pi(i)}^{\max}$, $i \ne i_1$, и $d_{\pi(i_1)} = d_{\pi(i_1)}^{\min}$. При таком выборе имеем $L_{\pi(i)} = L_{\pi(i)}^{\min}$, $i \ne i_1$, и $L_{\pi(i_1)} = L_{\pi(i_1)}^{\max}$. Поскольку $i_1 \in A_1$, то $L_{\pi(i_1)}^{\max} > L_{\pi(i)}^{\min}$, $i = 1,\ldots, n$. Следовательно, $L_{\max} = L_{\pi(i_1)} > L_{\pi(i)}$, $i \ne i_1$, и по теореме 1 имеем $d_{\pi(i)}^{\max} \le d_{\pi(i_1)}^{\min}$ для всех тех $i < i_1$, что $p_{\pi(i)} > 0$, и $\pi(i) \notin B_G(\pi(j))$ для тех $\pi(j)$, что $d_{\pi(j)}^{\max} \le d_{\pi(i_1)}^{\min}$ и $j \le i_1$. Повторив аналогичные рассуждения для $i_2,\ldots, i_a$, получим требуемое.

Если при $A_1 \ne \varnothing$ выполняется $|A_2| = b > 0$ и $A_2 = \{j_1,\ldots, j_b\}$, то положим $d_{\pi(j)} = d_{\pi(j)}^{\min}$ для всех $j \in A_2$ и $d_{\pi(i)} = d_{\pi(i)}^{\max}$ для всех $i \notin A_2$. Поскольку $A_1 \ne \varnothing$, при таком выборе имеем $L_{\max} = L_{\pi(j_1)} = \ldots = L_{\pi(j_b)} > L_{\pi(i)}$ для всех $i \notin A_2$. По условию теоремы 1 должен существовать

индекс $l \in A_2$, удовлетворяющий условию $d^{\max}_{\pi(i)} \le d^{\min}_{\pi(l)}$ для всех тех $i < l$, что $p_{\pi(i)} > 0$ и $\pi(i) \notin$ $\notin B_G(\pi(j))$ для тех $\pi(j)$, что $d^{\max}_{\pi(j)} \le d^{\min}_{\pi(l)}$ и $j \le l$.

Пусть $A_1 = \varnothing$. Из определения множеств A_1 и A_2 следует, что $A_2 = \{1 \le k \le n | L^{\min}_{\pi(k)}(\pi) = L^{\max}_{\pi(k)}(\pi) = \max\limits_{1\le j\le n} L^{\min}_{\pi(j)}(\pi)\}$. Соотношение $L_{\max} = L_{\pi(j_1)} = ... = L_{\pi(j_b)} > L_{\pi(i)}$ для всех $i \notin A_2$ обеспечивается условием $A_1 = \varnothing$, поскольку множество A_1 может содержать индексы только недетерминированных требований и $A_1 = \varnothing$ означает, что $L_{\pi(i)} < L_{\pi(j)}$ для всех $i \notin A_2$ и $j \in A_2$. Поэтому существование индекса l можно доказать так же, как это сделано в предыдущем абзаце. Необходимость доказана.

Достаточность. Из определения множеств A_1 и A_2 следует, что при любом выборе значений неопределенных параметров значение $L_{\max}(\pi)$ достигается, по крайней мере, на одном из элементов множества $A_1 \cup A_2$. Если для перестановки π выполняются условия теоремы и значение $L_{\max}(\pi)$ достигается на одном из элементов множества A_1, то для такого элемента $k \in A_1$ выполнены условия теоремы 1, т. е. π – оптимальная перестановка. Если значение $L_{\max}(\pi)$ достигается на одном из элементов множества A_2, то из определения A_2 следует, что $L_{\max}(\pi)$ достигается на каждом из элементов множества A_2. Из условия теоремы в этом случае следует существование индекса $l \in A_2$, удовлетворяющего условиям теоремы 1. Итак, при любом выборе значений неопределенных параметров перестановка π удовлетворяет условиям теоремы 1. ■

В случае, когда множество $A_1 \cup A_2$ из условия теоремы 2 содержит ровно одно требование, теорема принимает следующий вид.

Следствие 1. *Допустимая перестановка π является глобально оптимальной для задачи $1|prec; d_j \in [d^{\min}_j, d^{\max}_j]|L_{\max}$, если существует такой индекс k, что $L^{\min}_{\pi(k)}(\pi) \ge L^{\max}_{\pi(i)}(\pi)$, $i \ne k$, и требование $\pi(k)$ не имеет локальных улучшений в перестановке π.*

Следующий алгоритм строит перестановку, которая может удовлетворять условиям теоремы 2 или следствия 1.

Алгоритм 2

1. Полагаем $k = n$.
2. Находим множество T висячих вершин графа $G = (V, E)$.
3. Находим в T такую вершину u, что $d^{\min}_u = \max\{d^{\min}_v | v \in T\}$. Если имеется несколько вершин, удовлетворяющих этим условиям, то в качестве u выбираем среди них вершину с максимальным значением величины $d^{\max}_v$.
4. Полагаем $\pi(k) = u$, $k = k - 1$ и модифицируем граф G, полагая $V = V \setminus \{u\}$.
5. Если $k > 0$, переходим к выполнению шага 2. Если $k = 0$, построение перестановки π завершено.
6. Проверяем для перестановки π выполнение условий следствия 1. Если условия выполнены, то глобально оптимальная перестановка построена. В противном случае проверяем для π выполнение условий теоремы 2. Если условия выполнены, то глобально оптимальная перестановка построена, в противном случае глобально оптимальную перестановку построить не удалось.

Обоснованность попытки построить глобально оптимальную перестановку с помощью алгоритма 2 вытекает из следующей теоремы.

Теорема 3. *Для задачи $1|prec; d_j \in [d^{\min}_j, d^{\max}_j]|L_{\max}$ глобально оптимальная перестановка, удовлетворяющая условиям следствия 1, существует тогда и только тогда, когда перестановка, построенная в соответствии с алгоритмом 2, удовлетворяет условиям следствия 1.*

Доказательство. Достаточность очевидна. Покажем, что из существования глобально оптимальной перестановки π, удовлетворяющей условиям следствия 1, следует, что перестановка π^0, построенная в соответствии с алгоритмом 2, также удовлетворяет условиям следствия 1. Индекс k из условия следствия 1 будем называть критическим. Если в π имеется несколько критических индексов, то в качестве k будем рассматривать наименьший из них.

Найдем в π такой индекс $r \le n$, что $\pi(r) \ne \pi^0(r)$ и $\pi(r+1) = \pi^0(r+1), \ldots, \pi(n) = \pi^0(n)$. Пусть $\pi^0(r) = \pi(l)$, $l < r$. Перейдем от перестановки π к допустимой перестановке π', перемещая требование $\pi(l)$ с позиции l на позицию r и сдвигая требования $\pi(l+1), \ldots, \pi(r)$ на одну позицию влево.

Рассмотрим три случая: $l > k$, $l = k$ и $l < k$.

Случай $l > k$. Из описания алгоритма 2 следует, что $d^{\min}_{\pi(l)} \ge d^{\min}_{\pi(j)}$ для всех $j \le r$. Тогда $L^{\max}_{\pi(l)}(\pi') = C_{\pi(r)}(\pi) - d^{\min}_{\pi(l)} \le L^{\max}_{\pi(r)}(\pi)$. Для всех $j \ne k$ имеем $L^{\max}_{\pi(j)}(\pi') \le L^{\max}_{\pi(j)}(\pi)$, а $L^{\min}_{\pi(k)}(\pi') = L^{\min}_{\pi(k)}(\pi)$. Следовательно, перестановка π' также удовлетворяет условиям следствия 1.

Случай $l = k$. Из условия $L^{\min}_{\pi(k)}(\pi) \ge L^{\max}_{\pi(i)}(\pi)$, $i \ne k$, следствия 1 имеем $d^{\max}_{\pi(k)} \le d^{\min}_{\pi(j)}$, $j = k+1, \ldots, n$. Из определения $\pi^0(r)$ следует $d^{\min}_{\pi^0(r)} = d^{\min}_{\pi(k)} \ge d^{\min}_{\pi(j)}$, $j = 1, \ldots, r$. Поэтому $d^{\min}_{\pi(k)} = d^{\max}_{\pi(k)} = d^{\min}_{\pi(j)}$, $j = k+1, \ldots, r$. Кроме того, из определения $\pi^0(r)$ и перестановки π' следует $L^{\min}_{\pi(k)}(\pi) \le L^{\min}_{\pi(k)}(\pi') \le L^{\max}_{\pi(k)}(\pi') \le L^{\max}_{\pi(r)}(\pi)$. Отсюда с учетом $L^{\min}_{\pi(k)}(\pi) \ge L^{\max}_{\pi(j)}(\pi)$, $j \ne k$, получаем $L^{\min}_{\pi(k)}(\pi) = L^{\min}_{\pi(k)}(\pi') = L^{\max}_{\pi(k)}(\pi') = L^{\max}_{\pi(r)}(\pi)$. Последнее возможно, только если $p_{\pi(k+1)} = \ldots = p_{\pi(r)} = 0$. Отсюда следует, что перестановка π' удовлетворяет условиям следствия 1.

Случай $l < k$. Пусть $p_{\pi(l)} > 0$. Из условий следствия 1 имеем $d^{\min}_{\pi(l)} \le d^{\max}_{\pi(l)} \le d^{\min}_{\pi(k)}$, а из определения требования $\pi^0(r) = \pi(l)$ следует $d^{\min}_{\pi(l)} \ge d^{\min}_{\pi(k)}$, т. е. $d^{\min}_{\pi(l)} = d^{\max}_{\pi(l)} = d^{\min}_{\pi(k)}$. Поскольку $d^{\min}_{\pi(l)} = d^{\min}_{\pi(k)}$, то из определения требования $\pi^0(r) = \pi(l)$ следует $d^{\max}_{\pi(l)} \ge d^{\max}_{\pi(k)}$, что влечет $d^{\min}_{\pi(l)} = d^{\max}_{\pi(k)}$. Итак, $L^{\max}_{\pi(l)}(\pi') \le L^{\max}_{\pi(r)}(\pi) \le L^{\min}_{\pi(k)}(\pi) = L^{\max}_{\pi(k)}(\pi)$, а $L^{\max}_{\pi(l)}(\pi') \ge C_{\pi(k)}(\pi) - d^{\min}_{\pi(l)} = L^{\max}_{\pi(k)}(\pi)$, т. е. $L^{\max}_{\pi(l)}(\pi') = L^{\min}_{\pi(l)}(\pi') = L^{\max}_{\pi(k)}(\pi) = L^{\min}_{\pi(k)}(\pi)$. Отсюда следует, что $p_{\pi(k+1)} = \ldots = p_{\pi(r)} = 0$ и перестановка π' удовлетворяет условиям следствия 1 (единственным критическим индексом в π' является r). При $p_{\pi(l)} = 0$ очевидно, что перестановка π' удовлетворяет условиям следствия 1.

Выше везде предполагается, что $k < r$. При $k \ge r$ очевидно, что перестановка π' удовлетворяет условиям следствия 1.

Таким образом, в любом из рассмотренных случаев перестановка π' удовлетворяет условиям следствия 1 и в π' число последних элементов, совпадающих с последними элементами перестановки π^0, по крайней мере, на единицу больше, чем в π. ■

Таким образом, алгоритм 2 обеспечивает построение глобально оптимальной перестановки, удовлетворяющей условиям следствия 1, если она существует. Если такая перестановка не существует, алгоритм 2 может тем не менее построить глобально оптимальную перестановку. В последнем случае она не будет удовлетворять условиям следствия 1, но будет удовлетворять условиям теоремы 2. Алгоритм 2 обладает еще одним важным свойством, отражающим связь полученных результатов с известным критерием Вальда [3].

Следствие 2. *Алгоритм 2 реализует принцип гарантированного результата* (*строит вариант, оптимальный относительно критерия Вальда*).

Справедливость последнего утверждения следует из того, что в алгоритме 2 (являющемся, как нетрудно заметить, модификацией алгоритма Лоулера) в качестве директивных сроков требований используются величины $d^{\min}_j$, т. е. минимум целевой функции отыскивается при наихудшем варианте развития событий в условиях неопределенности, когда в качестве значений директивных сроков берутся их наименьшие возможные значения.

Заключение

Для задачи $1|\text{prec}; d_j \in [d^{\min}_j, d^{\max}_j]|L_{\max}$ минимизации максимального временного смещения в условиях неопределенности директивных сроков найдены необходимые и достаточные условия глобальной оптимальности расписания. Предложен алгоритм (алгоритм 2) построения

глобально оптимального расписания и показано, что в частном случае, когда $|A_1 \cup A_2|=1$, глобально оптимальное расписание существует тогда и только тогда, когда оно может быть построено с помощью алгоритма 2. Если условие $|A_1 \cup A_2|=1$ не выполнено, то алгоритм может тем не менее найти глобально оптимальное расписание, но гарантии, что оно будет найдено, в этом случае отсутствуют. Показано также, что построенное в соответствии с алгоритмом 2 расписание удовлетворяет критерию Вальда, т. е. обеспечивает минимизацию целевой функции при наихудшем стечении обстоятельств (когда все директивные сроки принимают наименьшие возможные значения). Дальнейшее развитие исследований связано с разработкой алгоритма, который гарантировал бы построение глобально оптимального расписания независимо от выполнения условия $|A_1 \cup A_2|=1$.

Список литературы

1. Lawler, E.L. Optimal sequencing of a single machine subject to precedence constraints / E.L. Lawler // Management Science. – 1973. – Vol. 19, № 5. – P. 544–546.

2. Lin, Y. Necessary and sufficient conditions of optimality for some classical scheduling problems / Y. Lin, X. Wang // European Journal of Operational Research. – 2007. – Vol. 176. – P. 809–818.

3. Wald, A. Statistical Decision Functions / A. Wald. – N. Y. : Wiley, 1950.

Поступила 01.08.12

[1]*Объединенный институт проблем информатики НАН Беларуси, Минск, Сурганова, 6*
e-mail: shafr@newman.bas-net.by

[2]*Lund University, Sweden*
e-mail: dmitry.slednev@gmail.com

Y.M. Shafransky, D.S. Sledneu

MINIMIZING MAXIMUM LATENESS FOR A SINGLE MACHINE UNDER THE DUE DATES UNCERTAINTY

The single machine problem $1|\text{prec}; d_j \in [d_j^{\min}, d_j^{\max}]|L_{\max}$ of minimizing the maximum lateness under the precedence constraints and the uncertainty of the due dates is considered. Necessary and sufficient conditions of schedule optimality are formulated for the deterministic case of the problem. For the case of uncertain due dates, necessary and sufficient conditions of the global schedule optimality are developed. We also propose an algorithm for constructing a globally optimal schedule.

OSTIS-2013

III Международная научно-техническая конференция «Открытые семантические технологии проектирования интеллектуальных систем»

Open Semantic Technologies for Intelligent Systems

21 – 23 февраля 2013, Минск, Республика Беларусь

И Н Ф О Р М А Ц И О Н Н О Е С О О Б Щ Е Н И Е

Приглашаем вас принять участие в III Международной научно-технической конференции «Открытые семантические технологии проектирования интеллектуальных систем» (OSTIS–2013).

Конференция пройдет в период с ***21 по 23 февраля 2013 г.*** в Белорусском государственном университете информатики и радиоэлектроники, Минск.

Рабочие языки конференции: русский, белорусский, английский.

ОСНОВНЫЕ ОРГАНИЗАТОРЫ КОНФЕРЕНЦИИ

- Российская ассоциация искусственного интеллекта (РАИИ)
- Белорусский государственный университет информатики и радиоэлектроники (БГУИР)
- Администрация Парка высоких технологий (Администрация ПВТ)
- Объединенный институт проблем информатики Национальной академии наук Беларуси (ОИПИ НАН Беларуси)

ДОКЛАДЫ НА КОНФЕРЕНЦИЮ БУДУТ ПРИНИМАТЬСЯ ПО СЛЕДУЮЩИМ НАПРАВЛЕНИЯМ:

1. *Базовые универсальные семантические модели представления и обработки знаний и их программная и аппаратная реализация.*
2. *Семантические технологии компонентного проектирования совместимых баз знаний, программ и пакетов программ, ориентированных на обработку знаний.*
3. *Семантические модели информационного поиска и решения задач. Семантические технологии компонентного проектирования совместимых интеллектуальных информационно-поисковых машин и решателей задач.*
4. *Семантические технологии компонентного проектирования совместимых мультимодальных и естественно-языковых интерфейсов интеллектуальных систем.*
5. *Методологии и менеджмент компонентного проектирования семантически совместимых интеллектуальных систем в рамках Open Source-проектов.*
6. *Семантически совместимые прикладные интеллектуальные системы.*

ЦЕЛЬ И ФОРМАТ ПРОВЕДЕНИЯ КОНФЕРЕНЦИИ

Целью конференции является обсуждение проблем создания комплексной технологии проектирования семантически совместимых интеллектуальных систем. Этим определяется и формат ее проведения, предполагающий последовательное проведение секций; ориентацию на «круглые столы», посвященные обсуждению различных вопросов; обеспечение возможности всем авторам не только выступить с докладами, но и продемонстрировать свои результаты на выставочных стендах.

Важнейшей задачей конференции является привлечение к ее работе не только ученых и аспирантов, но и студенческой молодежи, интересующейся проблемами искусственного интеллекта, а также коммерческих организаций, готовых сотрудничать с

научными коллективами, работающими над созданием современных технологий проектирования интеллектуальных систем.

ПОРЯДОК ПРЕДСТАВЛЕНИЯ МАТЕРИАЛОВ ДОКЛАДОВ

Материалы докладов (только по перечисленным выше направлениям) представляются в готовом для публикации виде. Текст доклада должен быть логически законченным и содержать новые научные и практические результаты.

Объем материалов доклада, включая краткий перевод, иллюстрации и список литературы, шесть или восемь полных страниц.

Переписка с авторами будет вестись только по электронной почте. Адрес электронной почты оргкомитета: ostisconf@gmail.com. Для переписки необходимо зарегистрироваться на сайте конференции.

Правила подготовки материалов докладов размещены на сайте конференции.

ФОРМИРОВАНИЕ ПРОГРАММЫ КОНФЕРЕНЦИИ

Программа конференции формируется программным комитетом по результатам рецензирования представленных материалов докладов и будет опубликована ***15 января 2013 г.*** на сайте конференции: http://conf.ostis.net.

ПУБЛИКАЦИЯ МАТЕРИАЛОВ КОНФЕРЕНЦИИ

Материалы докладов, включенных в программу конференции, печатаются в сборнике материалов конференции и публикуются на сайте конференции за неделю до начала конференции.

УСЛОВИЯ УЧАСТИЯ В КОНФЕРЕНЦИИ

Участие в конференции не предполагает организационного взноса.

СВЯЗЬ С ОРГАНИЗАТОРАМИ КОНФЕРЕНЦИИ

Сайт: http://conf.ostis.net **e-mail**: ostisconf@gmail.com

ПРАВИЛА ДЛЯ АВТОРОВ

1. Статьи принимаются в редакцию через электронную систему подачи по адресу **http://jinfo.bas-net.by** в формате файлов текстовых редакторов Microsoft Word 97 и Word 2000 для Windows. Основной текст статьи набирается с переносами шрифтом Times New Roman 11 пт, интервал между строками – одинарный, абзацный отступ 1 см, поля по 2,5 см со всех сторон.

2. Объем статьи не должен превышать 12 страниц (включая таблицы, иллюстрации, список литературы), количество иллюстраций – не больше пяти. Допускаются краткие сообщения до трех страниц.

3. Статья должна иметь индекс УДК (универсальная десятичная классификация).

4. Название статьи, фамилии всех авторов и аннотация должны быть переведены на английский язык. Для каждого из авторов приводится развернутое название учреждения с полным почтовым адресом, а также номер телефона и электронный адрес (e-mail) для связи с редакцией.

5. Формулы, иллюстрации, таблицы, встречающиеся в статье, должны быть пронумерованы в соответствии с порядком цитирования в тексте. Ссылки на рисунки и таблицы в тексте обязательны. Необходимо избегать повторения одних и тех же данных в таблицах, графиках и тексте статьи.

Рисунки должны быть представлены в виде файлов формата .cdr, .ai, .wmf, .psd, .jpg, .tif (.tiff) и выполнены с хорошим разрешением в масштабе, позволяющем четко различать надписи и обозначения. Подрисуночные подписи с расшифровкой всех позиций, представленных на рисунке, набираются шрифтом гарнитуры основного текста, размер символов 9 пт. Цветные иллюстрации печатаются только в том случае, когда это необходимо для понимания излагаемого материала.

6. Набор формул выполняется в формульных редакторах Microsoft Equation или Math Type и должен быть единообразным по применению шрифтов и знаков по всей статье.

Прямо (_) набираются: греческие и русские буквы; математические символы (sin, lg, ∞); символы химических элементов (C, Cl, $CHCl_3$); цифры (римские и арабские); векторы; индексы (верхние и нижние), являющиеся сокращениями слов.

Курсивом (~) набираются: латинские буквы – переменные, символы физических величин (в том числе и в индексе).

7. Сокращения в тексте статьи (за исключением единиц измерения) могут быть использованы только после упоминания полного термина. Единицы измерения физических величин следует приводить в Международной системе СИ.

8. Литература приводится автором общим списком в конце статьи. Ссылки на литературу в тексте идут по порядку и обозначаются цифрой в квадратных скобках. Ссылаться на неопубликованные работы не допускается. С примерами оформления библиографического описания в списке литературы можно ознакомиться в приложении 2 к *Инструкции по оформлению диссертации, автореферата и публикаций по теме диссертации* на сайте Высшей аттестационной комиссии Республики Беларусь **http://vak.org.by**.

9. Поступившие в редакцию статьи направляются на рецензирование специалистам. Основным критерием целесообразности публикации является новизна и информативность статьи. Если по рекомендациям рецензента статья возвращается автору на доработку, а переработанная рукопись вновь рассматривается редколлегией, датой поступления считается день получения редакцией ее окончательного варианта. Статьи не по профилю журнала возвращаются авторам после заключения редколлегии.

10. Статьи, направляемые на доработку, должны быть возвращены в исправленном виде с ответами на все вопросы.

11. Редакция журнала предоставляет возможность первоочередного опубликования статей, представленных лицами, которые осуществляют послевузовское обучение (аспирантура, докторантура, соискательство) в год завершения обучения.

12. Авторы несут ответственность за направление в редакцию статей, уже опубликованных ранее, или статей, принятых к публикации другими изданиями.

13. Редакция оставляет за собой право на редакционные изменения, не искажающие основное содержание статьи.

Журнал «Информатика» включен Высшей аттестационной комиссией Республики Беларусь в список научных изданий для опубликования результатов диссертационных исследований.

Индексы

00827

для индивидуальных подписчиков

008272

для предприятий и организаций